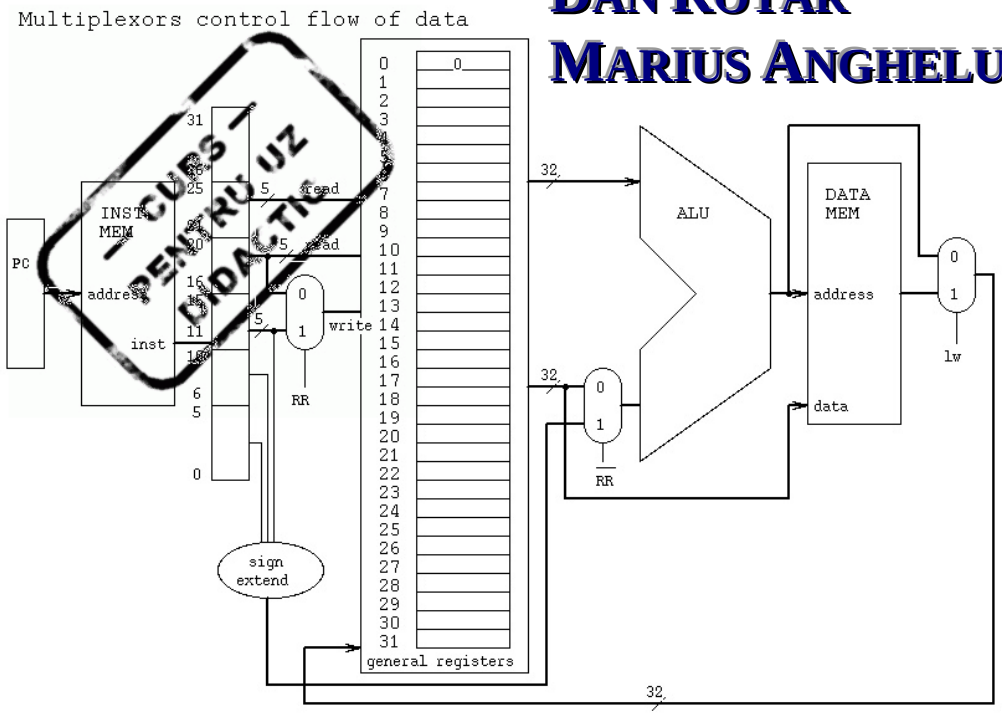
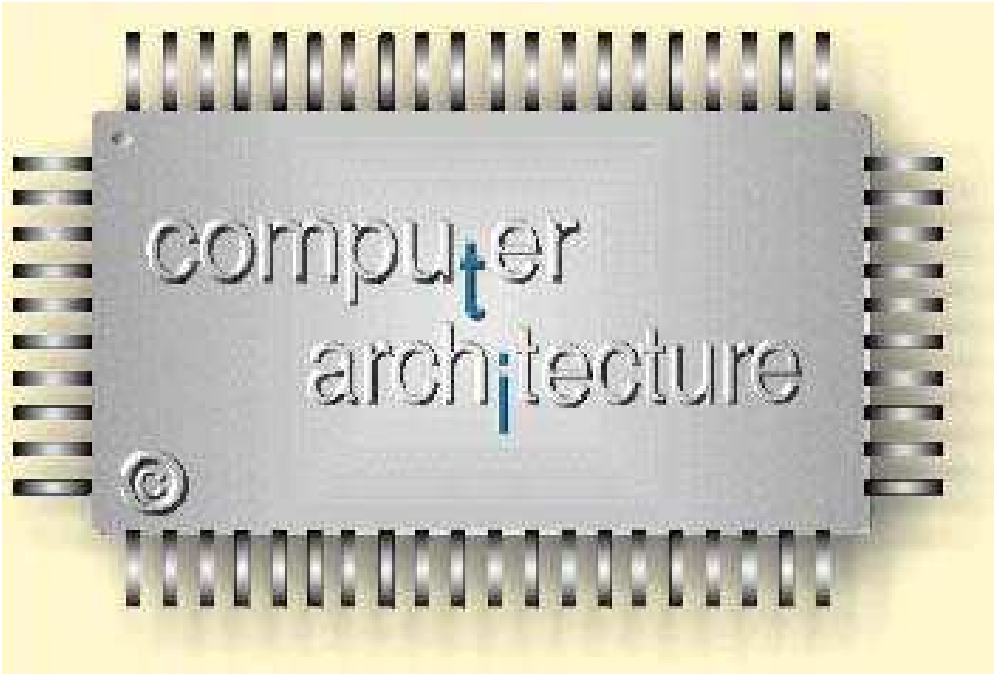
MINISTERUL EDUCAȚIEI
UNIVERSITATEA DIN BACĂU
FACULTATEA DE INGINERIE

DAN ROTAR
MARIUS ANGHELUȚ





EDITURA ALMA MATER
(actualizat şi revizuit 2013)

Cuprins

CAPITOLUL 1

BAZE DE NUMERAȚIE

1.1. Introducere

Valorile numerelor pot fi exprimate în diferite baze de numerație. Astfel, în activitățile umane obişnuite baza de numerație folosită este baza zece. Nu acelaşi lucru se întâmplă în sistemele numerice. Din considerente tehnologice, în sistemele numerice nu putem reprezenta decât două numere: zero şi unu. De regulă valoarea zero este asociată cu un nivel scăzut de tensiune iar valoarea unu este asociată cu un nivel ridicat de tensiune, diferența dintre tensiunea asociată valorii zero şi cea asociată valorii unu asigurând securitatea la perturbații a sistemului numeric (în sensul că dacă diferența între cele două tensiuni creşte, creşte şi imunitatea la perturbații a sistemului numeric).

O bază de numerație presupune existența unui număr de simboluri (numite adesea cifre) cu ajutorul cărora vor fi reprezentate numerele, un mod de scriere a numerelor şi o relație de calcul a valorii numărului. Numărul simbolurilor utilizate într-o bază de numerație defineşte numele bazei de numerație. Reprezentarea numerelor se face în prezent în exclusivitate prin scrierea poziționlă adică poziția numărului exprimă şi rangul acestuia (exponentul bazei de numerație). Simbolul (cifra) cel mai din dreapta reprezentării numărului are rangul minim (zero) iar simbolul (cifra) cel mai din stânga din reprezentarea numărului are rangul cel mai mare. Calculul valorii numărului se face prin înmulțirea valorii simbolului cu baza la puterea rangului. Atunci când se lucrează cu mai multe baze de numerație simultan este obligatorie indicarea bazei de numerație la fiecare număr scris. Acest lucru se face prin scrierea valorii bazei de numerație ca indice. De exemplu: 12_{10} sau 100110101_2 sau $1B2C_{16}$. Pentru indicarea bazei de numerație pot fi utilizate şi litere astfel:

z – (zecimal) pentru baza 10
b – (binar) pentru baza 2
o – (octal) pentru baza 8
h – (hexazecimal) pentru baza 16.

Exemplul de mai sus poate fi scris şi: : 12_z sau 100110101_b sau $1B2C_h$.

În continuare vor fi prezentate principalele baze de numerație utilizate în sistemele numerice.

1.2. Baza de numerație zece

Așa cum s-a arătat, baza de numerație zece, cea utilizată în activitățile umane, presupune existența a zece simboluri distincte pentru reprezentarea numerelor. Aceste simboluri sunt:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Reprezentarea numerelor este poziţională deci atunci când scriem un număr, poziția acestuia va reprezenta rangul său:

rang: 4 3 2 1 0
simbol: 7 5 2 1 3

Valoarea numărului reprezentat mai sus este: şapte zeci şi cinci de mii două sute treisprezece. Aceasta valoare a rezultat din regula prezentată mai sus adică:

valoare număr = $7 \times 10^4 + 5 \times 10^3 + 2 \times 10^2 + 1 \times 10^1 + 3 \times 10^0$

1.3. Baza de numerație doi

Numerele reprezentate în baza doi sunt numite în mod obişnuit numere binare. În baza doi sunt necesare numai două simboluri pentru reprezentarea numerelor şi acestea sunt: 0 (zero) şi 1 (unu). Datorită faptului că sistemele numerice lucrează, aşa cum s-a arătat mai sus, în sistem binar, studiul reprezentării numerelor în baza doi este important atunci când vorbim de tehnologia DSP. De asemenea prin asocierea unor valori de adevăr celor două simboluri: 0 – fals (false) şi 1 – adevărat (true) sistemele numerice vor putea lucra în logică binară sau Booleană. Abordarea logicii binare va fi făcută mai târziu în cuprinsul acestui manual.

TABELUL 1.1.

Număr binar	Număr zecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Rezenind la scrierea numerelor în baza doi trebuie spus faptul că toate regulile stabilite la baza zece se aplică şi aici. De exemplu, dacă vom scrie numărul binar:

rang 6 5 4 3 2 1 0
număr: 1 0 1 1 0 0 1

atunci valoarea acestuia va fi:

valoare = $1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 +$
$+ 0 \times 2^1 + 1 \times 2^0 = 64_z + 16_z + 8_z + 1_z = 89_z$

În tabelul 2.1 este prezentată echivalența primelor

16 numere binare cu cele zecimale. Pentru acest tabel putem face următoarele observaţii:

- daca citim numerele binare din acest tabel pe coloană observăm faptul că succesiunea cifrelor zero şi unu depinde de rangul cifrei. Astfel la rangul zero întâlnim succesiunea 0,1,0,1,0,1, ... , la rangul unu succesiunea 0,0,1,1,0,0,1,1, ... , şi aşa mai departe;
- un număr zecimal putere a lui doi va determina o cifră binară cu un singur simbol unu pe poziţia exponentului puterii lui doi.

Este evident faptul ca atunci când se doreşte conversia numerelor dintr-o bază în alta este dificil de utilizat tabele, în special atunci când se lucrează cu numere mari. Din acest motiv se stabilesc anumiţi algoritmi de conversie pentru simplificarea translatării numerelor dintr-o bază în alta.

1.3.1. Conversia binar-zecimală

Conversia unui numar binar întreg în echivalentul său zecimal, cunoscând că operaţiile se efectuează în sistemul zecimal, se face cu ajutorul metodei înmulţirii repetate cu 2. Pentru a justifica, se consideră numărul binar ***P*** exprimat prin:

$$P = b_{n-1}\,2^{n-1} + b_{n-2}\,2^{n-2} + ... + b_1\,2^1 + b_0\,2^0$$

sau:

$$P = (\{[\overbrace{(b_{n-1}\,2 + b_{n-2})}^{P_1}\,2 + b_{n-3}]\,2 + b_{n-4}\}\,2 + ... + b_1)\,2 + b_0$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{P_2}$$

relaţie din care rezultă algoritmul conversiei numerelor întregi binare în numere zecimale:

- pentru a forma mărimea intermediară P_1 se înmulţeşte cu 2 cifra cea mai semnificativă (cifra de rang maxim) a numărului binar, adăugând următoarea cifră semnificativă;
- pentru a obţine mărimea intermediară P_2, se înmulţeşte P_1 cu 2 adăugând cea de-a treia cifră semnificativă;
- se continuă acest algoritm până la adăugarea cifrei cel mai puţin semnificative a numărului binar (cea de rang minim);
- numărul zecimal obţinut este echivalentul zecimal al numărului binar dat.

EXEMPLU

Să se transforme numărul binar 10110110111 în echivalentul său zecimal:

1 x 2 + 0 = 2	$P_1 = 2$
2 x 2 + 1 = 5	$P_2 = 5$
5 x 2 + 1 = 11	$P_3 = 11$
11 x 2 + 0 = 22	$P_4 = 22$
22 x 2 + 1 = 45	$P_5 = 45$
45 x 2 + 1 = 91	$P_6 = 91$
91 x 2 + 0 = 182	$P_7 = 182$
182 x 2 + 1 = 365	$P_8 = 365$
365 x 2 + 1 = 731	$P_9 = 731$
731 x 2 + 1 = 1463	$P_8 = 1463$

deci: $10110110111_B = 1463_Z$

Pentru conversia binar-zecimală a unui număr binar fracționar se folosește metoda împărțirii repetate cu 2. Pentru a justifica, se consideră numărul binar fracționar ***Q*** dat de:

$$Q = b_{-1}\,2^{-1} + b_{-2}\,2^{-2} + ... + b_{-m}\,2^{-m}$$

care poate fi pus şi sub forma:

$$Q = 2^{-1}\,(b_{-1} + 2^{-1}\,\{\,b_{-2} + 2^{-1}\,[b_{-3} + ... + 2^{-1}\,\overbrace{(b_{-m+1} + 2^{-1}b_{-m})}^{Q_1}]\})$$

de unde rezultă algoritmul conversiei numerelor binare fracționare în numere zecimale:

- pentru a obține mărimea intermediară Q_1 se împarte cifra cea mai puțin semnificativă (de rang minim) cu 2, adăugând urmatoarea cifră semnificativă;
- pentru a obține mărimea intermediară Q_2 se împarte Q_1 cu 2 adăugând a treia – a treia cifră de la dreapta spre stânga – cifră semnificativă;
- se continuă acest algoritm până când împărțirea care corespunde cifrei 0 de la stânga virgulei a fost efectuată;
- numărul obținut este echivalentul zecimal al numărului binar dat.

EXEMPLU

Să se transforme numărul binar fracționar 0,001101111 în echivalentul său zecimal:

1	: 2 + 1 = 1,5	$Q_1 = 1,5$
1,5	: 2 + 1 = 1,75	$Q_2 = 1,75$
1,75	: 2 + 1 = 1,875	$Q_3 = 1,875$
1,875	: 2 + 0 = 0,9375	$Q_4 = 0,9375$
0,9375	: 2 + 1 = 1,46875	$Q_5 = 1,46875$
1,46875	: 2 + 1 = 1,734375	$Q_6 = 1,734375$
1,734375	: 2 + 0 = 0,8671875	$Q_7 = 0,8671875$

0,8671875	: 2 + 0 = 0.4335937	$Q_8 = 0,4335937$
0,4335937	: 2 + 0 = 0,2167968	$Q_9 = 0,2167968$

deci: $0,001101111_B = 0,2167968_Z$

Dacă se cere conversia unui număr binar care are atât parte întreagă cât şi parte fracționară în echivalentul său zecimal, se aplică părții întregi algoritmul corespunzător conversiei numerelor întregi, iar părții fracționare algoritmul corespunzător părții fracționare.

1.3.2. Conversia zecimal-binară

Conversia zecimal-binară a numerelor întregi se face după metoda împărțirii repetate prin 2. Pentru a justifica aceasta, se consideră numărul intreg ***N*** în baza 10, care poate fi exprimat în funcție de puterile lui 2 sub următoarea formă :

$$N = a_{n-1}\, 2^{n-1} + a_{n-2}\, 2^{n-2} + ... + a_1\, 2^1 + a_0\, 2^0$$

sau:

$$N = (a_{n-1}\, 2^{n-2} + a_{n-2}\, 2^{n-3} + ... + a_1)\ 2 + a_0 = N_1\, 2 + a_0$$

Din ultima egalitate rezultă că cifra cea mai semnificativă a_0 din reprezentarea binară a numărului N constituie restul împărțirii lui N cu 2.
În mod analog, numărul N_1 se poate exprima prin :

$$N = (a_{n-1}\, 2^{n-3} + a_{n-2}\, 2^{n-4} + ... + a_2)\ 2 + a_1 = N_2\, 2 + a_1$$

Adică a_1, care reprezintă cifra semnificativă de rang imediat următor în reprezentare binară, constituie restul împărțirii lui N_1 cu 2.

De aici rezultă algoritmul conversiei numerelor întregi zecimale în numere binare:

- se împarte numărul intreg în baza zece N prin 2; se obține câtul N_1 şi restul a_0;
- se împarte câtul N_1 prin 2; se obține câtul N_2 şi restul a_1;
- se continuă această operație până se ajunge la un cât N_n egal cu zero;
- resturile obținute sunt cifrele numărului binar (biții), a_0 fiind cifra cea mai puțin semnificativă, a_1 cifra următoare ş. a. m. d.

EXEMPLU

Să se transforme numărul zecimal 53 în echivalentul său binar.

53 : 2 = 26 + 1	$a_0 = 1$
26 : 2 = 13 + 0	$a_1 = 0$

13 : 2 = 6 + 1	$a_2 = 1$
6 : 2 = 3 + 0	$a_3 = 0$
3 : 2 = 1 + 1	$a_4 = 1$
1 : 2 = 0 + 1	$a_5 = 1$

Deci $53_{10} = 110101_2$

Folosind acelaşi algoritm se prezintă un alt mod de aranjare a calculelor, în aşa fel, ca la sfârşitul operaţiei să se obţină direct numărul binar fără a fi necesară rescrierea sa. În acest sens, se aşează succesiunea calculelor după schema prezentată mai jos, unde al *n*-lea cât este ultimul cât, egal cu zero :

Cât n ← Cât (n –1) ← Cât (n –2) ← .... Cât 2 ← Cât 1 ← Număr
Rest n Rest (n-1) .... Rest 3 Rest 2 Rest 1

Considerând acelaşi exemplu se obţine :

0	1	3	6	13	26	53
	1	1	0	1	0	1

Conversia zecimal-binară a numerelor fracţionare se face dup metoda înmulţirii repetate cu 2. Pentru a justifica, se consideră numărul fracţionar zecimal ***M***, care poate fi exprimat în funcţie de puterile numărului 2 prin :

$$M = a_{-1} 2^{-1} + a_{-2} 2^{-2} + ... + a_{-m} 2^{-m}$$

Prin înmulţirea ambelor părţi ale ecuaţiei de mai sus cu 2, se obţine :

$$2 M = a_{-1} + (a_{n-2} 2^{-1} + ... + a_{-m} 2^{-m+1}) = a_{-1} + M_1$$

partea dreaptă a egalităţii fiind formată din numărul întreg a_{-1}, care reprezintă cifra cea mai semnificativă a numărului binar subunitar şi fracţia M_1. Aplicând acelaşi procedeu lui M_1, rezultă:

$$2 M_1 = a_{-2} + (a_{n-3} 2^{-1} + ... + a_{-m} 2^{-m+2}) = a_{-2} + M_2$$

adică se obţine următoarea cifră semnificativă a_{-2} a numărului fracţionar binar şi fracţia M_2.

Cu aceasta, algoritmul conversiei numerelor fracţionare zecimale în numere fracţionare binare este următorul:

- se înmulţeşte numărul zecimal fracţionar M cu 2; rezultă bitul a_{-1} şi partea fracţionară M_1;
- se înmulţeşte partea fracţionară M_1 cu 2; rezultă bitul a_{-2} şi partea fracţionară M_2;

- se contiună acestă operație până când M_m devine egal cu zero sau ne limităm la un număr de cifre binare în funcție de precizia impusă.

EXEMPLU

1. Să se transforme numărul 0,40625 în echivalentul său binar.

0,40625 x 2 = 0,81250	$a_{-1} = 0$
0,8125 x 2 = 1,6250	$a_{-2} = 1$
0,625 x 2 = 1,250	$a_{-3} = 1$
0,25 x 2 = 0,5	$a_{-4} = 0$
0,5 x 2 = 1,0	$a_{-5} = 1$

deci $0,40625_z = 0,01101_b$

2. Să se transforme numărul zecimal 0,7 în echivalentul său binar.

0,7 x 2 = 1,4	$a_{-1} = 1$
0,4 x 2 = 0,8	$a_{-2} = 0$
0,8 x 2 = 1,6	$a_{-3} = 1$
0,6 x 2 = 1,2	$a_{-4} = 1$
0,2 x 2 = 0,4	$a_{-5} = 0$
0,4 x 2 = 0,8	$a_{-6} = 0$
.....	

deci $0,7_{10} = 0,101100...._2$

1.4. Baza de numerație opt (sistemul octal)

Utilizatorii primei generații de calculatoare au întâmpinat dificultăți în manevrarea cifrelor binare, întrucât un număr exprimat în sistemul binar are o lungime de circa trei ori mai mare decât în exprimarea zecimală. Din acest motiv, s-a căutat un sistem de numerație mai apropiat de baza zece, cu o contingență directă cu sistemul binar. Cunoscând că 8 este o putere întreagă a lui 2 ($2^3 = 8$), aceste calități sunt întrunite de sistemul de numerație octal, care are opt cifre pentru exprimare, de la 0 la 7 inclusiv. Ca urmare, conversia binar octală şi octal binară se poate face direct, în primul caz înlocuind un grup de trei cifre binare socotite de la dreapta şi de la stânga virgulei prin echivalentul lor octal, iar în al doilea caz înlocuind fiecare cifră octală printr-un grul de trei cifre binare.

EXEMPLU

1. Să se transforme numărul binar 10111,00 11 în echivalentul său octal.

$$\underbrace{010}_{2}\ \underbrace{111}_{7},\ \underbrace{001}_{1}\ \underbrace{100}_{4}{}_2 = 27,14_8$$

2. Să se transforme numărul octal 26,341 în echivalentul său binar.

$$26{,}341_8 = \underbrace{010}_{2}\ \underbrace{110}_{6},\ \underbrace{011}_{3}\ \underbrace{100}_{4}\ \underbrace{001}_{1}{}_2$$

Prin folosirea sistemului octal, calculele binare sau structura informației din interiorul calculatorului nu se schimbă, structură care poate fi verificată cu uşurință din exterior. Din acest motiv, calculatoarele din prima şi a doua generație aveau o lungime a cuvântului egală cu un multiplu a lui 3, obişnuit 24, 36 sau 42 de biți, pentru a facilita trecerea din sistemul octal în binar sau invers.

Se menționează că pentru transformarea numerelor din baza de numerație 10 în baza de numerație 8 sau invers, cu calculele făcute în baza 10, toți algoritmii deduşi anterior rămân valabili, cu observația că înmulțirile, respectiv împărțirile prin 2 se înlocuiesc cu înmulțiri, respectiv împărțiri, cu 8.

1.5. Baza de numerație şaisprezece (sistemul hexazecimal)

Pentru a reprezenta în calculator alte caractere decât numere, adică litere, semne de punctuație etc. (caractere alfanumerice), au fost utilizate inițial 6 cifre binare. În cursul operațiilor de citire sau imprimare, dispozitivul respectiv făcea automat transformarea fiecărui caracter în cele 6 cifre binare sau invers.

Folosind 6 cifre binare, adică două cifre octale, nu puteau fi reprezentate decât 64 de caractere distincte. Acest număr de caractere s-a dovedit a fi în scurt timp insuficient pentru aplicațiile practice şi din acest motiv sa-a trecut, la majoritatea calculatoarelor generației a treia, la alocarea a 8 cifre binare, adică două hexazecimale, fiecărui caracter. Un grup de 8 cifre binare este cunoscut şi sub denumirea de *octet* sau *bait* şi reprezintă cea mai mică diviziune a informației care poate fi prelucrată în aceste calculatoare.

Întrucât baza de numerație 16 este mai mare ca baza de numerație 10, cifrele mai mari de 9 sunt notate în ordine cu literele alfabetului latin, adică :

A – zece
B – unsprezece
C – doisprezece
D – treisprezece
E – paisprezece
F – cincisprezece.

Pentru conversia din baza 10 în 16 sau invers, sunt utilizate aceleaşi metode, făcându-se calculele în baza 10.

Conversia binar-hexazecimală şi hexazecimal-binară se face în mod similar celei binar-octale, respectiv octal-binare, cu observația că se vor lua grupe de câte 4 cifre binare în loc de 3.

CAPITOLUL 2

OPERAȚII ARITMETICE

2.1. Introducere

În acest capitol se vor prezenta operațiile aritmetice simple realizate cu numere binare (numere reprezentate în baza doi), numere octale (numere reprezentate în baza opt) sau hexazecimale (numere reprezentate în baza saisprezece).

Pentru exemplificarea operațiilor efectuate se vor folosi numai numere întregi pozitive (numere naturale) deoarece acesta este modul de bază al reprezentării informației într-un sistem de calcul numeric.

2.2. Operații aritmetice cu numere binare

2.2.1. Adunarea

Operația de adunare poate fi descrisă cu ajutorul celor patru combinații posibile între două numere binare:

0 + 0 = 0; transport la rangul superior: 0
0 + 1 = 1; transport la rangul superior: 0
1 + 0 = 1; transport la rangul superior: 0
1 + 1 = 0; transport la rangul superior: 1

În mod obişnuit se scrie: $1_b + 1_b = 10_b$ deoarece în această situație avem un transport la rangul superior. Este evident faptul că: $10_b = 2_z$.

EXEMPLE

1) Să se adune numărul binar 110111 cu numărul binar 101.

```
transport      111
             110111 +
                101
            ------
             111100
```

putem efectua verificarea in zecimal: $110111_b = 55_z$, $101_b = 5_z$, $111100_b = 60_z$.

Verificarea este imediată. Se observă că dacă trebuie să adunăm un sir de ***n*** cifre unu în binar, rezultatul pe rangul respectiv va fi zero dacă numărul de cifre unu adunate este par sau unu dacă numărul de cifre adunate este impar iar transportul se face peste

n:2 (împărțire întreagă) ranguri pornind de la rangul respectiv (cel la care se efectueaza adunarea).

2) Să se adune numărul binar 11111 cu numărul binar 1.

```
transport      1111
               11111 +
                   1
              ------
              100000
```

De fapt rezultatul poate fi văzut imediat dacă ne uităm la tabelul 1.1. Dupa un numar cu o succesiune de cifre unu urmează un număr binar care are o cifră unu urmat de o succesiune de cifre zero. Numărul cifrelor zero este egal cu numărul cifrelor unu al numărului care se adună. De exemplu $1111_b + 1_b = 10000_b$.

2.2.2. Scăderea

Operația de scădere poate fi descrisă cu ajutorul celor patru combinații posibile între două numere binare:

0 – 0 = 0; împrumut de la rangul superior: 0
0 – 1 = 1; împrumut de la rangul superior: 1
1 – 0 = 1; împrumut de la rangul superior: 0
1 – 1 = 0; împrumut de la rangul superior: 0

În mod obişnuit se scrie: $10_b – 1_b = 1_b$ deoarece în această situație avem un împrumut de la rangul superior. Este evident faptul că: $10_b = 2_z$.

EXEMPLE

1) Să se efectueze scăderea numerelor binare: 11001 şi 11.

```
împrumut       11
               11001 –
                  11
              ------
               10110
```

Verificarea in zecimal: $11001_b = 25_z$, $11_b = 3_z$, $10110_b = 22_z$. $25_z – 3_z = 22_z$.

2) Să se efectueze scăderea 11001 – 1011.

```
împrumut       111
               11001 –
                1011
              ------
               01110
```

Verificarea in zecimal: $11001_b = 25_z$, $1011_b = 11_z$, $1110_b = 14_z$. $25_z – 11_z = 14_z$.

Se pot face observații similare cu cele de la adunare.

3) Sa se scadă din 1000_b valoarea 1. Răspunsul este imediat: 111_b.

2.2.3. Înmulțirea

Operația de înmulțire poate fi descrisă cu ajutorul celor patru combinații posibile între două numere binare:

0 x 0 = 0; 0 x 1 = 0; 1 x 0 = 0; 1 x 1 = 1.

Operația de înmulțire se efectuează identic cu cea din baza zece. Vom analiza în continuare câteva exemple.

EXEMPLE

1) Să se efectueze înmulțirea în binar: 110101 x 1101.

```
  110101 x
    1101
  ------
  110101
 000000
110101
110101
----------
1010110001
```

Verificarea în zecimal: $110101_b = 53_z$, $1101_b = 13_z$, $1010110001_b = 689_z$. Rezultă 53_z x $13_z = 689_z$.

Din acest exemplu se observă că operația de înmulțire poate fi înlocuită cu operatii de deplasare a deînmulțitului spre stânga şi adunarea rezultatelor obținute. Pentru aceasta se parcurge înmulțitorul de la dreapta la stânga cifră cu cifră. Pentru fiecare cifră unu a înmulțitorului se scrie deînmulțitul deplasat spre stânga cu un număr de cifre egal cu rangul cifrei unu a înmulțitorului. La final se sumează rezultatele obținute.

2) Să se efectueze înmulțirea numărului 101101 cu numărul 1101.
Vom aplica metoda deplasării spre stânga a deînmulțitului şi adunarea şirurilor obținute.

deînmulțitul deplasat la stânga cu 0 paşi	101101
deînmulțitul deplasat la stânga cu 2 paşi	101101
deînmulțitul deplasat la stânga cu 3 paşi	101101
rezultatul (suma)	1001001001

Verificarea în zecimal: $101101_b = 45_z$, $1101_b = 13_z$, $1001001001_b = 585_z$.
Rezultă: 45_z x $13_z = 585_z$.

2.2.4. Împărțirea întreagă

Operația de împărțire poate fi descrisă cu ajutorul celor patru combinații posibile între două numere binare:

0 : 0 = 0; 0 : 1 = 0; 1 : 0 = imposibil; 1 : 1 = 1.

Operația de împărțire se efectuează identic cu cea din baza zece. Vom analiza în continuare câteva exemple.

EXEMPLE

1) Să se efectueze împărțirea numărului 11010110101 cu 1011.

```
11010110101|1011
 1011      10011100
 0010011
   1011
   10000
    1011
    001011
      1011
      000001
```

Rezultatul este: 10011100 rest 1. Verificarea în zecimal: $11010110101_b = 1717_z$, $1011_b = 11_z$, $10011100_b = 156_z$. $1717_z : 11_z = 156_z$ rest 1.

Din acest exemplu se observă faptul că operația de împărțire întreagă se reduce la operații succesive de deplasare la dreapta a împărțitorului şi scăderi din deîmpărțit, începând de la cifrele de rang maxim ale deîmpărțitului. Se contiunuă cu deîmpărțitul nemodificat dacă rezultatul scăderii este negativ sau cu rezultatul scăderii dintre deîmpărțit şi împărțitor dacă rezultatul scăderii este zero sau pozitiv. La fiecare scădere se notează cifra cîtului care este unu dacă rezultatul scăderii este zero sau pozitiv (scăderea se poate efectua) sau zero dacă rezultatul scăderii este negativ (scăderea nu se poate efectua). Operația de scădere se opreşte când se efectuează o scădere din bitul de rang zero al deîmpărțitului. Valoarea deîmpărțitului rămasă constituie restul împărțirii.

2) Să se efectueze împărțirea dintre numerele 111101101 si 101.
Vom apela la procedeul deplasării la dreapta a împărțitorului şi scăderea din deîmpărțit.

```
111101101
101
010
```

- rezultatul scăderii este pozitiv deci se continuă cu valoarea rămasă în urma scăderii şi se face o deplasare la dreapta; cît = 1

010101101
 101
0000 - rezultatul scăderii este zero deci se continuă cu valoarea rămasă în urma scăderii şi se face o deplasare la dreapta; cît = 11

000001101
 101 - rezultatul scăderii este negativ deci nu se efectuează scăderea şi se face deplasarea dreapta a împărțitorului; cît = 110

000001101
 101 - rezultatul scăderii este negativ deci nu se efectuează scăderea şi se face deplasarea dreapta a împărțitorului; cît = 1100

000001101
 101 - rezultatul scăderii este negativ deci nu se efectuează scăderea şi se face deplasarea dreapta a împărțitorului; cît = 11000

000001101
 101
 001 - rezultatul scăderii este pozitiv deci se continuă cu valoarea rămasă în urma scăderii şi se face o deplasare la dreapta; cît = 110001

000000011
 101 - rezultatul scăderii este negativ deci nu se efectuează scăderea; cît = 1100010 iar restul este 11 deoarece s-a incercat scăderea din cifra de rang zero a deîmparțitului.

Verificarea în zecimal: $111101101_b = 493_z$, $101_b = 5_z$, $1100010_b = 98_z$, $11_b = 3_z$.
$493_z : 5_z = 98_z$ rest 3_z.

2.3. Operații aritmetice cu numere reprezentate în octal şi hexazecimal

Reprezentarea numerelor în octal sau hexazecimal se face, asa cum s-a arătat, în scopul cresterii clarității reprezentării valorilor numerice. Trebuie subliniat aici, încă o dată, că în sistemele numerice de calcul, singurul mod de reprezentare a informației este cel binar. Utilizarea altor baze de numerație cum sunt cel octal sau cel hexazecimal se face doar în scopul creşterii clarității documentației însoțitoare. Baza de numerație opt şi baza de numerație şaisprezece au avantajul că au baze puteri a lui doi şi din acest motiv permit scrierea condensată şi comodă a numerelor binare.

Toate operațiile efectuate în altă bază de numerație decât baza zece se bazează pe principii similare, aşa cum s-a văzut şi la descrierea operațiilor aritmetice în baza doi.

Este important de reținut faptul că atunci când rezultatul adunării depăşeşte ca valoare valoarea bazei se produce un transport în rangurile superioare iar cînd se efectuează scăderea unui număr mai mare dintr-un număr mai mic se produce un împrumut de la rangurile superioare (identic cu ceea ce deja ştim de la operațiile în baza zece).

Cele mai importante operații efectuate în octal sau în hexazecimal sunt cele de adunare şi de scădere. Din acest motiv vom exemplifica pe scurt, în continuare modul în care se efectuează aceste operații în cele două baze de numerație.

2.3.1. Adunarea şi scăderea în octal

Adunarea, exemple

1) Să se adune numerele reprezentate în baza opt 723_o cu 523_o.

transport 1
723_o +
523_o
1446_o

În rangul doi, la adunarea cifrelor 7_o cu 5_o se produce un transport deoarece $7_z + 5_z = 12_z$ este o valoare mai mare decât valoarea bazei care este 8. În acest caz se procedează astfel: se scade din valoarea obținută valoarea bazei $12_z - 8_z = 4_z$, valoarea respectivă se scrie pe rangul doi şi avem un transport la rangul 3.
Trebuie ținut cont de faptul că în octal: $7_o + 1_o = 10_o$.

2) Să se efectueze adunarea în octal $375_o + 276_o$.

transport 11
375_o +
276_o
673_o

Scăderea, exemple

1) Să se efectueze scăderea în octal: $532_o - 251_o$.

împrumut 1
532_o –
251_o
261_o

La scăderea de pe rangul unu: $3_o - 5_o$, trebuie să se efectueze un împrumut din rangul superior. O cifră împrumutată din rangul superior înseamna opt unități în rangul inferior. Deci, în urma împrumutului, avem: $8_z + 3_z - 5_z = 6_z$.

2) Să se scadă în octal din $453_o - 264_o$.

împrumut 11
453_o –
264_o
167_o

Vrificarea în zecimal este imediată dacă se face conversia din baza opt în baza zece utilizând relația:

valoare zecimală = $o_{n-1}\,8^{n-1} + o_{n-2}\,8^{n-2} + ... + o_1\,8^1 + o_0\,8^0$

unde: o_0, o_1, o_2, …, o_n sunt cifrele numărului în baza opt.

2.3.2. Adunarea şi scăderea în hexazecimal

Având în vedere faptul că operațiile sunt similare cu cele din octal, cu excepția faptului că baza are valoarea şaisprezece, vom prezenta numai câte un exemplu pentru adunare şi unul pentru scădere.

Adunare, exemplu

Sa se adune numerele în hexazecimal: $2A57_h + 57B9_h$

transport	111
	$2A57_h$ +
	$57B9_h$
	8210_h

În acest caz depăşirea se produce când suma este mai mare decât şaisprezece. De exemplu, $7_z + 9_z = 16_z = 10_h$

Scădere, exemplu

Să se efecueze scăderea: $5C2B_h - 3ACF_h$.

împrumut	11
	$5C2B_h$ –
	$3ACF_h$
	$215C_h$

În cazul împrumutului, de pe rangul superior se împrumuta şaisprezece unități. De exemplu, în cazul $B_h - F_h$, avem un împrumut şi se obține: $16_z + 11_z$ (B_h) – 15_z (F_h) = 12_z (C_h).

CAPITOLUL 3

ARHITECTURA SISTEMELOR DE CALCUL

3.1. Introducere

Un sistem de calcul reprezintă o maşină automată destinată prelucrării informaţiilor. O astfel de maşină va interacţiona cu mediul extern pentru preluarea informaţiilor de intrare şi furnizarea informaţiilor rezultate în urma prelucrării informaţiei de intrare. Prelucrarea informaţiei se face într-un anumit scop ceea ce determină o anumită activitate internă a sistemului de calcul. În acest moment putem distinge mai multe faze ale prelucrării informaţiei de către un sistem de calcul.

În primul rând informaţia prezentată la intrarea sistemului de calcul reprezintă în, accepţia generală, un semnal. Acest semnal are un suport fizic şi este reprezentat de o mărime electrică, mecanică, optică etc. O astfel de mărime este descrisă de anumiti parametrii măsurabili şi are un suport energetic care permite interacţiunea cu această mărime. O astfel de mărime trebuie să fie transformată (adaptată) în aşa fel încât aceasta să fie compatibilă cu formatul de intrare acceptat de sistemul de calcul.

În al doilea rând informaţia o data prelucrată de sistemul de calcul trebuie furnizată mediului extern într-un anumit format impus de aplicaţia pentru care se efectuează prelucrarea informaţiei. Este limpede că va fi necesară o nouă trensformare a informatiei din formatul furnizat la ieşire de către sistemul de calcul în formanul impus de aplicaţie.

În sfârşit, în afară de cele două faze ce implică transformări ale suportului informaţiei, putem distinge şi a treia fază, cea a prelucrării interne, în sistemul de calcul a informaţiei.

Pentru a asigura o flexibilitate cât mai mare a unui astfel de sistem, cele trei faze ale prelucrării informaţiei vor fi asigurate de către elemente diferite. Astfel, o anumită modificare ce s-ar impune pe parcurs nu va afecta deât un anumit element din structura sistemului.

În prezent, cele trei faze ale prelucrării informaţiei, sunt realizate astfel:

- transformarea semnalelor într-un format compatibil cu cel acceptat de intrarea sistemului de calcul şi transformarea informaţiei furnizate de sistemul de calcul în semnale necesare aplicaţiei, este realizată de către dispozitivele periferice. Acestea, deşi sunt strâns legate de funcţionarea sistemului de calcul sunt considerate elemente externe sistemului de calcul;
- prelucrarea internă în sistemul de calcul a informaţiei va fi faza de prelucrare a informaţiei care determină o anumită structură a sistemului de calcul. Această structură va fi completată cu elemente de legătură cu dispozitivele periferice, elemente ce poartă denumirea generică de *interfeţe*.

Pentru început ne vom ocupa de structura internă a sistemului de calcul ce constituie arhitectura internă a acestuia după care se vor discuta câteva aspecte legate de periferice întregul reprezentând arhitectura unui sistem de calcul.

Din punct de vedere al tipului informației prelucrate în sistemul de calcul, aceste pot fi de două tipuri:

- *calculatoarele analogice* care prelucrează semnale continui în timp, adică semnale ce sunt descrise de o funcție continuă de timp;
- *calculatoare numerice* ce prelucrează numere, adică informația prezentată în format numeric.

Între aceste două tipuri de calculatoare există deosebiri esențiale deşi amândouă sunt destinate prelucrării informației. Primul aspect se referă la tipul informației prelucrate. Dacă la calculatoarele analogice informația trebuie să aibă un anumit nivel al suportului energetic pentru ca aceasta să poată fi prelucrată, la calculatoarele numerice informația este lipsită de suportul energetic fiind reprezentată de elemente abstracte cum sunt numerele. Acest lucru aduce o serie de avantaje importante în favoarea calculatoarelor numerice ceea ce face ca utilizarea acestora să fie din ce în ce mai răspândită.

Pentru a lămuri lucrurile vom considera un exemplu simplu. Un instrument de măsură a tensiunii (voltmetru) poate fi considerat ca un sistem de calcul foarte simplu. Un astfel de instrument preia o informație dein mediul extern (tensiunea electrică), o adaptează la formatul acceptat la intrare (de regulă printr-un divizor de tensiune), prelucrează informația (prin comparare cu un element etalon) şi furnizează la ieşire o informație compatibilă cu aplicația (în cazul cel mai simplu afişând rezultatul măsurătorii într-un anumit fel). În cazul unui voltmetru analogic (cu instrument de măsură) informația de intrare trebuie să furnizeze o anumită energie pentru ca măsurătoarea să poată fi efectuată (această energie preluată de la semnalul măsurat va determina o deplasare a acului indicator proporțională cu mărimea măsurată). În cazul unui voltmetru numeric prelucrarea informației se face în principal prin transformarea acesteia într-o valoare numerică. Din acest motiv energia preluată de la semnalul de intrare va fi cu mult mai mică (această energie nemaijucând nici un rol în prelucrarea semnalelor). Vom aminti numai două din concluziile ce pot fi desprinse din acest exemplu: măsurătoarea efectuată cu un instrument numeric este mai precisă prin scăderea cantității de energie preluate de la semnalul de intrare (cunoscându-se faptul că preluarea unei anumite cantități de energie din semnalul de măsurat duce la alterarea acestuia) şi mărimile reprezentate prin numere cum sunt cele de la voltmetrul numeric nu pot fi influențate de condițiile de mediu (temperatură, presiune, umiditate etc) cum sunt cele analogice prezente în cazul voltmetrului analogic.

Aşa cum s-a afirmat, în interiorul sistemului de calcul se realizează o anumită prelucrare a informației. Este important de văzut în ce fel anume se realizează această prelucrare. Dacă ne gândim la exemplul de mai sus este limpede că voltmetrul analogic îndeplineşte o anumită funcție prin modul în care este construit adică prin conexiunile electrice existente între elementele componente. Un astefel de sistem se numeşte cu *logică cablată* adică modul de prelucrare a informației este determinat de modul

perticular de conectare a elementelor componente. Acelaşi lucru se poate realiza şi în cazul unui sistem numeric şi deci şi astfel de sisteme pot fi în logică cablată. Un sistem realizat în logică cablată nu va putea îndeplini decât o anumită funcţie (în exemplul nostru, nu ne vom putea gândi să folosim voltmetrul la altceva decât la măsurarea tensiunii) cea pentru care s-a realizat un anumit mod de conectare între elementele componente. Rezultă că un sistem în logică cablată este lipsit de flexibilitate, modificarea funcţiei îndeplinite presupunând (cel puţin) modificarea conexiunilor existente şi (eventual) necesitatea adăugării unor componente noi. Un astfel de sistem are totuşi avantajul simplităţii în sensul că el nu va conţine decât elementele componente strict necesare apicaţiei particulare pentru care este destinat.

Sistemele numerice prin faptul că prelucrează informaţia sub formă numerică permit realizarea unor structuri generale, independente de aplicaţia pentru care vor fi folosite. În această situaţie prelucrarea internă a informaţiei nu mai este legată de coneziunile existente între elementele structurii fizice. Pe această structură fizică generică (independentă de aplicaţie) prelucrarea informaţiei se va face pe baza unui program (listă de comenzi) ce se va afla în *memoria* sistemului de calcul. Astfel de sisteme se numesc *cu logică* programată. Ele sunt flexibile (schimbarea programului înseamnă schimbarea sau modificarea aplicaţiei pentru care sunt destinate) dar şi redundante în sensul că structura fizică generală poate avea mai multe elemente decât sunt necesare pentru o anumită aplicaţie.

Calculatoarele analogice nu pot fi construite decât în logică cablată pe când cele numerice pot fi construite atât în logică cablată cât şi în logică programată. Trebuie precizat aici faptul că, deşi în cazul calculatoarelor numerice în logică programată se vorbeşte de o structură fizică generală, aceasta nu poate fi total independentă de aplicaţia (sau familia de aplicaţii) pentru care este folosit calculatorul respectiv.

Deşi varianta de sistem de calcul în logică programată prezintă o serie de avantaje ce au dus la dezvoltarea rapidă şi extinderea aplicării acesteia în cele mai variate domenii, calculatoarele analogice rămân importante şi cu aplicaţii extinse.

Revenind la structura unui sistem de calcul în logică programată, existenţa programului indică faptul că trebuie să existe un element care să poată înţelege şi executa comenzile existente în acest program. Acest element se numeşte *unitate centrală* şi pe lângă funcţiile amintite va mai avea şi alte sarcini în sistemul de calcul.

Din cele prezentate s-a conturat o structură generală a unui sistem de calcul numeric în logică programată şi care trebuie să conţină: unitatea centrală, memoria şi interfeţele.

Intern, un calculator prelucrează mai multe fluxuri de informaţie dintre care principalele fluxuri sunt reprezentate de datele numerice şi de instrucţiunile programului. După modul de prelucrare al acestor fluxuri informaţionale calculatoarele se pot clasifica în:

- maşini de tip SISD (Single Instruction Single Data) care prelucrează la un moment dat o singură instrucţiune program şi o singură valoare numerică;
- maşini de tip SIMD (Single Instruction Multiple Data) care prelucrează la un moment dat o singură instrucţiune program dar mai multe fluzuri de date numerice;

- maşini de tip MIMD (Multiple Instruction Multiple Data) care prelucrează la un moment dat mai multe instrucțiuni program şi mai multe date numerice.

Maşinile de tip SIMD sau MIMD fac parte din categoria calculatoarelor paralele care pot prelucra în paralel mai multe fluxuri de informație.

Din punct de vedere al puterii de calcul, în prezent calculatoarele se clasifică astfel:

- microsisteme, sisteme simple bazate pe un microcontroler (calculator integrat), un procesor de semnal – DSP – (microcontroler destinat prelucrării digitale a semnalelor) sau un microprocesor (unitate centrală integrată pe un singur chip) destinate automatizărilor, dispozitivelor periferice, bunurilor de lar consum, comunicațiilor etc;
- calculatoare personale (PC – Personal Computer), pentru un singur utilizator, construite pe baza unui microprocesor. Aceste maşini sunt de cele mai multe ori de tipul SISD sau SIMD şi sunt prevăzute cu dispozitive de intrare (de exemplu: tastatură, mouse, joystick, scanner, etc.) pentru introducerea datelor de intrare, dispozitive de ieşire (de exemplu: display, imprimantă, plotter, etc.) pentru prezentarea datelor de ieşire şi dispozitive de stocare a datelor de intrare şi de ieşire (discuri, bandă magnetică, disc optic, etc.);
- stații de lucru (WS - WorkStation), pentru un singur utilizator, similare calculatoarelor personale dar dotate cu microprocesoare mai puternice şi cu monitoare de calitate fiind destinate prelucrărilor complexe;
- minicalculatoarele, destinate mai multor utilizatori suportând de la 10 la câteva sute de utilizatori simultan. Sunt maşini de tip MIMD;
- calculatoare de tip mainframe, calculatoare mai puternice decât minicalculatoarele, destinate utilizării simultane de către mai mulți utilizatori (multi user) putând suporta de la câteva sute la câteva mii de utilizatori;
- supercalculatoarele, calculatoare extrem de rapide care pot executa sute de milioane de instrucțiuni pe secundă.

Având în vedere importanța unei astfel de sistem de calcul cât şi perspectivele de dezvoltare şi aplicare, în continuare vom discuta principalele probleme legate de arhitectura acestora.

Sistemele de calcul în logică programată, larg răspândite în prezent, folosesc două tipuri de arhitecturi: arhitectura *von Neumann* şi arhitectura *Harvard*.

Aceste arhitecturi se numesc *arhitecturi secvențiale*, care au o singură unitate centrală şi care execută în mod secvențial programul aflat în memorie spre deosebire de *arhitecturile paralele* care sunt structuri realizate cu două sau mai multe unități centrale.

3.2. Arhitectura von Neumann

Cea mai simplă structură (structura minimală) a unui sistem de calcul este prezentată în figura 3.1.

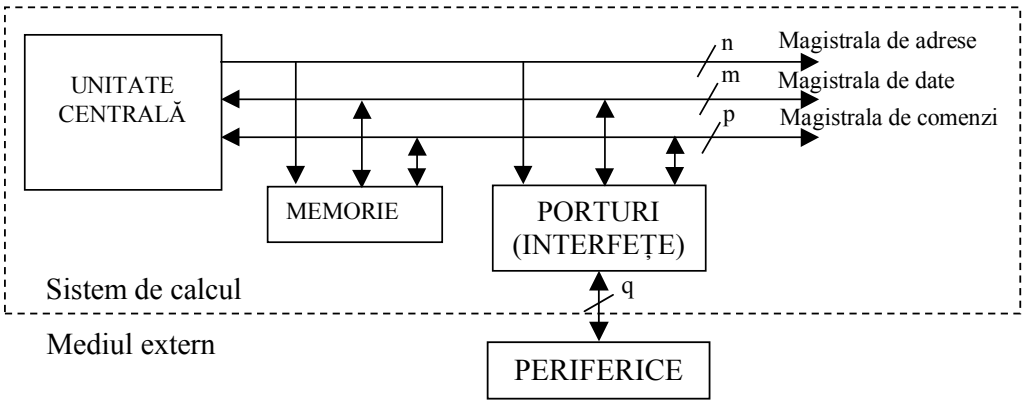


Fig. 3.1. Structura generală a unui sistem de calcul

Structura prezentată este structura minimală a unui sistem de calcul numeric numită şi arhitectură von Neumann. Evident, structura unui sistem de calcul poate conține şi alte elemente care vor duce la ridicarea performanțelor sistemului de calcul dar elementele prezentate în structura de mai sus sunt *indispensabile* funcționării unui sistem numeric în logică programată.

Trebuie observat faptul că această structură nu este legată de o anume aplicație ci ea este determinată doar de modul de funcționare al unui sistem numeric în logică programată (maşină von Neumann în acest caz) si va acoperi o arie largă de aplicații.

Elementele componente ale acestei structuri vor fi prezentate în continuare.

Unitatea centrală care are rolul de comandă şi control a sistemului de calcul şi de execuție a programelor ce se găsesc în memorie.

Un calculator poate avea o singură unitate centrală (in cele mai multe din cazuri) sau mai multe unități centrale ce lucrează în paralel. Sistemele de calcul cu o singură unitate centrală se numesc ***monoprocesor*** iar activitatea de prelucrare se numeşte ***monoprocesare*** iar cele care au mai multe unități de calcul ce lucrează în paralel se numesc ***multiprocesor*** iar activitatea de prelucrare a informației se numeste ***multiprocesare***. Din cauză că în continuare ne vom referi numai la calculatoarele monoprocesor nu se va mai specifica acest lucru explicit.

Memoria care are rolul de a stoca (păstra) programe şi date.

Porturile sau interfețele care au rolul de a realiza schimbul de informație dintre sistemul de calcul şi mediul extern prin trensformarea şi adaptarea semnalelor conform cerințelor.

Toate aceste elemente, care sunt realizate cu ajurorul unor circuite integrate pe scară largă şi foarte largă, sunt legate între ele prin intermediul unor conexiuni electrice. Prin aceste conexiuni circulă semnale electrice care au o anumită semnificație din punct de vedere al informației codificate (binar). Din acest motiv, conexiunile electrice sunt grupate după semnificația informației vehiculate şi un astfel de grup poartă numele de

magistrală (bus). Avem trei tipuri principale de magistrale: magistrala de adrese, magistrala de date şi magistrala de comenzi. Fiecare din aceste magistrale este alcătuită, aşa cum s-a arătat, din mai multe conexiuni electrice. În figura noastră, magistrala de adrese are ***n*** linii, magistrala de date are ***m*** linii iar magistrala de comenzi are ***p*** linii.

Perifericele nu aparţin structurii sistemului de calcul dar sunt menţionate aici deoarece sunt indispensabile funcţionării unui calculator. Vorbind la modul general, un periferic realizează conversia unei anumite forme de energie în energie electrică. Spre exemplu, tastatura, care este un periferic transformă energia mecanică cu care apăsăm tastele într-un semnal electric care este preluat de către interfaţă şi adaptat formatului intern propriu sistemului de calcul, iar monitorul (display-ul) transformă energia electrică în energie luminoasă.

Se observă că interfeţele sunt legate la rândul lor la periferice prin intermediul unor magistrale (de dimensiune ***q*** în figura noastră). Aceste magistrale sunt de diferite tipuri, au diferite dimensiuni şi poartă diferite denumiri legate de tipul interfeţei la care sunt legate.

Elementele care alcătuiesc structura sistemului de calcul se împart în două categorii: *elemente de comandă* (master) şi *elemente comandate* (slave). Într-o structură de calcul vom recunoaşte uşor elementele de comandă prin faptul că acestea pot genera adrese (sensul săgeţii magistralei de adrese este dinspre modulele de comandă spre cele comandate).

3.2. Arhitectura Harvard

Structura simplificată a arhitecturii Harvard este prezentată în figura 3.2.

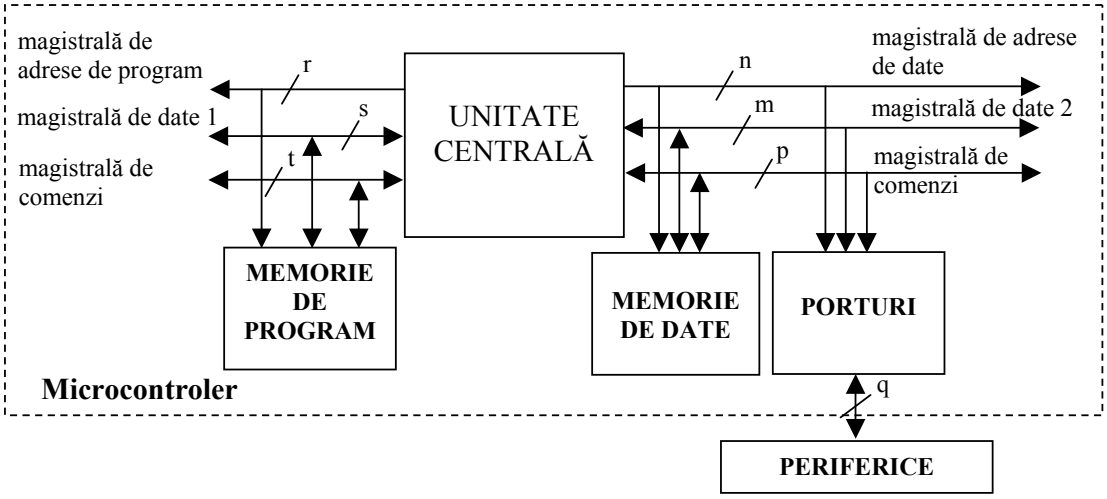


Figura 3.2. Arhitectura Harvard.

Principala caracteristică a acestui tip de arhitectură este reprezentată de utilizarea în structura sistemului de calcul a două memorii cu destinaţii diferite: o memorie de program şi o memorie de date. Acest lucru permite suprapunerea ciclului de extragere a codului operaţie din memorie cu ciclul de citire/scriere în memorie ceea ce duce la

creşterea corespunzătoare a vitezei de lucru. De asemenea astfel de sisteme folosesc cuvinte de lungimi diferite pentru codul instrucțiunii şi pentru date (de exemplu, coduri de instrucțiuni pe 14 biți şi date pe 8 biți) ceea ce permite utilizarea unui set complex de instrucțiuni.

Utilizarea memoriilor cu destinații diferite presupune existența unei magistrale de adrese de program şi a unei magistrale de adrese de date şi, de asemenea, existența unei magistrale de date pentru program – pe care sunt aduse codurile de instrucțiune – şi a unei magistrale de date pentru datele programului (fiecare dintre aceste magistrale putând avea dimensiune diferită).

O astfel de arhitectură este utilizată cu precădere în structura procesoarelor de semnal (DSP – Digital Signal Processor) sau a microcontrolerelor de tip Microchip PIC, Atmel AVR etc.

În astfel de sisteme, programul şi datele acestuia sunt stocate în memorii diferite. De regulă memoria program este o memorie de tip *FLASH* în aşa fel încât programul să nu se şteargă la întreruperea tensiunii de alimentare iar datele sunt stocate în memorii de tip *RAM* static şi memorii de tip *EEPROM*.

Asfel de sisteme reprezintă microcalculatoare integrate care au unitatea centrală, memoria şi porturile realizate pe un singur circuit integrat.

Un calculator poate avea mai multe magistrale de acelaşi tip care se deosebesc prin viteza de variație (frecvența) a semnalelor care parcurg aceste magistrale, dimensiunea şi destinația acestora. Din cauză că diferitele elemente conectate la magistrală au viteze de lucru diferite, elementele mai lente (cu viteză de lucru mai scăzută), vor impune frecvența maximă pe magistrale. Pentru creşterea performanțelor unui calculator, magistralele se realizează pe mai multe nivele cu frecvențe de lucru diferite. Astfel, pe magistrala cu viteza cea mai mare se conectează de obicei unitatea centrală şi memoriile rapide iar pe magistralele cu viteză mai scăzută se conectează memoriile lente şi porturile. Comunicația între magistralele ce lucrează la frecvențe diferite se realizează cu ajutorul unor circuite integrate specializate numite *controlere de magistrală*. Rezultă că un criteriu de performanță a unui calculator este reprezentat de frecvența maximă a magistralelor interne.

Diferitele structuri particulare de calculatoare pot conține şi alte elemente în afară de cele prezentate în figura 3.1, în scopul creşterii performanțelor calculatorului sau datorită destinației acestuia. Dintre aceste elemente suplimentare cele mai uzuale sunt: circuitul de acces direct la memorie (DMA), controlerul de întreruperi, controlerul video şi controlerul de comunicație. Este de remarcat aici faptul că în cazul calculatoarelor de proces apar în plus ceasul de timp real şi ceasul de gardă (watchdog timer) necesare rulării în timp real a aplicațiilor. În figura 3.3 este prezentată schema bloc a unui calculator de proces. În această figură se vede că sistemul este organizat în jurul a mai multor magistrale. Pe magistrala A se află sistemul de calcul propriu-zis iar pe magistrala B se află circuitele de interfață cu procesul. Fiecare dintre magistrale sunt alcătuite, la rândul lor, din mai multe magistrale care nu sunt arătate în figurră.

Pentru a evita confuziile, trebuie menționat faptul că în figura 3.3, printr-un abuz de limbaj, unitate centrală este denumită placa electronică, notată cu 880-P, ce conține atât unitatea centrală propriu-zisă cât şi porturi de intrare/ieşire. Tot aşa, prin extinderea acestei expresii, unitate centrală este denumită cutia sau dulapul în care se află această componentă din structura sistemului de calcul.

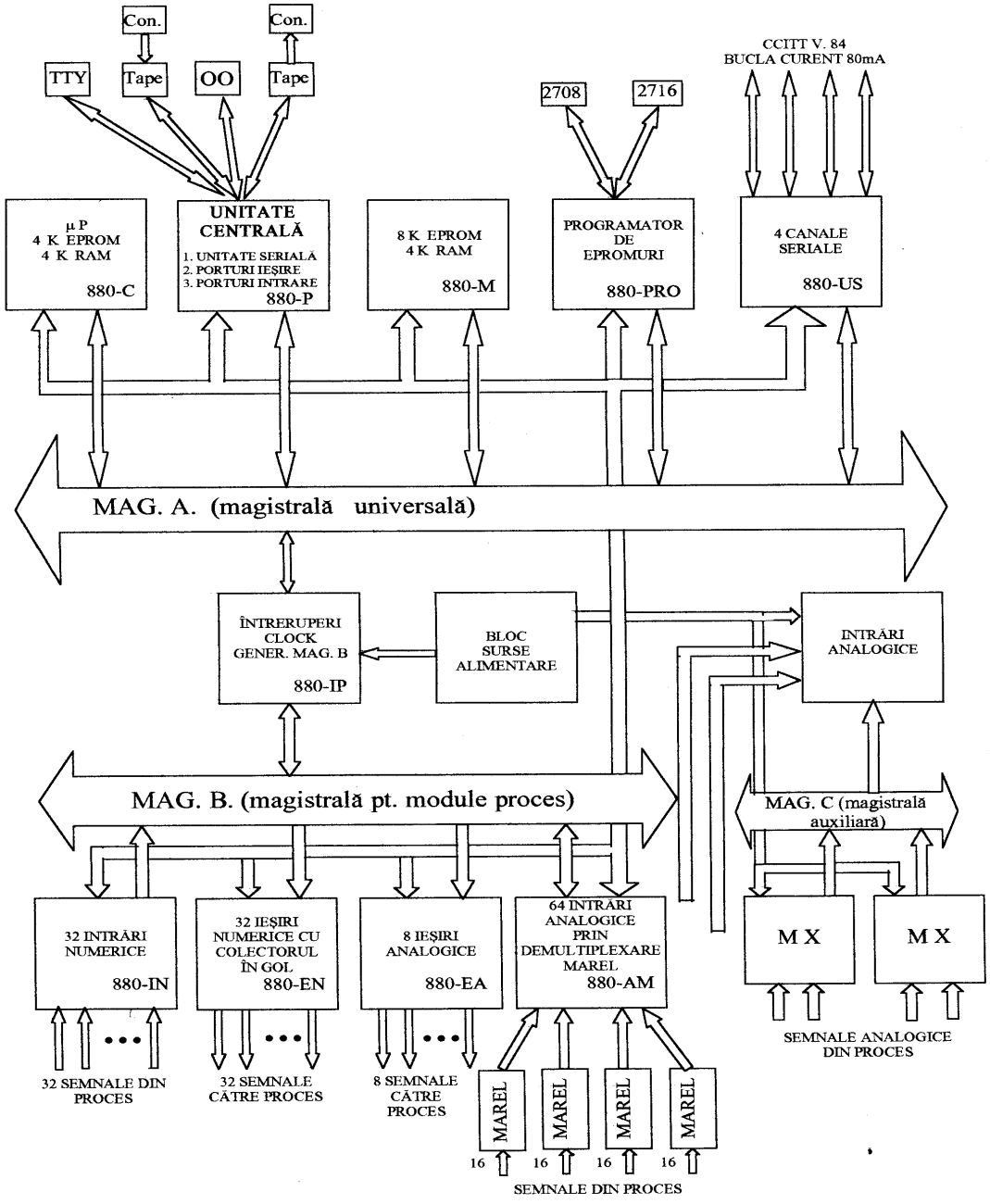


Fig. 3.3. Structura unui calculator de proces

CAPITOLUL 4

UNITATEA CENTRALĂ

4.1. Introducere

Unitatea centrală reprezintă componenta principală a sistemului care coordonează toate activitățile acestuia. De performanțele unității centrale depind în mod esențial performanțele calculatorului. La calculatoarele personale unitatea centrală este reprezentată de un singur circuit integrat numit microprocesor. Stațiile de lucru conțin unul, două sau mai multe microprocesoare, conectate într-o arhitectură scalară, care lucrează în paralel. În cazul minicalculatoarelor, a calculatoarelor mainframe sau a supercalculatoarelor, unitatea centrală este realizată cu ajutorul mai multor componente. De obicei sunt utilizate integrate de tip bit-slice care reprezintă o parte (felie) dintr-o unitate centrală. Prin conectarea în paralel a mai multor astfel de circuite se obțin unități centrale cu diferite performanțe. Aceste unități sunt microprogramabile de către utilizator ceea ce le conferă flexibilitate în exploatare dar complică procesul de proiectare. Microprogramarea este o metodă de realizare a automatelor de comandă prin înscrierea programelor acestora într-o memorie. Prin microprogramare se obțin microprograme, constituite din microinstrucțiuni. Microprogramarea presupune o cunoaştere de detaliu a echipamentului căruia îi sunt destinate microprogramele. Spre deosebire de marea majoritate a altor metode de programare instrucțiunile utilizate în microprogramare conțin atât codul operației cât şi eventualii operatori, cât şi adresa instrucțiunii următoare.

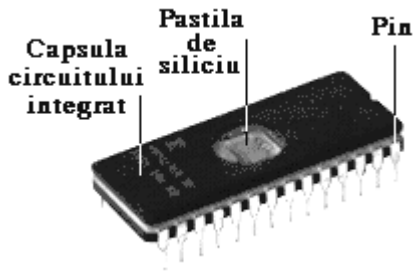


Fig. 4.1. Circuit integrat

Fig. 4.2. Microprocesorul INTEL 486

În figura 4.1 este prezentat modul de realizare a unui circuit integrat.

Principalele mărimi caracteristice ale unei unități centrale sunt reprezentate de: setul de instrucțiuni, numărul de biți prelucrați simultan (lărgimea magistralei de date) şi frecvența de ceas (viteza la care funcționează unitatea centrală).

Din punct de vedere al setului de instrucțiuni, unitățile centrale sunt de două categorii: unități centrale de tip RISC (Reduced Instruction Set Comuter) cu un număr

relativ redus de instrucțiuni şi unități centrale de tip CISC (Complex Instruction Set Computer) cu un număr mare de tipuri de instrucțiuni.

Unitățile de tip RISC sunt ieftine şi foarte rapide din cauză că setul simplu de instrucțiuni le permite execuția cu viteză mare a programului. Unitățile centrale de tip CISC au avantajul posibilităților extinse de lucru datorită setului complex de instrucțiuni iar în ultimul timp, ele sunt un concurent serios, la viteză, al unităților centrale de tip RISC. Cunoscutul microprocesor al firmei INTEL, Pentium, este considerat ca fiind o unitate centrală de tip CISC deşi conține în arhitectura sa multe elemente ale unei maşini RISC. În figura 2.4 este prezentat microprocesorul INTEL 486.

Numărul de biți prelucrați simultan de unitatea centrală reprezintă de fapt lățimea magistralei de date din cauză că pe această magistrală sunt vehiculate date şi instrucțiuni. Lățimea magistralei de date este de regulă un multiplu de opt, în prezent fiind întâlnite în mod uzual unități centrale ce lucrează pe 8, 16, 34, 64, 128 sau 256 de biți.

Frecvența semnalului de tact (semnalul de ceas) aplicat unității centrale reprezintă de asemenea un criteriu de performanță a acesteia. Având în vedere faptul că o instrucțiune este executată de unitatea centrală în una sau mai multe perioade de ceas, rezultă că viteza de calcul creşte odată cu creşterea frecvenței de ceas. În prezent calculatoarele personale performante utilizează frecvențe de tact de până la 600MHz cu perspective de a ajunge la 1GHz.

Aceste criterii de performanță nu sunt absolute din cauză că ele depind în mare măsură de arhitectura unității centrale. Astfel, spre exemplu, două unități centrale cu aceeaşi frecvență de ceas, pot merge cu viteze diferite, datorită structurii interne, cu până la de 20 de ori.

Un alt criteriu de performanță îl reprezintă dimensiunea memoriei interne, memoria cache, a unității centrale. Memoria cache este o memorie foarte rapidă care este folosită ca intermediar între unitatea centrală şi memoria principală a sistemului. Această memorie poate fi atât internă (în structura unității centrale) cât şi externă. Din acest motiv se spune că memoria cache este organizată pe nivele. Cu cât dimensiunea memoriei cache interne este mai mare, cu atât performanțele unității centrale vor fi mai bune.

Alte criterii de performanță ce pot fi luate în considerare în cazul unităților centrale sunt: numărul maxim adresabil de porturi de intrare/iesire, numărul şi dimensiunea registrelor interne, şi modalitățile de adresare.

O unitate centrală execută o instrucțiune în unul sau mai mulți ciclii maşină. Un ciclu maşină reprezintă o activitate elementară a unității centrale (extragerea din memorie a codului operației, citire/scriere din/în memorie, citire/scriere din/în port, achitare cerere întrerupere etc. Un ciclu maşină poate dura una sau mai multe perioade a ceasului de comandă al unității centrale.

Pentru determinarea performanțelor unităților centrale se foloseşte o metodă bazată pe detrminarea numărului de operații în virgulă mobilă (floating-point operatins) executate de aceasta într-o secundă. Unitatea de măsură este FLOPS (FLoating-point Operations per Second), în mod obişnuit performanțele unităților centrale fiind măsurate în megaFLOPS sau gigaFLOPS.

La punerea sub tensiune unitatea centrală porneşte dintr-o stare inițială cunoscută începând să execute programul aflat în memorie de la o adresă bine precizată. Pe timpul execuției unui program unitatea centrală poate ajunge dintr-un motiv sau altul într-o stare nedorită care să afecteze modul corect de funcționare al calculatorului. În aceste situații unitatea centrală trebuie readusă în starea inițială. Acest lucru se face fie în mod automat de către circuitele specializate ale calculatorului fie de către utilizator prin apăsarea unui buton (RESET). Este evident faptul că inițializarea unității centrale poate fi făcută şi prin întreruperea tensiunii de alimentare dar acest lucru este nerecomandabil de cele mai multe ori pentru că se pot pierde informații în curs de prelucrare de către periferice şi/sau informații legate de prelucrările curente. De asemenea şocurile termice şi de tensiune ce apar la pornirea calculatorului nu recomandă folosirea acestui procedeu pentru reinițializarea unității centrale. Butonul de inițializare a unității centrale (RESET) este marcat uneori şi cu simbolul: ▸●◂.

Datorită dezvoltării spectaculoase a microsistemelor şi datorită faptului că unitățile centrale a acestora, microprocesoarele, beneficiază de o întreagă gamă de inovații tehnologice, ne vom concentra atentia asupra unitaților centrale de tip microprocesor

Principalele avantaje oferite de microsisteme față de sistemele convenționale sunt:

- costul scăzut al hardware-ului;
- siguranța în funcționare îmbunătățită;
- interferențe electromagnetice scăzute - nivelul larg de integrare evită influiențele electromagnetice chiar şi la nivele mari de variație ale tensiunii şi curentului prin circuitele electronice de putere. În general pentru protecție este suficient ecranul cu care prevăzută componenta. Zgomotul de cuplare cu sursele de alimentare şi semnalele de intrare poate fi minimizat printr-o bună filtrare analog-digitală;
- absența împrăştierii sau a variației parametrilor;
- compatibilitate cu controlul digital ierarhic;
- hardware şi software universal;
- posibilitatea de diagnostic şi autodiagnostic.

Principalele dezavantaje prezentate de către microsisteme sunt:

- viteza de răspuns (calcul) este în unele cazuri insuficientă pentru elaborarea unui răspuns în timpul aşteptat;
- eroarea de cuantizare - eroarea de cuantizare poate fi micşorată prin creşterea numărului de biți ai cuvintelor microsistemului şi ai convertorului A/D;
- lipsa de acces la semnale software - nu se pot efectua măsurători cu osciloscopul asupra semnalelor software şi deci trebuie concepute măsuri adecvate de depanare.
- dezvoltarea software-ului poate fi uneori scumpă.

În cazul microsistemului cu microprocesor unitatea centrală şi celelalte elemente ale microsistemului sunt componente distincte fapt ce participă la flexibilitatea soluției. În

cazul microcontrolerelor într-o singură componentă sunt integrate pe lângă unitatea centrală şi o parte din elementele microsistemului. Deşi microcontrolerele prezintă o flexibilitate mai redusă ele prezintă avantajul miniaturizării şi a preţului de cost scăzut.

4.2. Microprocesorul universal (structura generală a unui microprocesor)

Structura unui microprocesor cuprinde: o unitate aritmetică şi logică, având drept scop efectuarea de operaţii aritmetice şi logice elementare; un set de registre pentru memorarea temporară şi manipularea cu viteză ridicată a unui numar relativ mic de rezultate intermediare; un bloc de comandă şi secvenţiere care asigură desfăşurarea ordonată a tuturor operaţiilor în interiorul microprocesorului, precum şi comunicaţia acestuia cu lumea exterioară; un bloc de decodificare a instrucţiunii curente, care interpretează instrucţiunea în curs şi determină acţiunile ce se impun; un bloc de tratare a cererilor de întrerupere uilizat pentru luarea în considerare a evenimentelor asincrone faţă de desfăşurarea programului; tampoane între microprocesor şi magistralele sistemului. Microprocesoarele din ultimile generaţii au inclus şi un coprocesor matematic ce le permite realizarea unor calcule matematice complexe cu viteză ridicată şi memoria cache.

Structura de principiu a microprocesorului universal este prezentată în figura 4.3.

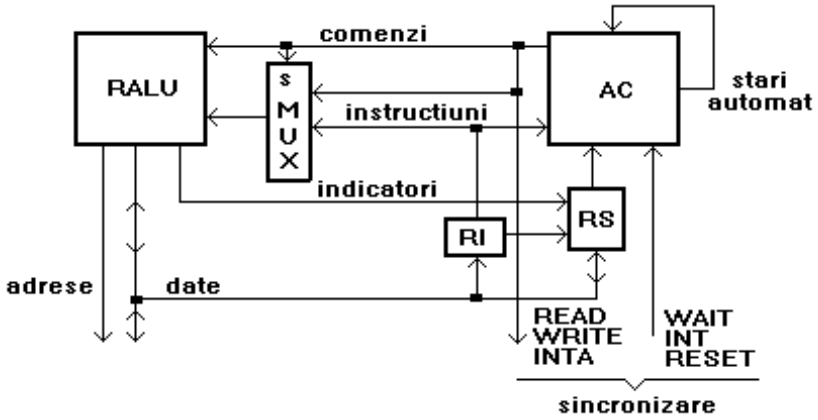


Fig.4.3. Schema bloc a microprocesorului universal

Această structură este formată din următoarele elemente componente:

- RALU - unitatea logică şi aritmetică şi regiştrii de uz general şi special;
- AC - automatul de control;
- RI - registrul de instrucţiune;
- RS - registrul de stare;
- MUX - un grup de multiplexoare ce asigură o parte din comenzile la bornele RALU de la AC sau RI, sub controlul (selecţia) semnalelor din AC.
- Semnalele interne care asigură funcţionarea sistemului sunt următoarele:
  - comenzi, generate de AC către subansambluri sau către exterior;

- instrucțiuni, generate din RI către AC şi prin intermediul MUX, către RALU;
- indicatori, generați de ALU din RALU către RS;
  - date - reprezintă un bus intern bidirecțional pe care sunt cuplate RALU, RS şi RI (unidirecțional);
  - adrese - cale de adrese ce se emite în exterior din RALU pentru controlul dispozitivelor de memorare exteroare microprocesorului.
  - Semnalele la bornele microprocesorului sunt următoarele:
  - adrese - cale unidirecțională;
  - date - cale bidirecțională;
  - semnale de sincronizare:
    - READ - semnal ce comandă citirea de la adresă configurației binare de pe calea date;
    - WRITE - semnal ce comandă la adresă a configurației binare de pe calea date;
    - WAIT - semnal recepționat ce impune trecerea microprocesorului într-o stare de aşteptare drept urmare a unei comenzi READ sau WRITE ce nu a fost încă executată;
    - INT - semnal de întrerupere a cărui recepționare stopează procesul curent de calcul pentru a da curs unui eveniment exterior sistemului;
    - INTA - semnal ce indică luarea în considerație a semnalului de întrerupere INT; ca urmare a acestui semnal, dispozitivul ce a activat INT va genera pe date o configurație binară specifică;
    - RESET - semnal de inițializare a funcționării microprocesorului.

Microprocesoarele sunt procesoare la care funcția de control intern este minimizată, în sensul că automatul de control AC este gândit foarte simplu. Procesoarele microprogramate se situează la extrema în care funcția de control intern are o pondere foarte importantă. Dacă un microprocesor foloseşte cât mai direct codul instrucțiunii (biții din RI comandă direct funcționarea RALU), în cazul imensei majorități a instrucțiunilor, la o structură microprogramată, codul instrucțiunii acționează cu preponderență asupra automatului de control, declanşând secvențe complexe de comandă.

4.3. Caracteristicile principalelor tipuri de microprocesoare

În continuare sunt prezentate pe scurt principalele caracteristici ale unor tipuri reprezentative de microprocesoare.

4.3.1. Microprocesorul ZILOG Z80

Deşi este un microprocesor care a apărut de o perioadă de timp destul de lungă el este unul dintre microprocesoarele pe 8 biți dintre cele mai populare. Pentru acest tip de microprocesor au fost dezvoltate o serie foarte mare de aplicații începând de la bunuri de larg consum până la aplicații industriale.

Microprocesorul Z80 împreună cu componentele sale auxiliare formează o familie tipică pentru generaţia de microprocesoare de 8 biţi. Componentele din familie asigură posibilitatea realizării tuturor funcţiilor clasice ale unui sistem cu microprocesor.

Principalele caracteristici ale acestui microprocesor sunt:

- magistrala de date pe 8 biţi;
- magistrala de adrese pe 16 biţi;
- 12 regiştrii de uz general pe 8 biţi cu utilizare pe 16 biţi, regiştrii cu selectare alternantă câte şase;
- 2 regiştrii acumulator şi 2 regiştrii indicatori de condiţii cu selectare alternantă;
- 2 regiştrii speciali, unul pentru determinarea vectorului de întrerupere şi unul pentru reîmprospătarea transparentă a memoriei dinamice;
- 2 regiştrii index pe 16 biţi;
- o întrerupere nemascabilă şi una mascabilă cu un mecanism foarte flexibil de răspuns la cererea de întrerupere;
- este capabil să execute operaţii aritmetice (mai puţin înmulţirea şi împărţirea) şi logice;
- frecvenţă de tact maximă 6MHz.

4.3.2. Microprocesoarele INTEL 80x86

Familia microprocesoarelor INTEL s-a îmbogăţit continuu cu noi membrii care au dus la creşterea performanţelor hardware şi software a acestor componente. Principalul avantaj al acestei familii de microprocesoare este păstrarea compatibilităţii software între diferitele generaţii începând cu microprocesorul 8086 ceea ce a permis dezvoltarea continuă a aplicaţiilor pentru aceste microprocesoare.

Începând cu microprocesorul 80286 s-au introdus două moduri de funcţionare diferite: ***modul real*** (Real Address Mode) în care microprocesorul poate adresa ca şi 80806 un spaţiu de memorie de 1Mo şi ***modul protejat*** (Protected Virtual Address Mode) în care spaţiul adreselor fizice a crescut la 16Mo iar cel al adreselor virtuale la 1Go. Microprocesorul 80286 a fost proiectat pentru a permite (în mod protejat) ***sisteme multitasking***, oferind câte un segment de stare ataşat fiecărui ***task***, care sunt structuri manevrate hardware şi conţin stările curente (incluzând toţi regiştrii) ale taskurilor. Selectoarele acestor segmente de stare identifică unic taskul ataşat. De asemenea, este facilitată şi comutarea taskurilor, care poate fi invocată printr-o singură instrucţiune. Fiecare task din sistem poate avea propriul lui spaţiu de adrese logice. Microprocesorul 80286 oferă şi un mecanism evoluat pentru comunicarea între taskuri, sincronizarea lor şi partajarea memoriei.

Istoria microprocesoarelor din familia 80x86 este strâns legată de cea a coprocesoarelor matematice. Fiecărui element de bază din familie i-a fost asociat un coprocesor pentru a îmbunătăţi performanţele sistemului de calcul. Pentru microprocesoarele 8086, 80286 şi 80386 sunt folosite coprocesoarele 8087, 80287 şi respectiv 80387.

Microprocesorul 80386 pe 32 de biţi a fost proiectat pentru utilizarea în aplicaţii intensive complexe. Datorită celor 32 de biţi de adrese, spaţiul adreselor fizice este de 4Go iar spaţiul adreselor logice de 64To. Este prevăzut cu microcod care suportă direct aplicaţii care utilizează întregi mari, structuri complexe de date şi un număr mare de programe. Microprocesorul are patru moduri distincte de funcţionare: real, virtual 8086, protejat 286 şi nativ 386. În figura 4.4 este prezentată schema bloc a microprocesorului 80386.

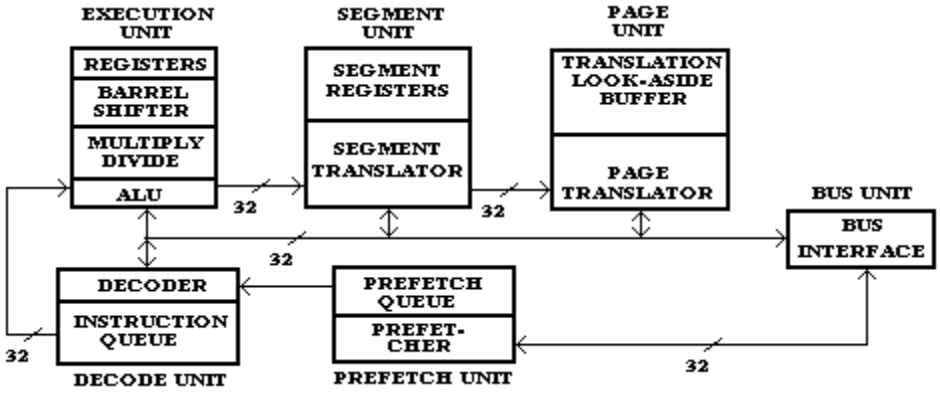


Fig. 4.4. Schema bloc a microprocesorului 80386

Microprocesorul 80486 oferă multe facilităţi noi: o memorie cache de 8ko, o unitate de gestiune a memoriei compatibilă 80386, un procesor 80386 şi un subset 80387 pe un singur chip, ceea ce permite ca softul existent (fiind vorba de înalta compatibilitate) să ruleze mai repede pe 80486 decât pe perechea 80386-80387.

Microprocesoarele de tip INTEL, începând cu microprocesorul de tip 80386 au patru moduri de funcţionare de bază: modul real, modul virtual şi modul protejat. Având în vedere faptul că sistemul de operare DOS este destinat în exclusivitate microprocesoarelor de tip INTEL şi datorită faptului că aceste mecanisme se întâlnesc şi la alte tipuri de microprocesoare, ele vor fi descrise pe scurt în continuare.

MODUL REAL. Acest mod este cel în care se intră după iniţializarea microprocesorului. De regulă, sub sistemul de operare DOS microprocesorul se află în modul real. Specific acestui mod de funcţionare este faptul că microprocesorul nu poate rula decât un singur program odată. Există posibilitatea rulării programelor sub sistemul de operare DOS şi în alte moduri ale microprocesorului, dar acest lucru necesită extensii ale sistemului de operare (existând în acest sens programe specifice).

MODUL VIRTUAL. Acest mod permite execuţia programelor în contextul mecanismelor de protecţie, gestiune a taskurilor şi management al memoriei. Un program poate rula în mod virtual în paralel cu alte programe care rulează în mod protejat. De asemenea, pot fi executate deodată mai multe programe în mod virtual. Sistemul de operare Windows 9x se bazează pe acest mod performant de funcţionare al microprocesorului, permiţând execuţia în paralel a mai multor programe (sesiuni DOS) cât şi a mai multor programe Windows.

MODUL PROTEJAT. În acest mod, microprocesorul emulează funcționarea unui microprocesor de tip 80286 în mod protejat. Modul protejat prezintă un mecanism sofisticat pentru protejarea datelor, integritatea sistemului, concurența taskurilor şi gestiunea memoriei incluzând şi cea a memoriei virtuale. În modul protejat, în cazul microprocesorului 80286, spațiul adreselor fizice este mărit de la 1 Moctet la 16 Mocteți, în timp ce spațiul adreselor virtuale a fost mărit la 1 Goctet. În acest mod, programele folosesc adrese virtuale, translatarea lor în adrese fizice făcându-se automat pe baza unor tabele cu descriptori de segmente. Acest mecanism permite implementarea eficientă a sistemelor cu memorie virtuală în care utilizatorul vede memoria internă şi cea externă ca o singură memorie. Modul protejat permite implementarea sistemelor multitasking, oferind câte un segment de stare ataşat fiecărui task, care sunt structuri manevrate hardware şi conțin stările curente (incluzând toți regiştrii) ale taskurilor. Selectoarele acestor segmente de stare identifică unic taskul ataşat. De asemenea, este facilitată şi comutarea taskurilor, care poate fi invocată printr-o singură instrucțiune. Fiecare task din sistem poate avea propriul lui spațiu de adrese logice, existând de asemenea un mecanism evoluat pentru comunicarea între taskuri, sincronizarea lor, partajarea memoriei etc.

MODUL NATIV. Acest mod foloseşte întreaga putere a microprocesorului. Memoria virtuală permite ca dimensiunea unui program să fie limitată de spațiul pe disc şi nu de dimensiunea memoriei interne. Mecanismele de protecție sunt destul de puternice pentru a evita accidentele între taskuri sau între utilizatori. Spațiul de adrese al taskurilor este complet separat, segmentele sunt tipizate şi au diferite drepturi de acces. De asemenea, se verifică şi depăşirea limitelor segmentelor. Există trei nivele de prioritate, sistemul de operare având nivelul zero (cel mai prioritar), iar aplicațiile programatorilor având nivelul trei (cel mai puțin prioritar).

Microprocesorul INTEL Pentium

Microprocesorul Pentium este integral compatibil cu procesoarele INTEL anterioare, dar se deosebeşte de acestea în multe privințe. Cel puțin una din aceste deosebiri este majoră: microprocesorul Pentium are două canale identice de procesare a datelor, ceea ce îi permite să execute două instrucțiuni în acelați timp. Această capacitate de a executa simultan două instrucțiuni este numită tehnologie superscalară (procesare paralelă). Această tehnologie asigură performanțe suplimentare față de cea a procesorului 486.

Prin modul numit BTB (Branch Target Buffer) care utilizează o tehnică numită branch prediction (predicția salturilor) în scopul reducerii timpului de aşteptare în canalele de procesare, cauzat de aducerea instrucțiunilor unei ramuri aflate la o altă locație de memorie, se realizează la microprocesorul Pentium o utilizare mai eficientă a memoriei cache interne. Modul BTB încearcă să prevadă când va apare o instrucțiune de salt şi să aducă în memorie instrucțiunile

corespunzătoare ramurii la care se va face saltul. Utilizarea tehnicii de prevedere a ramificării unui program permite microprocesorului să mențină în funcționare la viteză maximă, cele două canale pentru execuția instrucțiunilor.

Principalele caracteristici ale microprocesorului Pentium sunt:

- gama frecvențelor maxime: 450MHz;
- dimensiunea regiştrilor: 32 biți;
- magistrala de date externă: 64 biți;
- magistrala memoriei 32 biți;
- memoria maximă: 4Gb;
- memorie cache încorporată: 8kb pentru instrucțiuni şi 8kb pentru date la primele tipuri ajungând la 2Mb în prezent;
- tipul memoriei cache încorporate: cu două blocuri asociate, Write-Back Data;
- coprocesor matematic cu unitate de calcul în virgulă mobilă (FPU) inclusă;
- gestionarea memoriei: SMM (System Management Mode), extinsă la a doua generație.

Dezvoltarea microprocesoarelor a depaşit de mult acest prag. În prezent există microprocesoare cu mai multe unități centrale integrate care au mecanisme sofisticate.

Datorită faptului că înțelegerea funcționării unui microprocesor este pe deplin posibilă prin analiza funcționării microprocesorului INTEL 8086, datorită structurii simple dar revoluționare a acestuia, compatibilitatea cu generațiile INTEL dezvoltate ulterior şi datorită existenței unei consistente documentații în legătură cu acest domeniu, vom prezenta în continuare structura în detaliu a acestui microprocesor.

4.3.2.1. Microprocesorul INTEL 8086/8088

Microprocesoarele 8086 şi 8088 de generația a III-a sunt microprocesoare cu o structură pe 16 biți, 8088 este proiectat cu o magistrală de date externă de 8 biți în timp ce 8086 poate transfera 16 biți deodată. Datorită faptului că în afară de această deosebire funcționarea celor două tipuri de microprocesoare este identică, în continuare ne vom referi numai la microprocesorul 8086. Performanțele acestor microprocesoare sunt datorate structurii interne de 16 biți şi arhitecturii ***pipeline*** care permite instrucțiunilor să fie preîncărcate în timpul ciclurilor disponibile ale magistralei.

Principalele caracteristici ale acestor microprocesoare sunt:

- magistrala de date de 16 biți;
- magistrala de adrese de 20 biți;
- adresare segmentată a memoriei;
- 4 regiştrii de uz general pe 16 biți adresabile direct sau pe octeți;
- 4 regiştrii de segment pe 16 biți;
- 2 regiştrii index pe 16 biți;
- un sistem foarte flexibil de adresare a memoriei (7 moduri);

- sistem de întreruperi hardware şi software;
- posibilitatea funcționării în sisteme multiprocesor;
- posibilitatea efectuării operațiilor aritmetice (inclusiv înmulțire şi împărțire) şi operații logice;
- frecvență de tact 6MHz.
-

Microprocesorul 8086 este un microprocesor pe 16 biți şi memorie adresabilă direct de 1Mo. Structura unității centrale, cu elementele accesibile programatorului, este prezentată în figura 4.5.

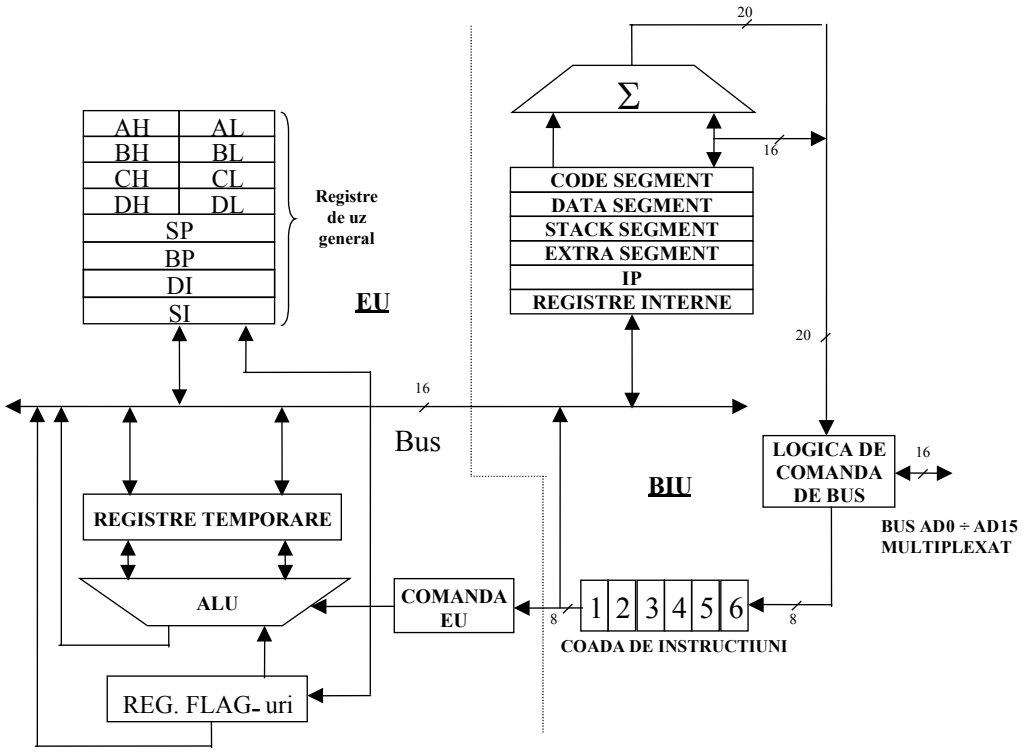


Figura 4.5. Structura unității centrale 8086.

Asa cum se arată în figura 4.5, structura microprocesorului I 8086 se compune din două unități : EU = unitate de execuție şi BIU = unitate de interfață cu magistrala (bus).

BIU execută toate ciclurile de bus (“READ”, “WRITE”, INTA”) fie la cererea EU sau pentru umplerea cu coduri a unei cozi Q de instrucțiuni. Coada de instrucțiuni este o memorie de tip FIFO cu 6 cuvinte.

BIU execută noi cicluri “FETCH” în intervalele cât EU nu solicită bus-ul.

EU obține coduri de la BIU (aşteaptă dacă Q este vidă), execută instrucțiunile, lucrând cu adrese şi date cu 8/16 biți, actualizează flag-urile şi furnizează adrese şi date către BIU.

EU calculează adresele efective ale operanzilor, conform modului de scheme utilizat. Adresa efectivă pe 16 biți, BIU generează adresa fizică cu 20 de biți, selectarea adresei efective, generate de EU.

Dacă Q este plină şi EU nu solicită transferuri pe bus apar cicluri în rotire (aşteptare).

Exemplu: T1 T2 T3 T4 TI TI ... TI TI T1 T2 T3 T4 ... TI = Idle State.

Modurile de lucru posibile pentru unitatea cuntrală I8086 sunt :

- Modul "minim" MN / MX = 1

Semnale de comandă: ALE, DT/K, DEN, M/IO, WR, INTA, HOLD, HOLDA.

Folosit în sisteme mici, de regulă monoprocesor. În figura alăturată circuitele "buffer"de tip 8286 pot lipsi dacă încărcarea electrică a bus-ului este redusă.

- Modul "maxim" MN / MX =0

UCP generează către controlerul de sistem I 8288 semnale de stare identificare ale ciclului de bus, în rest generează semnale corespunzătoare pe bus-ul de comandă.

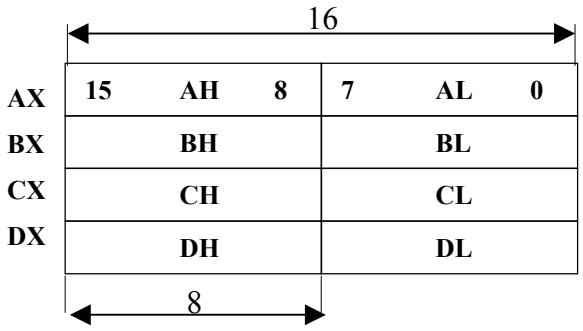
Conectarea UCP – I 8086 în modul "maxim".

În modul "maxim" liniile RQ/GT0, RQ/GT1, LOCK, QS0, QS1 se pot utiliza pentru conexiuni de tip multiprocesor. Se pot, de exemplu, conecta alte procesoare pe bus-ul local al UCP, utilizând pentru arbitrajul de bus liniile RQ/GT (care înlocuiesc pe HOLD/HOLDA din modul "minim").

Registrele unității centrale

Registrele unității centrale aflate în secțiunea **EU** (**E**xecution **U**nit) sunt:

AX, BX, CX, DX
SP, BP, SI, DI
F



Registre cu 16 biți de uz general, adresabile direct sau pe octet:

AX = (AH, AL)
BX = (BH, BL)
CX = (CH, CL)
DX = (DH, DL)

} fiecare din ele poate servi ca destinație a datelor (accumulator)

Utilizările implicite ale registrelor sunt:

AX: utilizat pentru operaţii aritmetice (*), (/) pe 16 biţi şi pentru operaţii de I/E pe 16 biţi; în mod analog AL este utilizat pe 8 biţi şi în plus pentru aritmetică zecimală şi conversii de cod; AH este utilizat la (*) şi (/) pe 8 biţi;
BX: utilizat în conversii de cod şi ca registru de bază de adrese;
CX: utilizat în operaţii cu şiruri, cu rol de contor de cicluri;
CL: utilizat în deplasări (stânga, dreapta – cu un număr de paşi daţi ca parametru de valoare lui CL);
DX: utilizat la (*), (/) pe 16 biţi şi ca registru de adresare indirectă la porţile de intrare – ieşire (I/E);
SP: utilizat implicit în toate operaţiile cu stiva;
SI, DI: utilizate în operaţiile asupra şirurilor de date;

F = registrul de flag-uri şi control al procesorului (se află în EU).
Semnificaţia fanioanelor din registrul **F** este:

CF = C, PF = P, AF = H, ZF = Z, SF = S – semnificaţii obişnuite :
CF = "carry flag": depăşire aritmetică;
PF = “parity flag”: paritate;
AF = “auxiliary flag”: transport între bitul 3 şi 4;
ZF = "zero flag”: valoare zero;
SF = “sign flag”: semnul.
TF = “trip flag”; TF = 1 determină UCP să lucreze în mod pas cu pas(“single step”), în care CA generează o întrerupere internă după fiecare execuţie a unei instrucţiuni;
IF = masca pentru întreruperi externe (IF =1 => validarea întreruperilor; IF =0 => invalidarea întreruperilor);
DF = “direction flag”- indică direcţia deplasării adresei la operaţiile cu şiruri de date (DF =1 => autodecrementare, DF =0 => autoincrementare, după o operaţie elementară);
OF = V (depăşire).

Registrele **SP**, **BP** sunt registre cu 16 biţi utilizate în operaţiile cu stiva.
Registrele **SI**, **DI** sunt registre cu 16 biţi utilizate în operaţiile cu şiruri; **SI** conţine adresa sursei iar **DI** adresa destinaţiei.

Registrele aflate în secţiunea **BIU** (**B**us **I**nterface **U**nit) a unităţii centrale 8086 sunt:

CS, DS, SS, ES = sunt registre segment care conţin adresele de bază ale segmentelor logice de cod, date, stivă şi extrasegment;
IP = Instruction Printer = contor de program, cu 16 biţi. Valoare ce reprezintă adresa relativă (offset-ul) a instrucţiunii curente în segmentul de cod (relativ la CS). În cazul unei instrucţiuni de salt, IP este salvat în vârful

stivei (împreună cu CS, deci saltul este inter-segment) şi apoi încărcat cu adresa relativă în segmentul de cod a instrucţiunii ţintă ;

Posibilităţile de lucru în sisteme multiprocesor

Arhitectura familiei I8086 conlucrarea între două tipuri de procesoare:

- independente (execută propriile secvenţe de instrucţiuni) ;
- coprocesoare – care obţin instrucţiuni din memoria unui procesor gazdă (host), urmărind ciclurile "FETCH"ale gazdei, le recunosc pe cele destinate lor şi le execută. Efectul obţinut este extinderea setului de instrucţiuni al gazdei.

Organizarea ierarhică a bus-urilor

Familia admite două tipuri de bus-uri: locale şi de sistem. Ambele pot fi multimaster (mai multe UCP conectate la bus). Între bus-ul local şi cel de sistem se conectează interfeţe.

Conexsiunile externe ale I8086

În modul "maxim" UCP livrează controlerului de sistem I 8288, în fiecare ciclu de bus, un cuvânt de comandă (S0, S1, S2), pe care acesta îl decodifică şi generează semnale de comandă pe magistrală (bus). În acest mod se pot conecta mai multe UCP la bus.

Semnalele externe ale microprocesorului I8086 sunt prezentate în figura 4.6.

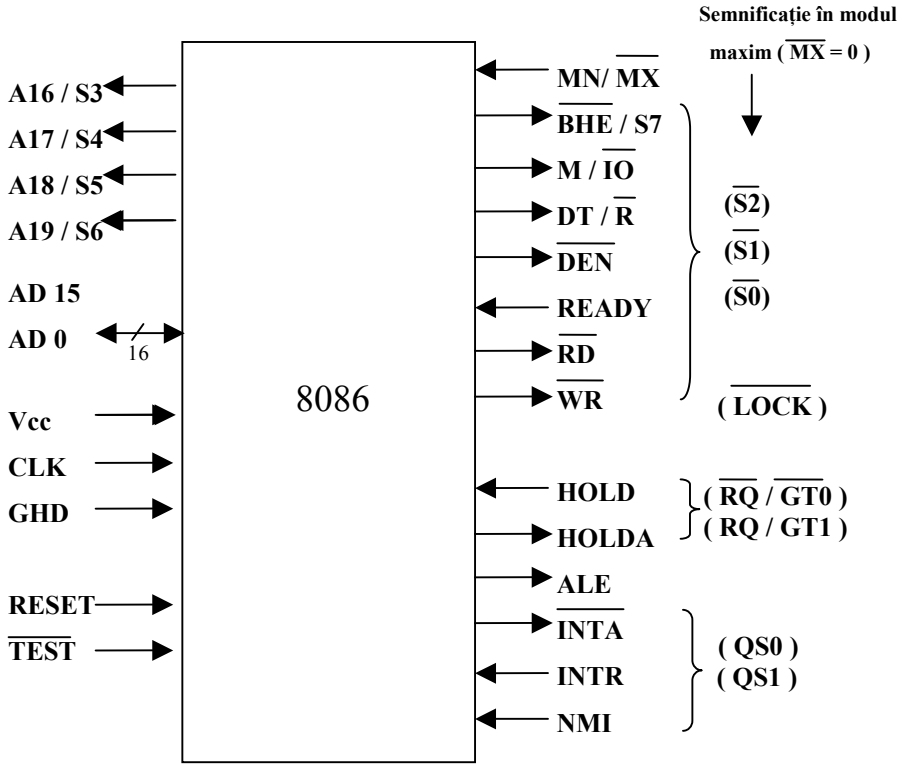


Figura 4.6. Semnalele externe ale microprocesorului I8086.

În tabelul 4.1 sunt date semnificaţiile semnalelor externe ale microprocesorului I8086 atât în modul "minim" cât şi cel "maxim". Microprocesorul I8086 este realizat în capsule de 40 de pini. Din acest motiv semnalele acestuia sunt multiplexate, în aşa fel

încât să poata fi generate toate semnalele pe aceşti pini (adrese – 20 de semnale, date – 16 semnale, comenzi şi alimentare), cele 16 linii de adresă A15 … A0 fiind multiplexate cu cele 16 linii de date D15 … D0.

TABELUL 4.1.

	Simbol	Nr. pin	Semnificație	Tip
SEMNIFICAȚII FIXE	AD15 – AD0	2÷16, 39	Adresa în T1/D este în T2, T3, Tw, T4	I/O, Z
	A19 / S6	35 ÷38	Adresa în T1 / semnele de stare în T2, T3, Tw, T4	
	A18 / S5			
	A17 / S4			
	A16 / S3			
	MN / MX	33	Comanda modului: 1 minim; 0 maxim	I
	BHE / S7	34	Validare bus (HIGH) în T1 / stare în T2 ÷ T4	**O, Z**
	RD	32	Comandă de citire pe bus-ul local	O, Z
	READY	22	Memorie / (I/E) "goto"pentru transfer	I
	RESET	21	Comanda de resetare	I
	TEST	23	Testată de instrucțiuni WAIT (Aşteaptă până ce TEST = 0	I
	INTR	18	Întreruperi mascabile externe; activă pe nivel	I
	NMI	17	Întreruperi nemascabile externe; activă pe front (+)	I
	CLK	19	Semnal de test generat de 8284	I
	VCC	40	Alimentare +5V	I
	GHN	1, 20	Masă	

SEMNIFICAȚII ÎN MOD MINIM	M / IO	28	Ieşire de stare. Selecție memorie (I / E)	O, Z
	WR	290	Comandă de scriere	O, Z
	DT / R	27	Comandă semnul transferului; 1=> transmisiune; 0 => recepție	O, Z
	DEN	26	Validare date. Activ în cicluri M, I/E, INTA	O, Z
	ALE	25	Indică prezența adresei pe AD0 ... AD15	**O**
	INTA	24	"READ"pentru cele două cicluri INTA ale lui 8086	O
	HOLD	31	Cerere de bus de la alt "master"	I
	HLDA	30	Confirmare de cedare a bus-ului	O
SEMNIFICAȚII ÎN MOD MAXIM	S2, S1, S0	26 ÷ 28	Semnale de stare pentru ciclu de bus	O
	RQ / GT1	30	Comenzi cerere/cedare pentru arbitraj de bus	I / O
	RQ / GT0	31		
	LOCK	29	Indicator că 8086 nu va ceda bus-ul altui "master"	O
	QS1 – QS0	24, 25	Starea cozii de instrucțiuni din interfața cu bus-ul a UCP	O

unde: I = intrări; O = ieşiri; Z = impedanță mare;

Observaţii:

- cu T_n se notează ciclurile unității centrale (T_1 – ciclul de extragere cod operație etc.)
- AD15 – AD0 : generează adrese în intervalul T1, devin intrări / ieşiri de date în T2 ÷ T4 şi trec în impedanță mare în timpul ciclurilor INTA, sau dacă UCP a cedat bus-ul (HDLA = 1);
- A19 / S6 – A16 / S3: generează adrese în intervalul T1 (cu valoare "0"dacă ciclul de transfer este I/E) şi semnale de stare în T2 ÷ T4.
  - S6 = 0 indică ocuparea bus-ului de către UCP;
  - S5 = IF copiează starea flag-ului de intrerupere. Astfel, starea de validare / invalidare se poate citi hardware din exteriorul UCP.

S4	S3		
0	0	"Alternate data"	Indică registrul segment curent utilizat pentru adresare.
0	1	"Stack"	
1	0	"Code or home"	
1	1	"Data"	

Aceşti pini trec în starea Z în timpul cât HDLA = 1

- BHE şi A0 determină tipul transferului pe 8 sau 16 biți, astfel:

BHE	A0	tipul transferului
0	0	16 biți
0	1	MSB (adresă impară)
1	0	LSB (adresă pară)
1	1	-

BHE = 0 atunci când un octet trebuie transferat pe cei 8 biți mai semnificativi (MSB) ai bus-ului cu 16 biți. El este activ în timpul ciclurilor "RD", "WR", "INTA" (este "0" în timpul primului ciclu "INTA").

BHE trebuie demultiplexat odată cu A0 ÷ A15 (prin memorare pe frontul negativ al semnalului I/E deoarece în T2 ÷ T4 el indică bitul de stare S7.

În modul "maxim", o parte din semnificațiile pinilor se modifică.

- S2, S1, S0 – identifică tipul de ciclu maşină în intervalele T4, T1, T2, informând controlerul de sistem I 8086 asupra tipului de transfer care urmează (ele sunt inactive în T3 şi Tw).

  S2, S1, S0
  0 – confirmarea acceptării întreruperii,
  1 – citire I/E,
  3 – HALT (oprire),
  4 – Citirea codului instrucțiunii,
  5 – Citire din memorie a unui operand,
  6 – Scriere în memorie a unui operand,

7 – Ciclu inactiv.
Pinii S2, S1, S0 trec în starea Z în intervalele în care UCP cedează bus-ul.

- Liniile bidirecționale RQ / GT, (RQ/GT0 au prioritate față de RQ/GT1) şi sunt utilizate de alte module "master"de pe bus-ul local pentru pentru a cere bus-ul de la UCP. Cererea se face prin RQ = 0 (intrare); la sfârşitul ciclului maşină curent UCP cedează bus-ul şi generează GT = 0 ("GranT"). Semnalul LOCK = 0 indică faptul că UCP nu va ceda bus-ul deoarece execută o secțiune de transfer neîntreruptibilă ce trebuie terminată.
- Semnalele QS0 şi QS1 indică starea cozii de instrucțiuni existente în unitatea de interfață de bus a UCP.

Firma Intel a creat o serie de circuite integrate necesare realizării microsistemelor cu microprocesor I8086. O parte din aceste circuite au devenit modele pentru dezvoltările ulterioare, foarte multe din circuitele moderne fiind compatibile cu aceste circuite. Principalele circuite dezvoltate de firma Intel sunt :

- 8086 (IAPX 86/10), IAPX 186, IAPX 286 – UCP cu 16 biți;
- 8087 (IAPX 86/20) – procesor aritmetic în virgulă mobilă;
- 8088 (IAPX 88/10) – UCP cu 16 biți în interior şi 8 biți în exterior;
- 8089 UCP specializat în operații I/E;
- IAPX 86/30, IAPX 88/30 – procesor specializat conținând un sistem de operare în timp real încorporat în HW;
- 8284 – generator de tact;
- 8288 – controler de sistem (generează semnalele de comandă pe bus-uri);
- 8289 – arbitru de bus (coordonează funcționarea mai multor UCP-uri pe acelaşi bus) pentru sisteme multiprocesor;
- 8259A – controler de întreruperi;
- 8237A – controler DMA;
- 8282, 8283 – circuite lactch (8biți);
- 8286, 8287 – circuite buffer (8biți).

În figura 4.7 este prezentat un microsistem realizat cu microprocesorul I8086 conectat în modul "minim".

Din această figură se observă existența celor trei magistrale : magistrala de comenzi, magistrala de adrese şi magistrala de date. Magistrala de adrese şi cea de date, datorită faptului că sunt multiplexate, necesită circuite suplimentare pentru generare : circuitul I8282 latch şi circuitul I8286 buffer.

În figura 4.8. este prezentat modul în care microprocesorul I8086 poate fi conectat în modul "maxim". Aşa cum s-a arătat, în acest mod sunt generate în mod suplimentar semnale de comandă a magistralelor cu ajutorul circuitului controler de sistem I8288.

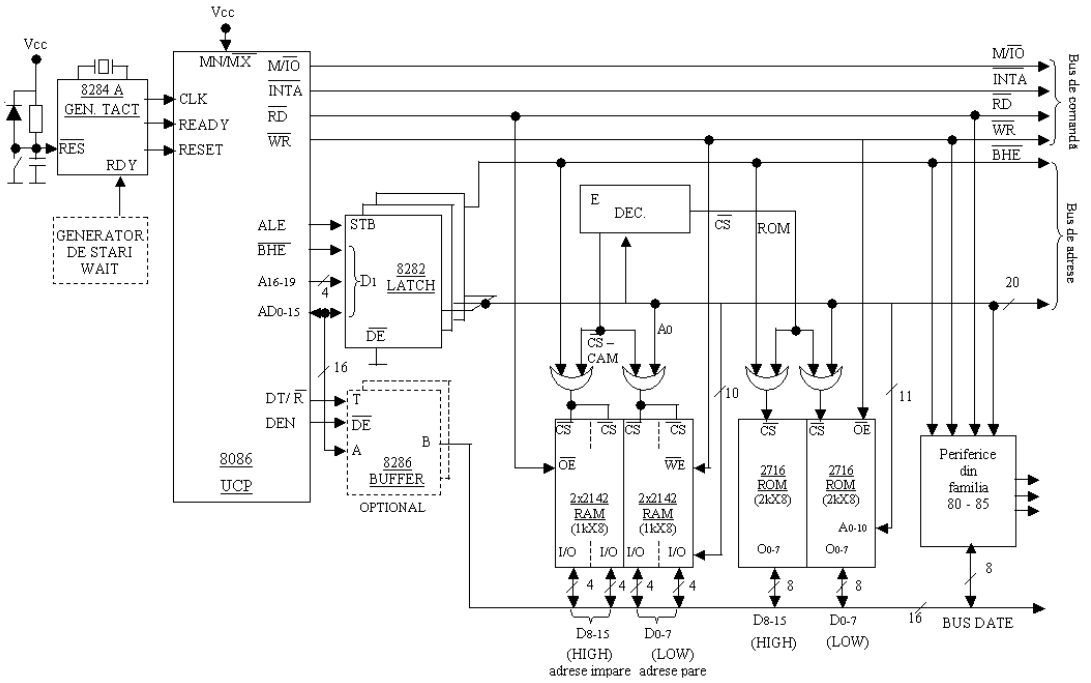


Figura 4.7. Microsistem cu microprocesor I8086 conectat în modul "minim".

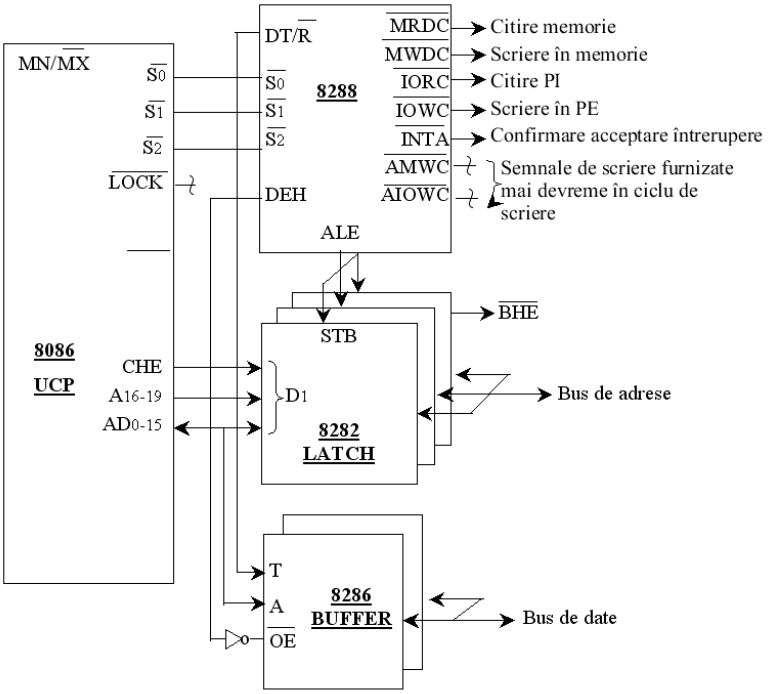


Figura 4.8. Conectarea microprocesorului I8086 în modul "maxim"

Organizarea memoriei principale

Microprocesorul poate adresa direct 1Mo de memorie la adresele 0 ÷ 0FFFFFH.

Tipuri de date memorate:

- 8 biți (octeți)
- 16 biți (cuvinte)
- 32 biți (cuvinte duble sau pointer-i)

Nu există restricții privind plasarea acestor date în memorie (ele pot "începe"la orice adresă). În funție de plasarea datelor la scheme pare sau impare UCP va executa automat numărul de cicluri necesare pentru citirea lor.

Convenția de reprezentare a datelor multi-cuvânt în memorie este cea standard la microprocesoarele INTEL: octeții mai puțin semnificativi sunt plasați la adrese mai mici (în grupul de octeți alocat cuvântului).

Memoria lui I 8086 este segmentată.

Pentru generarea adresei fizice AF, BIU execută operația dată de relația:

$$AF = S \times 2^4 + O \qquad (4.1)$$

S = conținutul registrului segment

O = adresa efectivă (offset)

Se impune deci restricția ca un segment să înceapă la o adresă absolută multiplu de 16.

Registrele segment pot fi implicite sau explicite (la dorința programatorului):

Tipul de referire la memorie	Registrul segment utilizat implicit	Alte registre utilizate	OFFSET (adresa relativă în cadrul segmentului)
"FETCH"	CS	_	IP
* Operații cu stiva	SS	_	SP
* Date variabile (cu excepțiile de mai jos)	DS	CS, SS, ES	Adresa efectivă
* Sursa la operațiile cu şiruri	DS	CS, SS, ES	SI
* Destinația la operațiile cu şiruri	ES	_	DI
* BP folosit ca registru de bază	SS	CS, DS, ES	Adresa efectivă

Utilizarea altor registre segment față de cele utilizate implicit trebuie indicată de programator în instrucțiunea respectivă cu ajutorul unor prefixe speciale (1 octet).

Memoria stivă

Stiva sistemului se organizează în memoria principală. Într-o aplicație pot exista mai multe stive (fiecare cu dimensiunea maximă de 64Ko). Adresa de bază a stivei (valoarea inițială a registrului SP într-un program) este diferită de adresa de bază a segmentului de memorie alocat stivei.

Modul de organizare al stivei microprocesorului I8086 este arătat în figura 4.9.

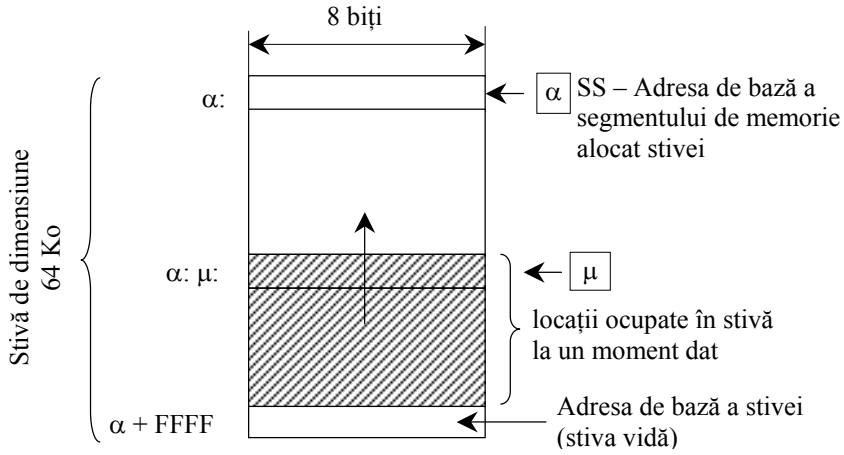


Figura 4.9. Organizarea stivei microprocesorului 8086.

Anumite zone din memoria principală a sistemului sunt rezervate pentru diferite utilizări de către unitatea centrală.

Adrese rezervate:

- 0 ÷ 13H = dedicate pentru întreruperi interne;
- 14H ÷ 7FH = rezervate pentru dezvoltări ale familiei de componente (rezervate de firma INTEL);
- 0FFFF0H ÷ 0FFFFBH = dedicate pentru instrucțiunile executate resetarea procesului;
- 0FFFFC ÷ 0FFFFFH = rezervate de firmă

Porturile de intrare / ieşire (I / E)

Spațiul de adrese de intrare / ieşire este separat de spațiul adreselor memoriei. Spațiul este nesegmentat şi cu dimensiunea de 64 Ko ceea ce asigură posibilitatea de adresare directă a unui mare număr de porturi I/E cu 8 sau 16 biți.

Într-un transfer I/E, UCP citeşte/scrie 16 biți/ciclu de bus dacă portul I/E este localizat la adresa pară şi 8 biți/ciclu dacă portul I/E este localizat la adresă impară.

Facilități pentru lucru în sisteme multi-master

Facilitățile pentru lucrul multi-master sunt prezentate în continuare.

- Interzicerea accesului la bus a altor module master cu ajutorul semnalului LOCK. Bus-ul se utilizează prin diviziune în timp,

întreţeserea ciclurilor de acces la bus a diverselor module "master" se face la nivel de ciclu de bus (şi nu la nivel de ciclu de instrucţiune).
LOCK = 0 indică faptul că 8086 execută o instrucţiune ce nu poate fi întreruptă (instrucţiunea e prefixată "LOCK" – ex. actualizarea unui pointer de 4 octeţi).

- Utilizarea liniilor RQ/GT0 şi RQ/GT1
  Liniile RQ/GT0 şi RQ/GT1 bidirecţionale permit controlul accesului la un bus local comun mai multor procesoare. Protocolul de cerere/cedare a bus-ului decurge astfel:
  - procesorul solicitator cere acces prin generarea unui impuls (RQ=0) către UCP;
  - UCP (la sfârşitul ciclului maşină curent) cedează bus-ul şi răspunde prin impulsul (GT=0);
  - după terminarea accesului, celălalt procesor eliberează bus-ul informând asupra acestui fapt printr-un nou impuls (GT=0).

Sistemul de întreruperi

- Clasificarea întreruperilor
  Întreruperile interne (software)

  - INT n, unde tipul este TIP=n, n ∈ {0, ..., 255}
  - INTO (INTerrupt ou Overflow), provoacă o întrerupere TIP=4 dacă flag-ul overflow a fost setat în urma apariţiei unor depăşiri la efectuarea operaţiilor aritmetice;
  - IDIV, cu TIP=0, generată automat dacă apare o eroare de depăşire la operaţia de împărţirre;
  - SINGLE STEP, cu TIP=1, se generează automat după execuţia fiecărei instrucţiuni, dacă flag-ul TF ("test flag") a fost setat în "1". Acest tip de întrerupere practic UCP în modul de lucru "pas cu pas", util pentru depanarea programelor.

Întreruperile externe (hardware) se generează prin aplicarea unor semnale pe intrările de întrerupere.

  - NMI – pentru întreruperile nemascabile, cu TIP=2; semnalul este activ pe frontul pozitiv.
  - INTR – pentru întreruperi mascabile (prin flag-ul IF din registrul de flag-uri). Semnal activ prin nivel logic 1; acesta trebuie menţinut activ până la recunoaşterea întreruperii de către UCP. Tipul întreruperii externe mascabile se generează de către dispozitivul întrerupător (de exemplu controlerul de întreruperi I8259A).

- Servirea întreruperilor

Tipul unei întreruperi este utilizat de către UCP ca adresă relativă într-un tablou de "pointer"-i, amplasat în memoria principală între adresele 0÷3FFH (256x4 octeţi).

Un pointer conține adresa logică pentru servirea întreruperii. Pentru a efectua saltul la această adresă se efectuează următoarele operații:

IP ← TP(T+1,T); CS←TP(T+3, T+2)
T=4xTIP

TP(α+1, α) reprezintă cuvântul cu 16 biți format din octeții de adresă α-1 şi α din tabloul de "pointer"-i. Efectul este asemănător cu cel al instrucțiunii CALL intersegment

Prioritățile de servire pentru diversele tipuri de întreruperi (care pot apare eventual simultan) rezultă din organigrama alăturată.

Ciclurile "INTA" pentru întreruperile externe mascabile (INTR) – pe durata celor două cicluri INTA, semnalul LOCK este activ. Vectorul de întrerupere este în acest caz numărul indicator al tipului şi este citit de către UCP în al doilea ciclu "INTA".

În figura 4.10 se prezintă modul de alocare în memorie a vectorilor de întrerupere.

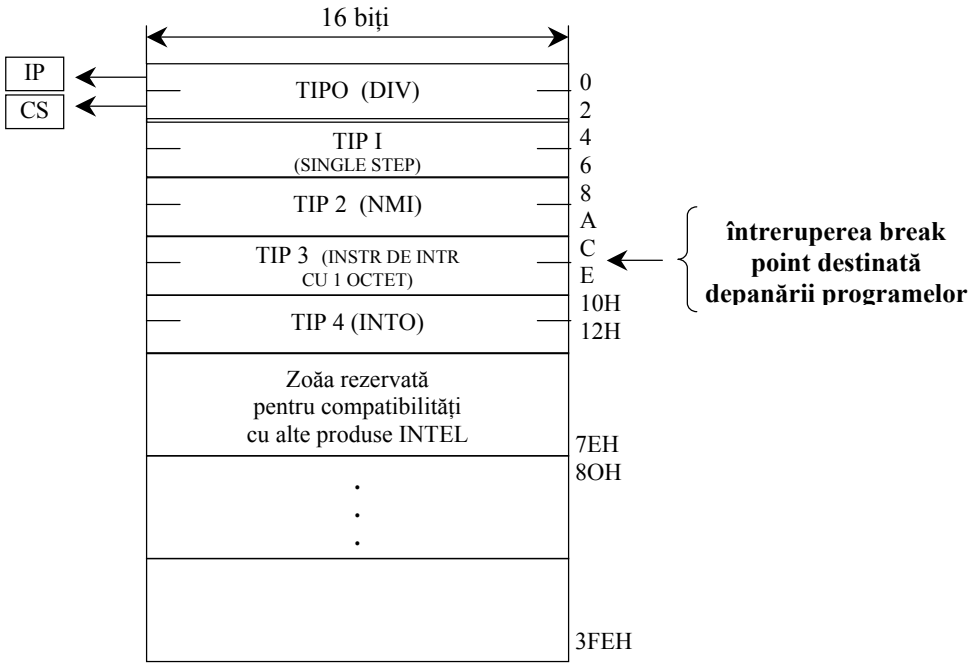


Figura 4.10. Tabel de "pointer"-i pentru întreruperi

În figura 4.11 se prezintă organigrama corespunzătoare servirii unei întreruperi de către unitatea centrală I8086.

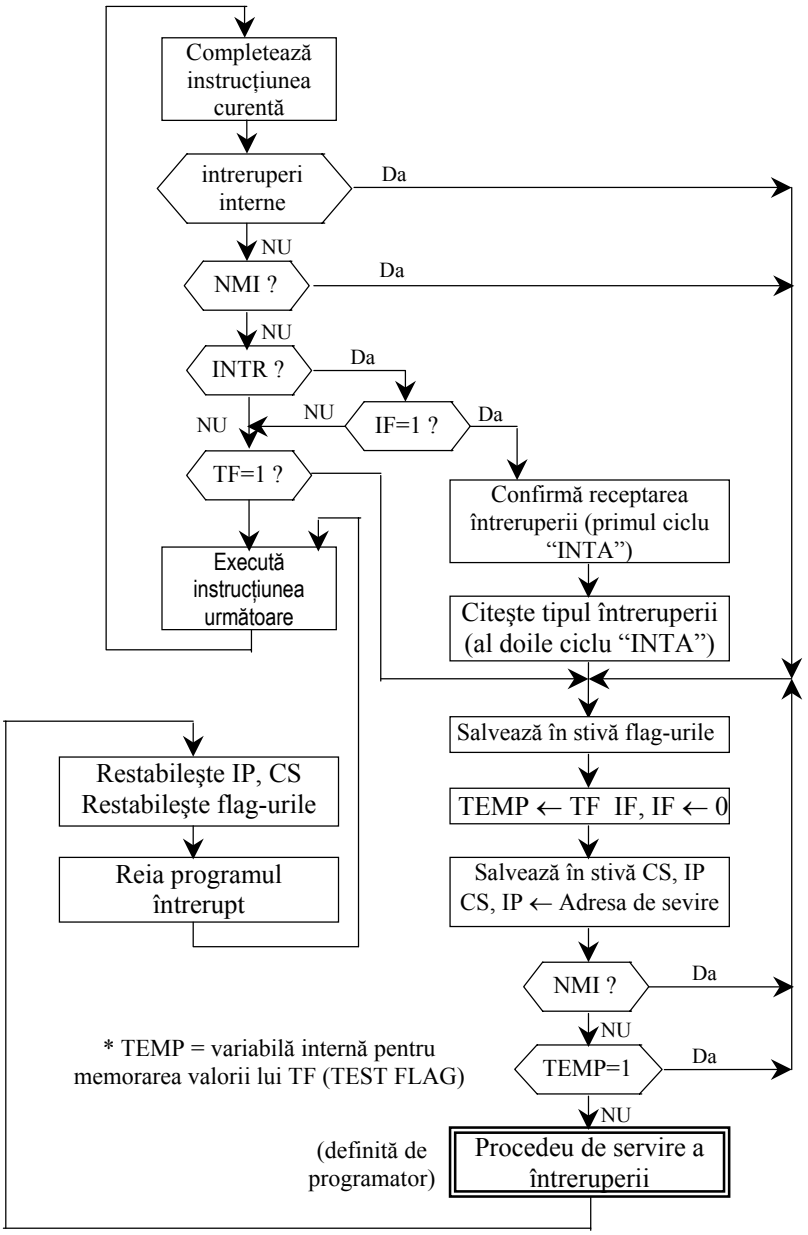


Figura 4.11. Organigrama servirii unei întreruperi.

Inițializarea unității centrale. Starea HALT. Utilizarea intrării TEST

- 1 pe RESET determină:

CS ← FFFFH, IP ← 0 ; ceea ce determină generarea adresei fizice 0FFFF0H cu adresă absolută a primei instrucțiuni executată de I8086.

Intreruperile mascabile se invalidează IF ← 0.

- HALT (la apariția instrucțiunii) stopează toate activitățile UCP până ce apare o întrerupere sau semnal RESET. În această sare o cerere de bus este (HOLD=1) este recunoscută şi şi acceptată de către UCP.
- Intrarea TEST este testată de instrucțiunea WAIT şi dacă TEST ≠ 0, atunci procesorulaşteaptă (într-o stare inactivă) până ce TEST ← 0. În acest timp, se pot servi întreruperile apărute la intrări, după care se reia aşteptarea. Intrarea TEST poate servi pentru sincronizarea UCP cu evenimente externe.

4.4. Procesoare de semnal digitale

O dată cu dezvoltarea tehnicii digitale şi în special cu creşterea performamțelor microprocesoarelor s-a dezvoltat un segment aparte de componente numerice specializate în prelucrarea numerică a semnalelor. Asfel de componente au căpătat numele de procesoare de semnal digitale, prescurtat DSP (Digital Signal Processor). Astfel prescurtarea DSP capătă d dublă semnificație; una din semnificații se referă la tehnica prelucrării semnalelor în format numeric iar cea de-a doua semnificație se referă la dispozitivele (procesoarele) specializate, destinate implementării tehnicilor de prelucrare a semnalelor în format numeric.

Un procesor destinat prelucrării numerice a semnalelor reprezintă o unitate centrală specializată care este capabilă să execute cu viteză ridicată secvențe de instrucțiuni cum sunt cele de deplasare a conținutului unui registru şi adunarea conținutului acestuia sau cele de înmulțire şi adunare care sunt operații uzuale în algoritmii de prelucrare a semnalelor.

Spre deosebire de un microprocesor care este o unitate centrală de uz general, un dispozitiv DSP este destinat anumitor domenii de utilizare definite de tehnica de prelucrare numerică a semnalelor având instucțiuni speciale, adaptate scopului propus iar aplicatiile sunt executate în timp real ceea ce presupune untimp scurt de execuție a instrucțiunilor şi o structură specială a sistemului de întreruperi. Pe de altă parte, un dispozitiv DSP lucrează de obicei într-o structură ierarhizată de calcul în care dispozitivul DSP asistă un microprocesor de uz general.

Deşi există o mare varietate de dispozitive DSP acestea sunt proiectate cel mai adesea să îndeplinească aceleaşi funcții de bază. Rezultă că există un set de caracteristici de bază pentru toate procesoarele DSP, caracteristici de bază care pot fi împărțite în trei categorii:

- dispozitivele sunt specializate în efectuarea cu viteză ridicată a operațiilor aritmetice;
- sunt prevăzute cu mecanisme diversificate de transfer a datelor din şi către lumea reală;
- sunt utilizate arhitecturi cu acces multiplu la memorie.

Activitățile unui dispozitiv DSP impun efectuarea unor operații specifice cum sunt: (figura 4.12):

- adunări şi înmulțiri;

- întârziri;
- manipulari de matrici.

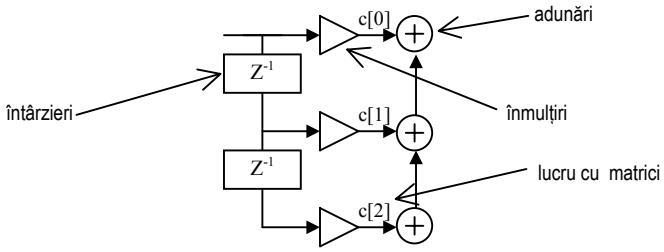


Fig. 4.12. Operații specifice prelucrării digitale a semnalelor

Operațiile de adunare şi de înmulțire sunt folosite pentru:

- extragerea simultană a doi operanzi;
- realizarea adunării şi înmulțirii (de obicei simultan);
- stocarea rezultatului sau reținerea acestuia în vederea repetării operației.

Întârzierile sunt folosite pentru reținerea unei valori pentru utilizare ulterioară.
Manipularea matricilor este necesară pentru:

- extragerea operanzilor aflați în locații sucesive de memorie;
- copierea datelor de la memorie la memorie.

Această suită de operații sunt de obicei executate de dispozitivul DSP astfel:

- executarea în paralel a operațiilor de înmulțire şi adunare;
- acces multiplu la memorie în scopul extragerii simultane a doi operanzi şi memorarea rezultatului;
- folosirea mai multor regiştrii pentru reținerea temporară a datelor;
- generarea eficientă a adreselor pentru utilizarea matricilor;
- facilități speciale cum sunt întârzierile sau adresarea circulară.

Pentru realizarea operațiilor aritmetice dispozitivele DSP au o structură specială, de mare viteză, a unității logice şi aritmetice deoarece ele trebuie să execute simultan adunări şi înmulțiri. Din acest motiv dispozitivele DSP au de regulă o structură specială a circuitelor de adunare şi înmulțire ce permite acestor operații în paralel, de către o singură instrucțiune, aşa cum este arătat în figura 4.13.

Întârzierea permite ca valoarea intermediară rezultată în urma calculelor să fie memorată pentru o utilizare ulterioară. Acest lucru poate fi necesar, de exemplu, când trebuie calculat un total; totalul poate fi păstrat în procesor pentru a evita scrierile şi citirile repetate la memorie. Din acest motiv procesoarele DSP au un număr relativ mare de regiştrii, în virgulă fixă sau virgulă mobilă, care pot fi folosiți pentru stocarea valorilor intermediare.

Utilizarea matricilor permite ca datele să poată fi manevrate eficient în locații succesive de memorie. Acest lucru necesită generarea adresei de memorie unde se găseşte valoarea următoare şi pentr aceasta dispozitivele DSP au regiştrii de adrese ce sunt folosiți pentru păstrarea adresei şi care permit generarea adresei următoare în mod eficient.

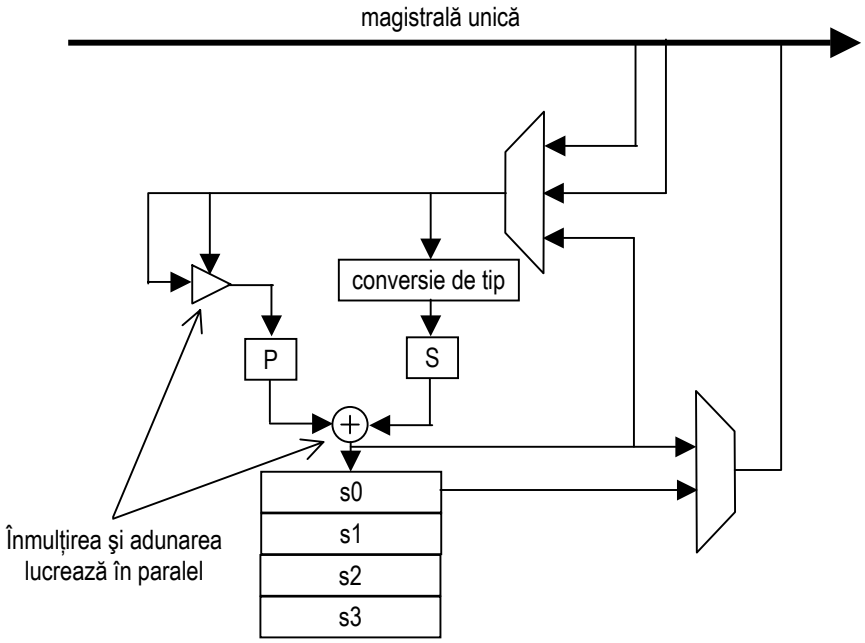


Fig. 4.13. Fluxul de date la realizarea inmulțirii şi adunării

TABELUL 4.2.

Simbol	**Adresare**	**Observații**
*rP	adresare indirectă prin registru	este citită data din memorie indicată de adresa conținută în registrul rP
*rP++	postincrementare	după citirea datei din memorie, adresa din registrul rP este incrementată, în aşa fel încât să indice adresa următoarei date din matrice
*rP--	postdecrementare	după citirea datei din memorie, adresa din registrul rP este decrementată, în aşa fel încât să indice adresa următoarei date din matrice
*rP++rI	postincrementare în funcție de registru	după citirea datei din memorie, adresa conținută în registrul rP este incrementată cu valoarea conținută de registrul rl, adresa obținută indicând noua valoare ce urmează a fi citită din memorie further down the array
*rP++rIr	cu biți inversați	după citirea datei din memorie, valoarea ce indică adresa este incrementată pentru a indica valoarea următoare din matrice, biții adresei fiind aşezați în ordine inversă

Posibilitatea generării eficiente a unei adrese noi este o facilitate caracteristică a procesoarelor DSP. În mod obişnuit adresa următoare poate fi generată pe durata

extragerii sau stocării datelor. Procesoarele DSP au un set bogat de instrucțiuni pentru generarea adreselor. Un exemplu de astfel de instrucțiuni este dat în tabelul 4.2.

Sintaxa în limbaj de asamblare pentru instrucțiunile prezentate în tabelul 4.2 este foarte asemănătoare cu limbajul C. De câte ori un operand este extras din memorie prin utilizarea modului de adresare indirect prin regiştrii, registrul de adresă poate fi incrementat pentru a indica următoarea valoare necesară din matrice. Incrementarea adresei este liberă – nu sunt implicate resurse pentru calculul adresei – şi mai multe astfel de adrese pot fi generate într-o singură instrucțiune. Generarea adreselor este un factor important în creşterea vitezei procesoarelor DSP şi a operațiilor specializate ale acestora.

Ultimul mod de adresare – cu biți inversați – arată cât de specializate pot fi procesoarele DSP. Adresarea cu inversare de biți este necesară atunci când un tabel de valori este reordonat prin inversarea ordinii biților de adresă astfel:

- inversarea ordinii biților în fiecare adresă;
- amestecarea datelor prin inversarea biților, adresarea făcându-se în ordine crescătoare.

O astfel de operație este utilizată exclusiv la calculul tansformatei Fourier rapide. Se poate spune deci, că dispozitivele DSP sunt proiectate special pentru a calcula în mod eficient transformata Fourier rapidă.

Din punct de vedere al legăturii unității centrale cu mediul extern, un dispozitiv DSP este prevăzut cu mai multe interfețe interne (on chip) care lucrează într-un sistem specializat de întreruperi. Cele mai frecvente interfețe ale unui dispozitiv DSP sunt:

- intrări/ieşiri numerice;
- convertoare analog-numerice (CAN);
- convertoare numeric-analogice (CNA);
- comparatoare;
- numărătoare programabile;
- interfațe seriale sincrone de mare viteză pentru conectarea perifericelor externe sau a dispozitivelor DSP;
- interfețe seriale standard de tip RS-232 pentru lucrul pe port serial asincron.

4.4.1. Procesorul de semnal digital, TMS320F240

Circuitul DSP de tip TMS320F240 este un circuit integrat pe scară foarte largă în tehnologie CMOS şi se compune din trei unități funcționale: unitatea centrală de tip C2xx DSP, memoria internă şi unitatea circuitelor periferice. De asemenea, în afară de aceste unități funcționale sunt prevăzute o serie de facilități sistem care se referă la gestionarea memoriei, initializarea blocurilor, intreruperi, controlul intrărilor/ieşirilor, generearea ceasului intern şi comutarea în regim de consum redus. Circuitul TMS320F240 utilizează o arhitectură Harvard avansată pe folosirea a două magistrale separate: de date şi de program.

În continuare sunt prezentate foarte sumar principalele facilități oferite de circuitul DSP în scopul formării unei imagini asupra structurii complexe a acestuia:

- unitate centrală de tip C2xx cu arhitectură paralelă ce oferă posibilitatea execuției în paralel a instrucțiunilor, posibilitatea de prelucrare a mai multor fluxuri de informații simultan, execuția instrucțiunilor într-un singur ciclu maşină (inclusiv a inmulțirii);
- memorie de program inclusă în circuitul integrat de 16KB (cuvinte de 16 biți) de tip flash EEPROM şi memorie RAM cu acces dublu pentru program/date de 544 cuvinte de 16 biți;
- oscilator extern de 10MHz, bucla PLL internă a DSP lucrând la 20MHz;
- 28 de intrări/ieşiri numerice;
- două convertoare analog-numerice pe 10 biți ce pot lucra simultan, numărul intrărilor analogice fiind extins prin multiplexare la 16;
- trei numărătoare programabile independente pe 16 biți;
- interfață serială sincronă de mare viteză pentru conectarea perifericelor externe;
- interfață serială standard de tip RS-232 pentru lucrul pe port serial asincron;

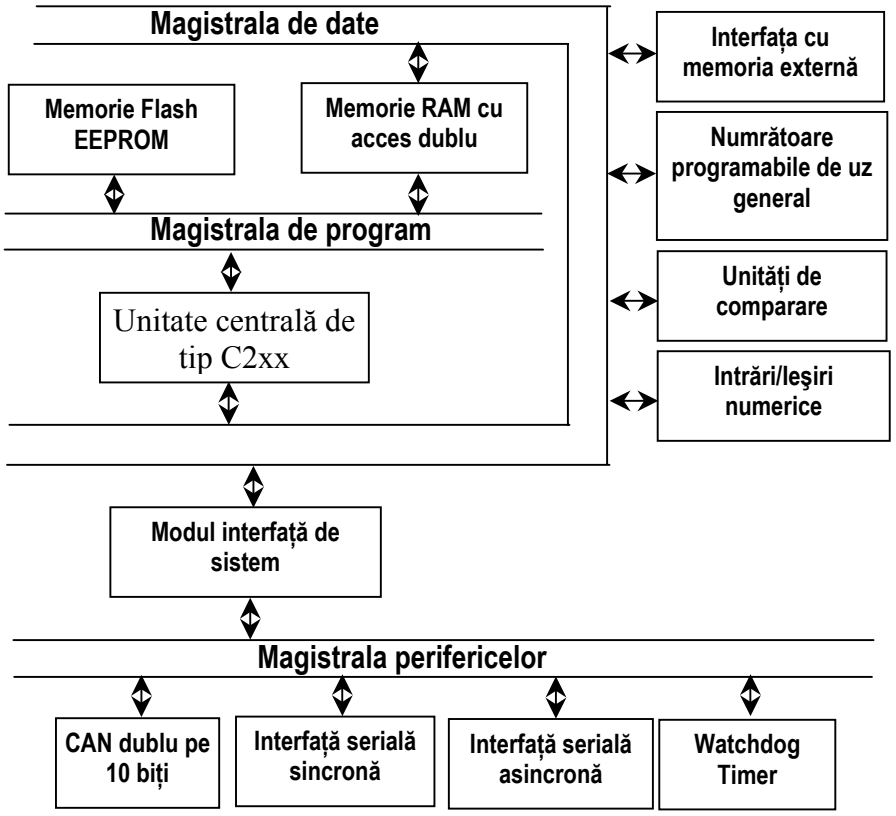


Fig. 4.14. TMS320F240 schema bloc

Schema bloc simplificată a circuitului DSP de tip TMS320F240 este prezentată în figura 4.14.

Prezența celor două magistale, magistrala de date şi magistrala de program permite prelucrarea simultană a instrucțiunilor de program şi a datelor.

Organizarea memoriei procesorului este prezentată în tabelele 4.3, 4.4 şi 4.5.

TABEL 4.3.

Memorie Program

Hex	
0000 003F	Intreruperi (Memorie externă)
0040 FDFF	Memorie externă
FE00 FEFF	On-Chip DARAM B0 (CNF=1) sau **Memorie externă** (CNF=0)
FF00 FFFF	Rezervat

MP/$\overline{MC}$ =1

Mod microprocesor

TABEL 4.4.

Memorie Program

Hex	
0000 003F	Intreruperi On-Chip Flash
0040 3FFF	Memorie Flash On-Chip
4000 FDFF	Memorie externă
FE00 FEFF	On-ChipDARAM B0 (CNF=1) sau **Memorie externă** (CNF=0)
FF00 FFFF	Rezervat

MP/$\overline{MC}$ =0

Mod Microcomputer

Dispozitivul DSP este prevăzut cu un pin MP/$\overline{MC}$ (microprocessor mode / microcomputer mode) cu ajutorul căruia se pot obține două configurații ale memoriei de program:

- MP/$\overline{MC}$ = 1 (microprocessor mode) modul în care memoria on chip flash şi ROM sunt dezactivate (Tabelul 4.3);
- MP/$\overline{MC}$ = 0 (microcomputer mode) modul în care memoria on chip flash şi ROM sunt activate (Tabelul 4.4).

Bitul CNF care se găseşte în registrul de stare ST1 permite activarea memoriei RAM cu acces dublu on chip, blocul 0 (CNF = 1) sau a memoriei externe (CNF = 0).

Organizarea memoriei de date este arătată în tabelul 4.5. Se observă că în memoria de date vom găsi corespondentul regiştrilor unității centrale şi a perifericelor ceea ce permite lucrul cu aceştia identic cu lucrul cu orice altă locație de memorie. Acest fapt duce la creşterea vitezei de execuție şi simplitatea programării.

Împărțirea memoriei în două tipuri: memorie de program şi memorie de date impune ca orice program să fie alcătut din două segmente corespunzătoare acestor tipuri, amestecarea instrucțiunilor cu datele nefiind permisă.

TABEL 4.5.
Memorie Date

Hex	
0000 005F	Regiştrii şi zonă rezervată
0060 007F	Memorie On-Chip DARAM B2
0080 01FF	Rezervat
0200 02FF	Memorie On-Chip DARAM B0 (CNF=0) sau zonă rezervată (CNF=1)
0300 03FF	Memorie On-Chip DARAM B1
0400 07FF	Rezervat
0800 6FFF	Ilegal
7000 73FF	Regiştrii perifericelor (System, WD, ADC, SPI, SCI, Intreruperi, I/O)
7400 743F	Regiştrii perifericelor (Manager-ul de evenimente)
7440 77FF	Rezervat
7800 7FFF	Ilegal
8000 FFFF	Memorie externă

Organizarea întreruperilor unității centrale

Sistemul de întreruperi externe ale unității centrale (miezul procesorului C2xx - CPU) este format din şase întreruperi mascabile (INT1-INT6) şi una nemascabilă (NMI). Prioritatea maximă o are întreruperea NMI iar prioritățile întreruperilor mascabile scad de la INT6 la INT1, INT1 având prioritatea minimă.

Fiecărei întreruperi îi corespunde un bit în registrul fanioanelor de întrerupere a unității centrale (IFR) şi fiecărei întreruperi mascabile îi corespunde un bit în registrul măştilor unității centrale (IMR). O întrerupere mascabilă este mascată (nu va genera o întrerupere către miez) când bitul corespunzător în IMR este 0. De asemenea miezul procesorului mai are un bit pentru mascarea generală a întreruperilor (INTM) în registrul de stare ST0. Când INTM este setat la 1 toate întreruperile mascabile sunt dezactivate.

Răspunsurl la întrerupere a miezului C2xx

Când apare o tranziție de la unu la zero pe o intrare de întrerupere a miezului, bitul corespunzător al fanionului din IFR este setat în 1. O întrerupere este generată către miez dacă aceasta nu este mascată - întreruperile globale sunt permise (INTM=0) - şi nu există o altă întrerupere nemascată de prioritate mai mare în aşteptare (acest lucru însemnând că nu există nici un fanion setat al unei întreruperi nemascate de prioritate mai mare). Fanionul este şters de către hardware o dată ce cererea de întrerupere este trimisă către miez. Un fanion de întrerupere poate fi de asemenea şters de către programul utilizator scriind un 1 în bitul corespunzător.

4.5. Microcalculatoare integrate, microcontrolere

4.5.1. Prezentare generală

Electronica digitală se bazează astăzi în bună măsură pe utilizarea circuitelor integrate VLSI (Very Large Scale Integration – circuite integrate pe scară foarte largă) de tipul microprocesor, microcalculator sau microcontroler.

Dintre acestea se detaşează net microcalculatoarele integrate şi microcontrolerele care înglobează în structura hard pe lîngă o unitate centrală puternică cu facilităţi speciale pentru lucrul în timp real şi periferice de tipul convertoarelor A/D, interfeţe paralele, interfeţe seriale, sisteme de generare a impulsurilor modulate, timere, circuite de supraveghere a duratei de execuţie a programelor, executiv de timp real integrat etc. ceea ce uşureaza mult munca proiectantului şi duce la creşterea substanţială a performanţelor sistemului.

Nu exista o diferenţă netă între microcalculatoare integrate şi microcontrolere. Microcontrolerele prin setul de instrucţiuni mai redus, sunt mai intim legate de aplicaţiile de control, urmărire şi automatizare industrială.

Între microcalculatoarele integrate mai cunoscute se pot enumera: INTEL 8048/8035, MOTOROLA 6801 şi 6805 şi familia ZILOG Z8.

INTEL 8048 a fost primul microcalculator integrat; acest dispozitiv conţine într-o capsulă de 40 de pini urmatoarele:

- o unitate centrală de 8 biţi;
- o memorie RAM de 64 octeţi;
- o memorie ROM de 1 koctet;
- un număr de 27 de linii de I/E (Intrare/iEşire);
- un oscilator pilot;
- un circuit de ceas de 8 biţi.

Setul de instrucţiuni este tributar într-o oarecare masură primului microprocesor de uz general I8080, cum de altfel este şi concepţia hardware-ului, cuprinde: operaţii cu acumulatorul, transferuri de date, salturi, chemări şi reveniri din subrutine, operaţii de intrare/ieşire şi operaţii de control cu registre şi indicatori.

MC 6801 este un microcalculator integrat de 8 biţi care s-a dezvoltat din familia microprocesorului de uz general MC6800 (MOTOROLA).

Instrucţiunile microcalculatorului integrat MC6801 sunt compatibile cu instrucţiunile microprocesorului mamă. Timpul de execuţie a fost redus şi câteva noi instrucţiuni au fost adaugate; între acestea şi instrucţiunea de înmulţire fără semn.

MC6801 poate funcţiona fie ca microcalculator de sine stătător fie ca microcalculator de uz general ce poate utiliza o memorie externă de cel mult 64ko. MC6841 include pe lângă microprocesorul propriu-zis:

- o memorie fixă (ROM) de 2 ko;
- o memorie citeşte/scrie (RAM) de 128 de octeţi;
- un număr de 29 linii de I/E;

- trei circuite de ceas programabile, de 16 biți fiecare;
- un oscilator pilot.

Familia ZILOG Z8 cuprinde o variantă standard Z8611 şi mai multe versiuni de dezvoltare. Principalele caracteristici ale acestei versiuni sunt:

- 43 de tipuri de instrucțiuni;
- 124 de registre de uz general, 4 registre de I/E, 16 registre de control şi configurare;
- 32 de linii de intrare/ieşire;
- un circuit de intrare/ieşire serie;
- 2 circuite de ceas (fiecare ceas având un divizor programabil de 8 biți şi un predivizor de 6 biți);
- 6 întreruperi vectorizate, mascabile şi aranjabile în şir de priorități;
- timp de execuție mediu a instrucțiunilor de 2,2 μs.

Dintre microcontrolerele reprezentative se pot aminti:

- microcontrolerul AVR AT90S2313 al firmei ATMEL
- microcontrolerul PIC 16F87X al firmei MICROCHIP TECHNOLOGY ;

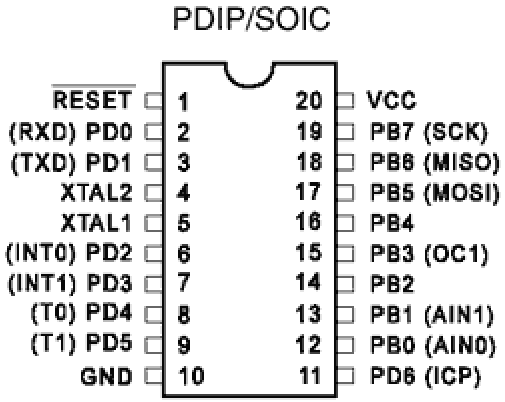
4.5.2. Microcontrolerul AT90S2313

Microcontrolerul AT90S2313 are o memorie FLASH programabila în sistem de 2K octeti.
Caracteristici principale:

- Bazat pe arhitectura AVR @ RISC cu performanțe ridicate şi consum redus:
  - 118 instrucțiuni puternice - majoritatea se execută într-un ciclu maşină;
  - 32 x 8 registre generale de lucru;
  - viteza pană la 10 MIPS @ 10 MHz.
- Memorie nevolatilă de date:
  - Mărimea memoriei Flash: 2K octeti programabilă In sistem;
    - suportă 1 000 cicli de stergere/inscriere;
  - 128 octeți de Memorie SRAM.
  - 128 octeti de memorie EEPROM programabilă In sistem;
    - suportă 1 000 000 cicli de ştergere/înscriere;.
  - Protecție programabilă pe 8 biți a memoriei de date Flash si EEPROM.
- Periferice:
  - un temporizator/numărător cu prescaler separat pe 8 biți;
  - un temporizator/numărător cu prescaler separat pe 16 biți;
  - comparator, moduri de captură şi PWM pe 8, 9 şi 10 biți;
  - comparator analogic on-chip;

  - temporizator Watchdog programabil cu oscilator încorporat in chip;
  - interfață serială SPI pentru programarea In-system;
  - port serial UART full duplex.
- Funcții speciale ale microcontrolerului:
  - consum redus în caz de inactivitate precum şi power-down de economisire a energiei;
  - surse de înrerupere externă şi internă.
- Specificatii:
  - tehnologie CMOS - consum redus, viteză mare;
  - mod de operare complet static.
- Consum de energie la 4 MHz, 3V, 25°C:
  - activ: 2.8 mA;
  - inactiv: 0.8 mA;
  - power-down: <1 μA.
- Intrări/Ieşiri şi Capsula:
  - 15 linii de intrare/iesire programabile;
  - capsula PDIP si SOIC de 20 pini.
- Tensiuni de operare:
  - 2.7 - 6.0V (AT90S2313-4);
  - 4.0 - 6.0V (AT90S2313-10).
- Viteza de operare:
  - 0 - 4 MHz (AT90S2313-4);
  - 0 - 10 MHz (AT90S2313-10).

Configurația pinilor:



Descrierea pinilor microcontrolerului:

- VCC Pin de alimentare (tensiune pozitiva).
- GND Pin de masă.
- Portul B (PB7..PB0). Portul B este un port I/O bidirectional pe 8 biți. La ieşirile terminalelor se pot configura rezistoare pull-up (legare software la plusul sursei de alimentare a rezistoarelor interne – selectabil pentru fiecare pin in parte). PB0 şi PB1 de asemenea

serversc ca intrare pozitivă (AIN0) şi intrare negativă (AIN1), pentru comparatorul analogic in chip. Portul B furnizează la ieşire un curent de 20 mA asftel încât poate fi legat direct la un afişor LED. Atunci când pinii PB0 la PB7 sunt utilizaţi ca intrări şi din exterior sunt comandaţi cu semnal digital LOW, ei vor consuma energie daca rezitenţele interne de pull-up sunt activate. Dupa un Reset pinii Portului B vor fi în starea de inaltă impedanţă chiar dacă semnalul de ceas nu este activ.

- Portul D (PD6..PD0). Portul D posedă şapte pini bidirecţionali de I/O (curent debitat de 20 mA) cu rezistenţe pull-up. Pinii portulul D intră în starea de inaltă impedanţă în cazul acţionării Resetului.

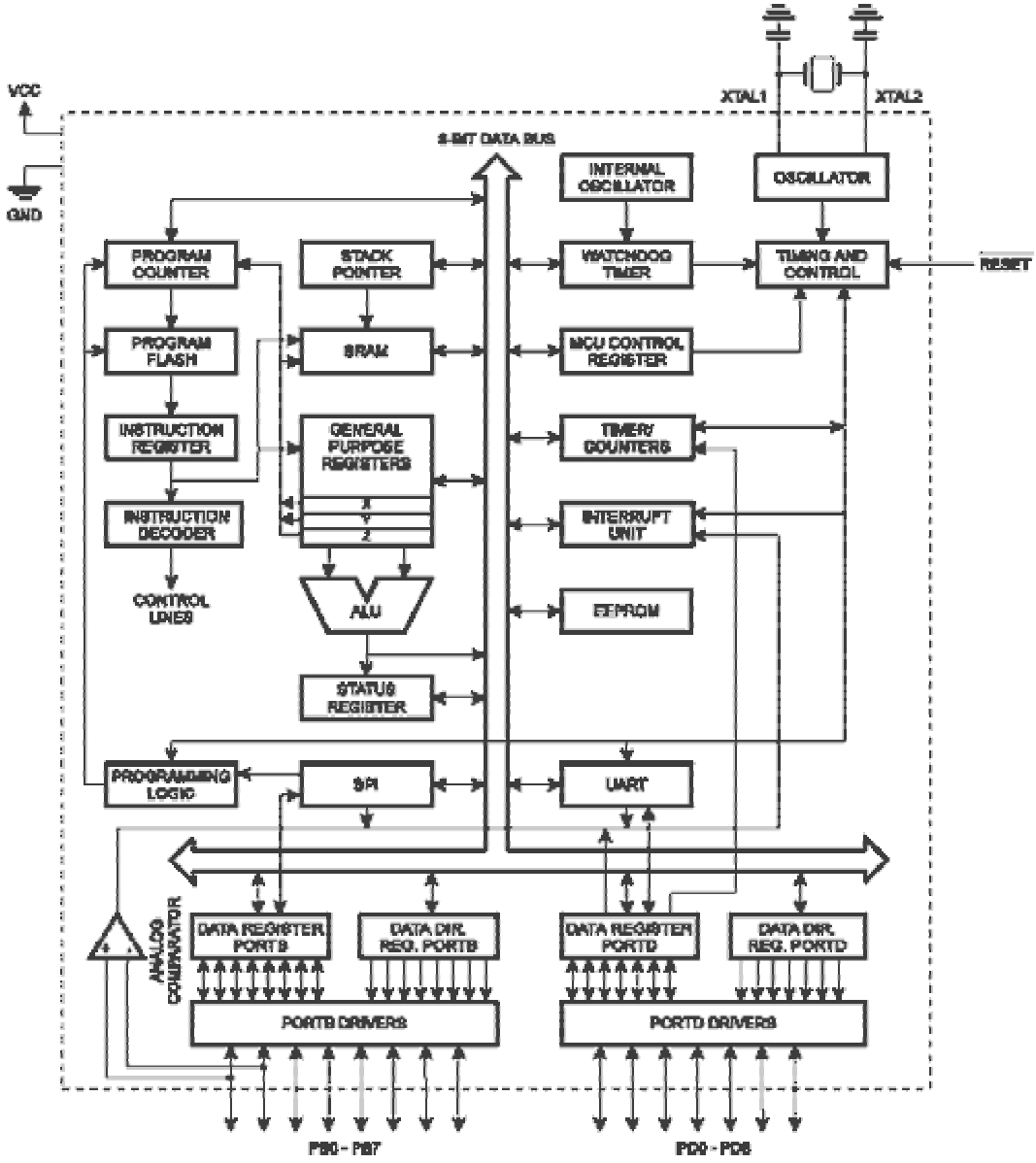


Figura 4.15. Schema bloc a microcontrolerului AT90S2313

- RESET. Pin de intrare. Un semnal LOW pe acest pin pe o durată de 50 ns va genera un Reset, chiar daca semnalul de clock nu este activ. Un semnal mai scurt nu garanteaza conditia de Reset.
- XTAL1. Intrare inversată pe un oscilator amplificat şi intrare în circuitul de semnal de ceas.
- XTAL2. Ieşire inversată de la un amplificator oscilator.

AT90S2313 este un microcontroler CMOS pe 8 biți, cu consum redus bazat pe arhitectura AVR RISC. Executând instrucțiuni puternice într-un ciclu maşină, AT90S2313 execută aproape 1 million de instrucțiuni pe secundă.

Miezul AVR combină setul bogat de instrucțiuni cu 32 de registre de lucru generale. Toate cele 32 de registre sunt conectate direct la Unitatea Aritmetică şi Logică (ALU), astfel este permisă accesarea a două registre independente într-un singur ciclu maşină. Arhitectura rezultată este mult mai eficientă din punct de vedere al codului astfel încât se obține o putere de calcul până la zece ori mai mare decât cel al arhitecturii microcontrolerelor CISC.

În modul "inactiv" procesorul se opreşte în timp ce memoria SRAM, temporizatorii/numararorii, portul SPI şi întreruperile de sistem continuă să funcționeze.

În modul "power-down" microcontrolerul salvează conținutul registrelor şi opreşte oscilatorul, se opresc toate funcțiile din chip pană la următorea întererupere externă sau un Reset hardware.

Microcontrolerul este fabricat în tehnologia memorie nevolatilă de densitate înalta. Memoria Flash internă, programabilă In sistem permite reprogramarea In sistem utilizând interfața serială SPI sau un programator de memorii nevolatile convențional.

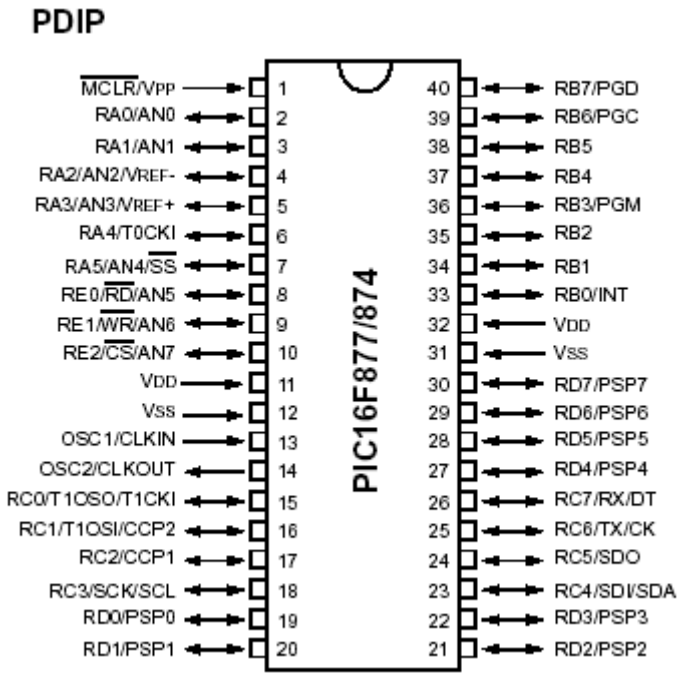
4.5.3. Microcontrolerul PIC 16F877

Microcontrolere de 28/40-Pini cu 8-Bit CMOS FLASH.
Specificațiile unității centrale:

- unitate centrala (CPU) de tip RISC;
- 35 de instrucțiuni;
- toate instrucțiunile se execută într-un ciclu al unității centrale, cu excepția instrucțiunlor de ramificare a programului care se execută în două cicluri CPU;
- frecvența de ceas: 20 MHz (durata ciclului instrucțiunii de 200 ns);
- memoria:
  - maximum 8k cuvinte de 14 biți de memorie FLASH;
  - maximum 368 octeți de memorie de date - Data Memory - (RAM);
  - maximum 256 octeți de memorie de date EEPROM;
- microcontrolerul este compatibil pin la pin cu PIC16C73B/74B/76/77
- posibilitatea tratării întreruperilor de la 14 surse de întrerupere;
- stivă hardware cu opt nivele;
- moduri de adresare directă, indirectă şi relativă;

- Power-on Reset (POR) – reset la aplicarea tensiunii de alimentare;
- Power-up Timer (PWRT) – temporizare la aplicarea tensiunii de alimentare, şi Oscillator Start-up Timer (OST) – stabilizarea oscilatorului la pornire;
- Wtchdog Timer (WDT) cu propriul oscilator RC on-chip pentru funcționare sigură;
- mecanism de protecție a codului programabil;
- mod SLEEP pentru economisirea energiei;
- opțiuni selectabile pentru oscilator;
- tehnologie CMOS FLASH/EEPROM de consum redus şi de viteză ridicată;
- circuitul este în întregime de tip static;
- In-Circuit Serial Programming (ICSP) - programarea serială directă a circuituui – prin intermediul a doi pini;
- posibilitate de programare serială cu o singură tensiune de 5V;
- In-Circuit Debugging – depanare directă la circuit – prin intermediul a doi pini;
- acces scriere/citire a procesorului la memoria de program;
- domeniul tensiunilor de alimentare: 2,0V la 5,5V;
- curent maxim absorbit: 25mA;

Conexiunile externe



Microcontrolerele PIC16F876/873 au 28 de pini iar microcontrolerele PIC16F877/874 au 40 de pini. Portul paralel slave nu este implementat la dispozitivele cu 28 de pini.

Descrierea pinilor pentru PIC16F874 şi PIC16F877

Numele pinului	DIP Nr. pin	PLCC Nr. pin	tip I/O/P	Tip buffer	Descriere
OSC1/CLKIN	13	14	I	ST/CMOS(4)	Intrarea oscilatorului cu cristal/sursă externă de ceas
OSC2/CLKOUT	14	15	O	-	Ieşirea oscilatorului cu cristal. Conectarea la cristal sau la rezonator în modul oscilator cu cristal. În modul RC, pinul OSC2 furnizează CLKOUT care are 1/4 din frecvența la OSC1 şi indică durata unui ciclu de instrucțiune.
~~MCLR~~/VPP	1	2	I/P	ST	Intrarea resetului principal – Master Clear (Reset) sau intrarea tensiunii de programare. Acest pin este activ în zero atunci când se aplică RESET.
					PORTA este un port I/O bidirectional
RA0/AN0	2	3	I/O	TTL	RA0 sau intrarea pentru semnal analogic 0.
RA1/AN1	3	4	I/O	TTL	RA1 sau intrarea pentru semnal analogic 1.
RA2/AN2/VREF-	4	5	I/O	TTL	RA2 sau intrarea pentru semnal analogic 2 sau referința negativă a semnalului analogic.
RA3/AN3/VREF+	5	6	I/O	TTL	RA3 sau intrarea pentru semnal analogic 3 sau referința pozitivă a semnalului analogic.
RA4/T0CKI	6	7	I/O	ST	RA4 sau intrarea de ceas a Timer0. Ieşirea este de tip open drain.
RA5/~~SS~~/AN4	7	8	I/O	TTL	RA5 sau intrarea pentru semnal analogic 4 sau selecția slave pentru portul serial sincron.
					PORTB este un port I/O bidirecțional. La PORTB poat fi conectate prin program rezistențe interne la toate intrările.
RB0/INT	33	36	I/O	TTL/ST(1)	RB0 sau pin pentru întreruperea externă
RB1	34	37	I/O	TTL	RB1
RB2	35	38	I/O	TTL	RB2
RB3/PGM	36	39	I/O	TTL	RB3 sau intrare de programare de tensiune scăzută
RB4	37	41	I/O	TTL	Pin cu generarea unei întreruperi la schimbare

RB5	38	42	I/O	TTL	Pin cu generarea unei întreruperi la schimbare sau pin In-Circuit Debugger. Ceasul programării seriale.
RB6/PGC	39	43	I/O	TTL/ST(2)	
RB7/PGD	40	44	I/O	TTL/ST(2)	Pin cu generarea unei întreruperi la schimbare sau pin In-Circuit Debugger. Datele programării seriale.
					PORTC este un port I/O bidirecțional
RC0/T1OSO/T1CKI	15	16	I/O	ST	RC0 sau ieşirea oscilatorului Timer 1 sau intrerea de ceas Timer 1
RC1/T1OSI/CCP2	16	18	I/O	ST	RC1 sau intrarea oscilatorului Timer 1 sau intrerea Capture2/Ieşirea Comare2/ieşirea PWM2
RC2/CCP1	17	19	I/O	ST	RC2 sau intrerea Capture1/Ieşirea Comare1/ieşirea PWM1
RC3/SCK/SCL	18	20	I/O	ST	RC3 sau intrarea/ieşirea ceasului serial sincron pentru modurile SPI şi I^2C
RC4/SDI/SDA	23	25	I/O	ST	RC4 sau intrare de date SPI (în modul SPI) sau data I/O (în modul I^2C)
RC5/SDO	24	26	I/O	ST	RC5 sau ieşire de date SPI (în modul SPI)
RC6/TX/CK	25	27	I/O	ST	RC6 sau ieşirea de date a transmițătorului USART sau ceasul sincron
RC7/RX/DT	26	29	I/O	ST	RC7 sau intrerea de date a tramsmițătorului USART sau datele sincrone
					PORTD este un port bidirecțional I/O sau un port paralel slave atcnci când se interfațează cu bus-ul unui microprocesor
RD0/PSP0	19	21	I/O	ST/TTL(3)	
RD1/PSP1	20	22	I/O	ST/TTL(3)	
RD2/PSP2	21	23	I/O	ST/TTL(3)	
RD3/PSP3	22	24	I/O	ST/TTL(3)	
RD4/PSP4	27	30	I/O	ST/TTL(3)	
RD5/PSP5	28	31	I/O	ST/TTL(3)	
RD6/PSP6	29	32	I/O	ST/TTL(3)	
RD7/PSP7	30	33	I/O	ST/TTL(3)	
					PORTE este un port bidirecțional I/O
RE0/~~RD~~/AN5	8	9	I/O	ST/TTL(3)	RE0 sau comanda de citire pentru portul paralel slave sau intrarea analogică 5

RE1/~~WR~~/AN6 RE2/~~CS~~/AN7	9 10	10 11	I/O I/O	ST/TTL(3) ST/TTL(3)	RE1 sau comanda de scriere pentru portul paralel slave sau intrarea analogică 6 RE2 sau comanda de selacție pentru portul paralel slave sau intrarea analogică 7
VSS	12,31	13,34	P	-	Masa (referința de tensiune) pentru pinii digitali I/O
VDD	11,32	12,35	P	-	Tensiunea de alimentare pozitivă prntru pinii digitali I/O
NC	-	1,17, 28,40		-	Aceşti pini nu sunt conectați intern. Aceşti pini trebuie să rămână neconectați

Legendă: I = intrare O = ieşire I/O = intrare/ieşire P = alimentare
- = nefolosit TTL = intrare TTL ST = intrare trigger Schmitt
~~SEMNAL~~ = semnalul SEMNAL negat

Notă:
1: acest buffer este o intrare trigger Schmitt când este configurat ca întrerupere externă;
2: acest buffer este o intrare trigger Schmitt când este folosit în modul de programare serială;
3: acest buffer este o intrare trigger Schmitt când este configurat I/O de uz general şi ca intrare TTL când este folosit în modul port paralel slave (pentru interfațarea cu magistrala unui microprocesor);
4: aceast buffer este o intrare trigger Schmitt când este configurat în modul oscilator RC şi intrare CMOS în celelalte cazuri.

Caracteristici principale	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Frecvența de lucru	DC-20 MHz	DC-20 MHz	DC-20 MHz	DC-20 MHz
RESET (Întârzieri)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
Memoria de program FLASH (cuvinte de 14 biți)	4k	4k	8k	8k
Memoria de date (octeți)	192	192	368	368
Memoria de date EEPROM (octeți)	128	128	256	256
Întreruperi	13	14	13	14
Porturi I/O	Porturile A,B,C	Porturile A,B,C,D,E	Porturile A,B,C	Porturile A,B,C,D,E
Timer-e	3	3	3	3
Module Captură/Comparare/PWM	2	2	2	2
Comunicație serială	MSSP,	MSSP,	MSSP,	MSSP,

	USART	USART	USART	USART
Comunicație paralelă	-	PSP	-	PSP
Modul de conversie analog-digitală pe 10 biți	5 canale	8 canale	5 canale	8 canale
Set de instrucțiuni	35 instrucțiuni	35 instrucțiuni	35 instrucțiuni	35 instrucțiuni

Organizarea memoriei

Microcontrolerul are 3 blocuri de memorie. Memoria de program şi memoria de date au magistrale separate şi deci se pote realiza accesul simultan la date şi la program.

Contorul de program are 13 biți ce poate adresa un spațiu de memorie program de 8K x 14 biți. Accesarea locațiilor în afara spațiului fizic al memoriei implementate va produce o adresare wraparound.

Vectorul RESET este 0000h iar vectorul de întrerupere este 0004h.

Memoria este paginată iar paginile sunt:

- pagina 0 la adresele 0005h la 07FFh inclusiv;
- pagina 1 la adresele 0800h la 0FFFh inclusiv;
- pagina 2 la adresele 1000h la 17FFh inclusiv;
- pagina 3 la adresele 1800h la 1FFFh inclusiv.

Organizarea memoriei de date

Memoria de date este împărțită în 4 bank-uri ce conțin registrele de uz general (General Purpose Registers) şi registrele funcțiilor speciale (Special Function Registers).

Selecția bank-urilor se face cu ajutorul biților RP1 (STATUS<6>) şi RP0 (STATUS<5>).

RP1:RP0	Bank
00	0
01	1
10	2
11	3

Fiecare bank din memoria statică are 128 de octeți (7Fh). Locațiile la adresele mici sunt rezervate regiătrilor funcțiilor speciale (SFR) iar sub acestea se găsesc regiştrii de uy general (GPR). Fiecare bank are proprii regiştrii SFR dar anumiți regiştrii SFR dintr-un bank se pot găsi şi într-un alt bank pentru reducerea dimensiunii codului şi pentru acces mai rapid la aceşti regiştrii.

Regiştrii de uz general pot fi accesați atât în mod direct cât şi indirect prin intermediul regiştrilor de selecție: File Select Register (FSR).

Harta regiştrilor microcontrolerelor PIC16F877/876

Bank 0		Bank 1		Bank 2		Bank 3	
Adresa de mem.	Registrul din memorie	Adresa de mem.	Registrul din memorie	Adresa de mem.	Registrul din memorie	Adresa de mem.	Registrul din memorie
00h	adresare indirectă(*)	80h	adresare indirectă(*)	100h	adresare indirectă(*)	180h	adresare indirectă(*)
01h	TMR0	81h	OPTION_REG	101h	TMR0	181h	OPTION_REG
02h	PCL	82h	PCL	102h	PCL	182h	PCL
03h	STATUS	83h	STATUS	103h	STATUS	183h	STATUS
04h	FSR	84h	FSR	104h	FSR	184h	FSR
05h	PORTA	85h	TRISA	105h		185h	
06h	PORTB	86h	TRISB	106h	PORTB	186h	TRISB
07h	PORTC	87h	TRISC	107h		187h	
08h	PORTD(1)	88h	TRISD(1)	108h		188h	
09h	PORTE(1)	89h	TRISE(1)	109h		189h	
0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah	PCLATH
0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh	INTCON
0Ch	PIR1	8Ch	PIE1	10Ch	EEDATA	18Ch	EECON1
0Dh	PIR2	8Dh	PIE2	10Dh	EEADR	18Dh	EECON2
0Eh	TMR1L	8Eh	PCON	10Eh	EEDATH	18Eh	rezervat(2)
0Fh	TMR1H	8Fh		10Fh	EEADRH	18Fh	rezervat(2)
10h	T1CON	90h		110h	Regiştrii de uz general 16 octeți	190h	Regiştrii de uz general 16 octeți
11h	TMR2	91h	SSPCON2	111h		191h	
12h	T2CON	92h	PR2	112h		192h	
13h	SSPBUF	93h	SSPADD	113h		193h	
14h	SSPCON	94h	SSPSTAT	114h		194h	
15h	CCPR1L	95h		115h		195h	
16h	CCPR1H	96h		116h		196h	
17h	CCP1CON	97h		117h		197h	
18h	RCSTA	98h	TXSTA	118h		198h	
19h	TXREG	99h	SPBRG	119h		199h	
1Ah	RCREG	9Ah		11Ah		19Ah	
1Bh	CCPR2L	9Bh		11Bh		19Bh	
1Ch	CCPR2H	9Ch		11Ch		19Ch	
1Dh	CCP2CON	9Dh		11Dh		19Dh	
1Eh	ADRESH	9Eh	ADRESL	11Eh		19Eh	
1Fh	ADCON0	9Fh	ADCON1	11Fh		19Fh	
20h 7Fh	Regiştrii de uz general 96 octeți	A0h EFh	Regiştrii de uz general 80 octeți	120h 16Fh	Regiştrii de uz general 80 octeți	1A0h 1EFh	Regiştrii de uz general 80 octeți
		F0h FFh	acces 70h-7Fh	170h 17Fh	acces 70h-7Fh	1F0h 1FFh	acces 70h-7Fh

Locație de memorie neimplementată, la citire se citeşte zero.

* Nu este un registru fizic.

Notă: 1: aceste registre nu sunt implementate la PIC16F876.
2: aceste registre sunt rezervate, ele nu trebuie scrise.

Biții de configurare (directiva __CONFIG)

__CONFIG Directive Symbols (From Microchip Header Files)

Feature	SYMBOLS
Oscillators	_RC_OSC
	_EXTRC_OSC
	_EXTRC_OSC_CLKOUT
	_EXTRC_OSC_NOCLKOUT
	_INTRC_OSC
	_INTRC_OSC_CLKOUT
	_INTRC_OSC_NOCLKOUT
	_LP_OSC
	_XT_OSC
	_HS_OSC
Watch Dog Timer	_WDT_ON
	_WDT_OFF
Power-up Timer	_PWRTE_ON
	_PWRTE_OFF
Brown-out Reset	_BODEN_ON
	_BODEN_OFF
Master Clear Enable	_MCLRE_ON
	_MCLRE_OFF
Code Protect	_CP_ALL
	_CP_ON
	_CP_75
	_CP_50
	_CP_OFF
Code Protect Data EEPROM	_DP_ON
	_DP_OFF
Code Protect Calibration Space	_CPC_ON
	_CPC_OFF

Note 1: Not all configuration bit symbols may be available on any one device. Please refer to the Mlcrochip include file of that device for available symbols.

```
;==========================================================================
;
;       Configuration Bits
;
;==========================================================================

_CP_ALL                 EQU     H'1FFF'
_CP_OFF                 EQU     H'3FFF'
_DEBUG_OFF              EQU     H'3FFF'
_DEBUG_ON               EQU     H'37FF'
_WRT_OFF                EQU     H'3FFF'     ; No prog memmory write
protection
```

```
_WRT_256                EQU   H'3DFF'     ; First 256 prog memmory write
protected
_WRT_1FOURTH            EQU   H'3BFF'     ; First quarter prog memmory write
protected
_WRT_HALF               EQU   H'39FF'     ; First half memmory write
protected
_CPD_OFF                EQU   H'3FFF'
_CPD_ON                 EQU   H'3EFF'
_LVP_ON                 EQU   H'3FFF'
_LVP_OFF                EQU   H'3F7F'
_BODEN_ON               EQU   H'3FFF'
_BODEN_OFF              EQU   H'3FBF'
_PWRTE_OFF              EQU   H'3FFF'
_PWRTE_ON               EQU   H'3FF7'
_WDT_ON                 EQU   H'3FFF'
_WDT_OFF                EQU   H'3FFB'
_RC_OSC                 EQU   H'3FFF'
_HS_OSC                 EQU   H'3FFE'
_XT_OSC                 EQU   H'3FFD'
_LP_OSC                 EQU   H'3FFC'
```

Cuvântul de configurare (adresa 2007h)

Valoarea citită pentru biții de configurare neprogramați (șterși) este 3FFFh.

Un bit programat este citit ca zero iar cel neprogramat este citit ca unu. Aceşti biți se găsesc în spațiul de program şi locația de memorie de la adresa 2007h poate fi modificată numai pe timpul programării.

R/P-1	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	U-0	U-0	R/P-1	R/P-1	R/P-1
CP	-	DEBUG	WRT1	WRT0	CPD	LVP	BORDEN	-	-	~~PWRTEN~~	WDTEN	F0SC1	F0SC0
Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

bit 13 CP: bit de protecție a codului din memoria FLASH

1 = protecția codului off
0 = codul din memoria de program este protejat – odată protejată, memoria FLASH nu mai poate fi scrisă.

bit 12 Neimplementat. Citit ca unu.

bit 11 DEBUG: bit pentru depanare In-Circuit

1 = depanator In Circuit dezactivat. RB6 şi RB7 sunt pini de uz general I/O
0 = depanator In Circuit activat. RB6 şi RB7 sunt atribuiți depanatorului

bit 10-9 WRT1:WRT0 biți de validare a scrierii memoriei de program FLASH

11 = nu este activată protecția la scriere; toată memoria de program poate fi scrisă sub controlul EECON
10 = 0000h la 00FFh este protejat la scriere; 0100h la 1FFFh poate fi scrisă sub controlul EECON
01 = 0000h la 07FFh este protejat la scriere; 0800h la 1FFFh poate fi scrisă sub controlul EECON
00 = 0000h la 0FFFh este protejat la scriere; 1000h la 1FFFh poate fi scrisă sub controlul EECON

bit 8 CPD: bit de protecție a memoriei de date EEPROM

1 = memoria EEPROM nu este protejata
0 = memoria EEPROM este protejată

bit 7 LVP: bit de validare a tensiunii scăzute la programarea serială in-circuit

1 = pinii RB3/PGM au funcția PGM; programarea cu tensiune scăzută este validată;
0 = RB3 este I/O digitală, trebuie utilizată HV la ~~MCLR~~ pentru programare

bit 6 BORDEN: bit de validare a Brown-out Reset
1 = BOR activat
0 = BOR dezactivat

bit 5-4 Neimplementat. Citit ca unu.

bit 3 ~~PWRTEN~~ bit de validare a Power-up Timer

1 = PWRT dezactivat
0 = PWRT activat

bit 2 WDTEN: bit de validare a Watchdog Timer

1 = WDT activat
0 = WDT dezactivat

bit 1-0 FOSC1:FOSC0: biții de selecție a oscilatorului

11 = oscilator RC
10 = oscilator HS
01 = oscilator XT

00 = oscilator LP

Regiştrii cu funcţii speciale

Adr.	Nume	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valoarea în: POR, BOR
Bank 0										
0h(3)	INDF	Adresând această locaţie de memorie se foloseşte conţinutul FSR pentru a adresa memoria de date (nu este un registru fizic)								0000 0000
01h	TMR0	Registrul Timer0								xxxx xxxx
02h(3)	PCL	Contorul de program (PC) cel mai puţin semnificativ octet								0000 0000
03h(3)	STATUS	IRP	RP1	RP0	~~TO~~	~~PD~~	Z	DC	C	0001 1xxx
04h(3)	FSR	Pointer pentru adresarea indirectă a memoriei de date								xxxx xxxx
05h	PORTA	-	-	Memorare date PORTA la scriere, pini PORTA la citire						--0x 0000
06h	PORTB	Memorare date PORTB la scriere, pini PORTB la citire								xxxx xxxx
07h	PORTC	Memorare date PORTC la scriere, pini PORTC la citire								xxxx xxxx
08h(4)	PORTD	Memorare date PORTD la scriere, pini PORTD la citire								xxxx xxxx
09h(4)	PORTE	-	-	-	-	-	RE2	RE1	RE0	---- -xxx
0Ah(1,3)	PCLATH	-	-	-	Buffer de scriere a celor mai semnificativi 5 biţi ai contorului de program (PC)					---0 0000
0Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x
0Ch	PIR1	PSPIF(3)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000
0Dh	PIR2	-	CMIF	-	EEIF	BCLIF	-	-	CCP2IE	-0-0 0—0
0Eh	TMR1L	Registrul de stocare al celui mai puţin semnificativ octet al registrului de 16 biţi al TMR1								xxxx xxxx
0Fh	TMR1H	Registrul de stocare al celui mai semnificativ octet al registrului de 16 biţi al TMR1								xxxx xxxx
10h	T1CON	-	-	T1CKPS1	T1CKPS0	T1OSCEN	~~T1SYNC~~	TMR1CS	TMR1ON	--00 0000
11h	TMR2	Registrul Timer2								1111 1111
12h	T2CON	-	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS1	TMR2ON	T2CKPS1	T2CKPS0	-000 0000
13h	SSPBUF	Registrul de transmisie/buffer de recepţie a portului serial sincron								xxxx xxxx
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000
15h	CCPR1L	Registrul 1 (LSB) Captură/Comparare/PWM								0000 0000
16h	CCPR1H	Registrul 1 (MSB) Captură/Comparare/PWM								-
17h	CCP1CON	-	-	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M2	CCP1M0	--00 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x
19h	TXREG	Registrul datelor de transmis al USART								0000 0000
1Ah	RCREG	Registrul datelor recepţionate al USART								0000 0000
1Bh	CCPR2L	Registrul 2 (LSB) Captură/Comparare/PWM								xxxx xxxx
1Ch	CCPR2H	Registrul 2 (MSB) Captură/Comparare/PWM								xxxx xxxx
1Dh	CCP2CON	-	-	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000
1Eh	ADRESH	Registrul rezultat al octetului cel mai semnificativ al conversiei A/D								xxxx xxxx
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/~~DONE~~	-	ADON	0000 00-0

Legendă: x = necunoscut, u = neschimbat, q = valoarea depinde de situaţie, - = neimplementat – se citeşte zero, r = rezervat

Notă: 1. Cel mai semnificativ octet al contorului de program nu este accesibil direct. Registrul PCLATH conține biții PC<12:8> care vor fi transferați în octetul cel mai semnificativ al contorului de program.
2. Biții PSPIE şi PSPIF sunt rezervați la PIC16F873A/876A şi trebuie menținuți şterşi.
3. Aceşti regiştrii pot fi adresați din orice bank.
4. PORTD, PORTE, TRISD şi TRISE nu sunt implementați la PIC16F873A/876A şi la citire sunt zero.
5. Bitul 4 al EEADRH este implementet numai la PIC16F876A/877A.

Adr.	Nume	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valoarea în: POR, BOR
Bank 1										
80h(3)	INDF	Adresând această locație de memorie se folosește conținutul FSR pentru a adresa memoria de date (nu este un registru fizic)								0000 0000
81h	OPTION_REG	~~RBPU~~	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111
82h(3)	PCL	Contorul de program (PC) cel mai puțin semnificativ octet								0000 0000
83h(3)	STATUS	IRP	RP1	RP0	~~TO~~	~~PD~~	Z	DC	C	0001 1xxx
84h(3)	FSR	Pointer pentru adresarea indirectă a memoriei de date								xxxx xxxx
85h	TRISA	-	-	Registru direcție PORTA						--11 1111
86h	TRISB	Registru direcție PORTB								1111 1111
87h	TRISC	Registru direcție PORTC								1111 1111
88h(4)	TRISD	Registru direcție PORTD								1111 1111
89h(4)	TRISE	IBF	OBF	IBOV	PSPMODE	-	Biți direcție PORTE			0000 -111
8Ah(1,3)	PCLATH	-	-	-	Buffer de scriere a celor mai semnificativi 5 biți ai contorului de program (PC)					---0 0000
8Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x
8Ch	PIE1	PSPIE(2)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000
8Dh	PIE2	-	CMIE	-	EEIE	BCLIE	-	-	CCP2IE	-0-0 0—0
8Eh	PCON	-	-	-	-	-	-	~~POR~~	~~BOR~~	---- --qq
8Fh	-	Neimplementat								-
90h	-	Neimplementat								-
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000
92h	PR2	Registru de perioada al Timer2								1111 1111
93h	SSPADD	Registrul de adresă al portului serial sincron (mod I^2C)								0000 0000
94h	SSPSTAT	SMP	CKE	D/~~A~~	P	S	R/~~W~~	UA	BF	0000 0000
95h	-	Neimplementat								-
96h	-	Neimplementat								-
97h	-	Neimplementat								-
98h	TXSTA	CSRC	TX9	TXEN	SZNC	-	BRGH	TRMT	TX9D	0000 -010
99h	SPBRG	Registrul generator Baud Rate								0000 0000
9Ah	-	Neimplementat								-
9Bh	-	Neimplementat								-
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111
9Dh	CVRCON	CVREN	CVROE	CVRR	-	CVR3	CVR2	CVR1	CVR0	000- 0000
9Eh	ADRESL	Registrul octetului cel mai puțin semnificativ al conversiei A/D								xxxx xxxx
9Fh	ADCON1	ADFM	ADCS2	-	-	PCFG3	PCFG2	PCFG1	PCHG1	0--- 0000

Legendă: x = necunoscut, u = neschimbat, q = valoarea depinde de situație, - =

neimplementat – se citeşte zero, r = rezervat

Notă: 1. Cel mai semnificativ octet al contorului de program nu este accesibil direct. Registrul PCLATH conține biții PC<12:8> care vor fi transferați în octetul cel mai semnificativ al contorului de program.
2. Biții PSPIE şi PSPIF sunt rezervați la PIC16F873A/876A şi trebuie menținuți şterşi.
3. Aceşti regiştrii pot fi adresați din orice bank.
4. PORTD, PORTE, TRISD şi TRISE nu sunt implementați la PIC16F873A/876A şi la citire sunt zero.
5. Bitul 4 al EEADRH este implementet numai la PIC16F876A/877A.

Adr.	Nume	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valoarea în: POR, BOR
Bank 2										
100h(3)	INDF	Adresând această locație de memorie se folosește conținutul FSR pentru a adresa memoria de date (nu este un registru fizic)								0000 0000
101h	TMR0	Registrul Timer0								xxxx xxxx
102h(3)	PCL	Contorul de program (PC) cel mai puțin semnificativ octet								0000 0000
103h(3)	STATUS	IRP	RP1	RP0	~~TO~~	~~PD~~	Z	DC	C	0001 1xxx
104h(3)	FSR	Pointer pentru adresarea indirectă a memoriei de date								xxxx xxxx
105h	-	Neimplementat								
106h	PORTB	Memorare date PORTB la scriere, pini PORTB la citire								xxxx xxxx
107h	-	Neimplementat								-
108h	-	Neimplementat								-
109h	-	Neimplementat								-
10Ah(1,3)	PCLATH	-	-	-	Buffer de scriere a celor mai semnificativi 5 biți ai contorului de program (PC)					---0 0000
10Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x
10Ch	EEDATA	Octetul cel mai puțin semnificativ (L) al registrului de date al EEPROM								xxxx xxxx
10Dh	EEADR	Octetul cel mai puțin semnificativ (L) al registrului de adresă al EEPROM								xxxx xxxx
10Eh	EEDATH	-	-	Octetul cel mai semnificativ (H) al registrului de date al EEPROM						--xx xxxx
10Fh	EEADRH	-	-	-	-(5)	Octetul cel mai semnificativ (H) al registrului de adresă al EEPROM				---- xxxx
Bank 3										
180h(3)	INDF	Adresând această locație de memorie se folosește conținutul FSR pentru a adresa memoria de date (nu este un registru fizic)								0000 0000
181h	OPTION_REG	~~RBPU~~	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111
182h(3)	PCL	Contorul de program (PC) cel mai puțin semnificativ octet								0000 0000
183h(3)	STATUS	IRP	RP1	RP0	~~TO~~	~~PD~~	Z	DC	C	0001 1xxx
184h(3)	FSR	Pointer pentru adresarea indirectă a memoriei de date								xxxx xxxx
185h	-	Neimplementat								
186h	TRISB	Registru de direcție a datelor PORTB								xxxx xxxx
187h	-	Neimplementat								-
188h	-	Neimplementat								-
189h	-	Neimplementat								-
18Ah(1,3)	PCLATH	-	-	-	Buffer de scriere a celor mai semnificativi 5 biți ai contorului de program (PC)					---0 0000
18Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x
18Ch	EECON1	EEPGD	-	-	-	WRERR	WREN	WR	RD	x--- x000

18Dh	EECON2	Registrul de control 2 (nu este un registru fizic) EEPROM	---- ----
18Eh	-	Rezervat, trebuie menţinut şters	0000 0000
18Fh	-	Rezervat, trebuie menţinut şters	0000 0000

Legendă: x = necunoscut, u = neschimbat, q = valoarea depinde de situaţie, - = neimplementat – se citeşte zero, r = rezervat

Notă:
1. Cel mai semnificativ octet al contorului de program nu este accesibil direct. Registrul PCLATH conţine biţii PC<12:8> care vor fi transferaţi în octetul cel mai semnificativ al contorului de program.
2. Biţii PSPIE şi PSPIF sunt rezervaţi la PIC16F873A/876A şi trebuie menţinuţi şterşi.
3. Aceşti regiştrii pot fi adresaţi din orice bank.
4. PORTD, PORTE, TRISD şi TRISE nu sunt implementaţi la PIC16F873A/876A şi la citire sunt zero.
5. Bitul 4 al EEADRH este implementet numai la PIC16F876A/877A.

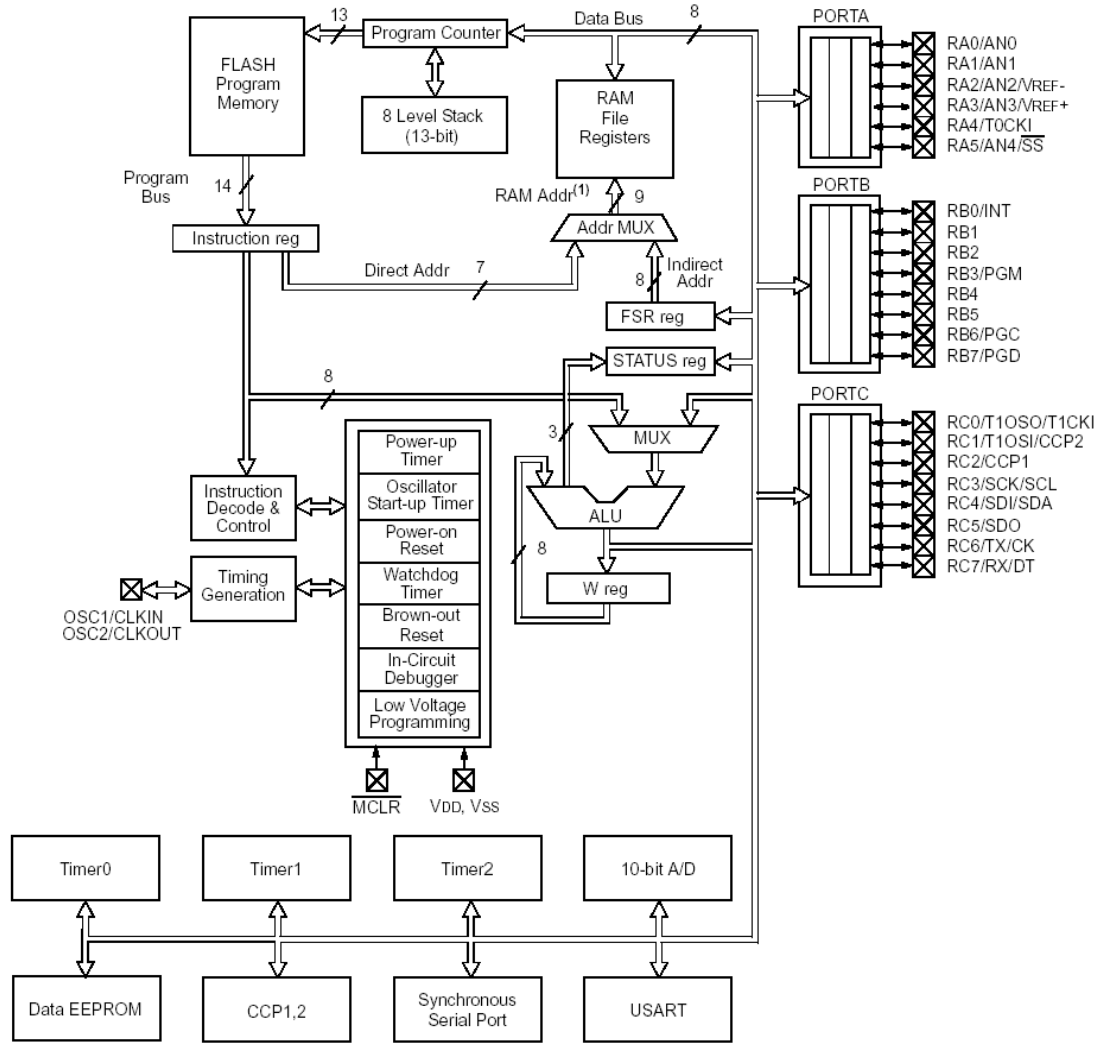


Figura 4.16. Schema bloc a microcontrolerului PIC16F877.

CAPITOLUL 5

MEMORIA

5.1. Prezentare generală

Memoria unui calculator are rolul de a stoca temporar sau permanent date sau programe. Durata de stocare a datelor se referă la timpul cât memoria este alimentată la tensiunea electrică. O memorie poate stoca permanent date dacă informația memorată nu se pierde la întreruperea tensiunii de alimentare a memoriei. În caz contrar, atunci când memoria stochează informația numai pe durata de timp cât este alimentată cu energie electrică, se spune că informația este memorată temporar. Este momentul sa precizăm faptul că un calculator numeric nu prelucrează decât numere. Din acest motiv împărțirea informației în date şi instrucțiuni are caracter pur convențional. Astfel, într-o locație de memorie se poate afla numărul ($10110110_B = 182_Z$) care pentru unitatea centrală poate însemna fie o valoare numerică egală cu 128 fie codul unei instrucțiuni de tipul: 'aduna numarul a cu numarul b'. Unitatea centrală decide dacă numărul citit din memorie reprezintă o valoare numerică (dată) sau o comandă (instrucțiune program). Prin convenție, la pornire sau după inițializare (RESET) unitatea centrală consideră că valoarea citită din memorie reprezintă o comandă. Să presupunem ca acesta comanda este: 'aduna numarul a cu numarul b'. Asta înseamna ca urmatoarele două valoari citite din memorie se vor considera ca reprezentând valoarile numerelor a şi b. După executarea adunării unitatea centrală va citi o nouă valoare din memorie pe care o va considera în mod automat ca reprezentând codul numeric al unei instrucțiuni de program. Rămâne deci responsabilitatea programatorului ca în memorie să fie înscrisă o succesiune corectă de date care reprezintă numere sau coduri de instrucțiuni.

După modul în care sunt stocate datele în memorie se poate face o primă clasificare a acestora:

- memorii de tip ROM (Read Only Memory) care sunt memorii ce pot stoca permanent datele. Aşa cum le arată şi numele ele nu pot fi decât citite de către unitatea centrală, înscrierea lor fiind făcută prin procedee speciale, fie pe calculator, fie separat, pe un dispozitiv special numit inscriptor de memorii ROM;
- memorii de tip RAM (Random Access Memory) care sunt memorii ce pot stoca temporar datele. Caracteristic acestor memorii este faptul că ele pot fi scrise şi citite în mod curent de unitatea centrală.

Cele două tipuri de memorie sunt amândouă folosite în calculator în scopuri diferite. Memoria de tip ROM stochează de obicei programul executat de unitatea

centrală imediat după pornire sau la inițializare. Acest program se mumeşte program ***monitor*** sau program de ***boot***. Dacă în calculator s-ar folosi numai memorie de tip RAM problema pornirii calculatorului nu ar putea fi rezolvată. La punerea sub tensiune o memorie RAM are un conținut aleator care nu poate ajuta la pornirea calculatorului; cum unitatea centrală, după inițializare aşteaptă o comandă validă, rezultă că trebuie folosită o memorie care să nu-şi piardă conținutul atunci când tensiunea de alimentare este întreruptă.

Structura unei memorii este similară cu cea a unei matrici. Vom considera un exemplu în care în fiecare element al matricii se află un modul elementar de memorie care nu poate memora decât un bit adică o valoare numerică egală cu zero sau cu unu. Cea mai simplă structură de memorie este prezentată în figura 5.1. Localizarea celulei elementare de memorie care va fi citită sau scrisă la un moment dat se face cu ajutorul semnalelor de adresă iar valoarea citită sau valoarea de înscris în memorie este disponibilă pe respectiv este depusă pe magistrala de date. În acest fel se spune că selecția unei celule de memorie se face cu ajutorul semnalelor de pe magistrala de adrese iar valoarea citită din memorie sau înscrisă în memorie se face cu ajutorul semnalelor de pe magistrala de date.

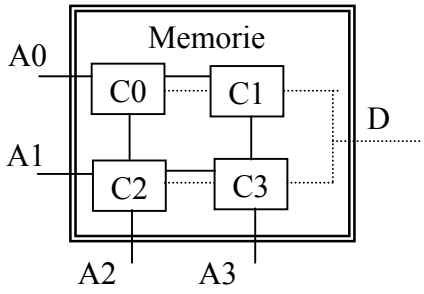


Fig. 5.1. Structura unei memorii

În figura 5.1, C0, C1, C2 şi C3 reprezintă celulele elementare de memorie, D magistrala de date (o singură linie) iar A0, A1, A2 şi A3 magistrala de adrese (patru linii). Traseul datelor a fost reprezentat cu linie punctată iar cel al adreselor cu linie continuă. Din cauză că la un moment dat numai una din celulele elementare de memorie este activată, pentru date este suficientă o singură linie. Considerăm că o celulă elementară de memorie este activată pentru citire sau scriere dacă ea se află la intersecția liniilor de adresă pe care se află simultan cifra unu. Funcționarea memoriei, în acest caz este dată în tabelul 5.1. Dacă pe magistrala de adrese se aplică, de exemplu, numărul 0101 (în succesiunea A3,A2,A1,A0) atunci pe magistrala de date se găseşte conținutul celulei C0 la citire sau poate fi înscrisă celula C0 cu valoarea plasată pe magistrala de date, la scriere.

TABELUL 5.1.

A3	A2	A1	A0	D
0	1	0	1	Conținutul celulei C0
1	0	0	1	Conținutul celulei C1
0	1	1	0	Conținutul celulei C2
1	0	1	0	Conținutul celulei C3

Din cauză că folosirea a patru linii de adresă este neeconomică în acest caz, în realitate la o astfel de memorie nu se află decât două linii de adresă din cauză că se pot obține patru combinații distincte numai cu două numere binare (00, 01, 10, 11). Translatarea de la cele patru combinații binare la cele patru adrese de selacție a celulelor de memorie se face în interiorul memoriei cu ajutorul unui circuit de decodificare.

Pentru a obține mai multe date simultan la ieşire memoriile se conectează în paralel aşa cum este reprezentat schematic în figura 5.2.

În structura din figura 5.2 dacă se aplică la intrare (pe liniile de adresă) combinația 0101 (în succesiunea A3,A2,A1,A0), atunci vor fi selectate simultan celulele

de memorie C00 şi C01, C00 fiind conectată la linia de date D0 iar C01 la D1. Toate celulele elementare de memorie selectate simultan, formează o celulă de memorie care permite manipularea simultana a mai multor biţi.

Din cele prezentate până acum rezultă că atât magistrala de adrese cât şi cea de date furnizează informaţii legate de dimensiunea (capacitatea memoriei).

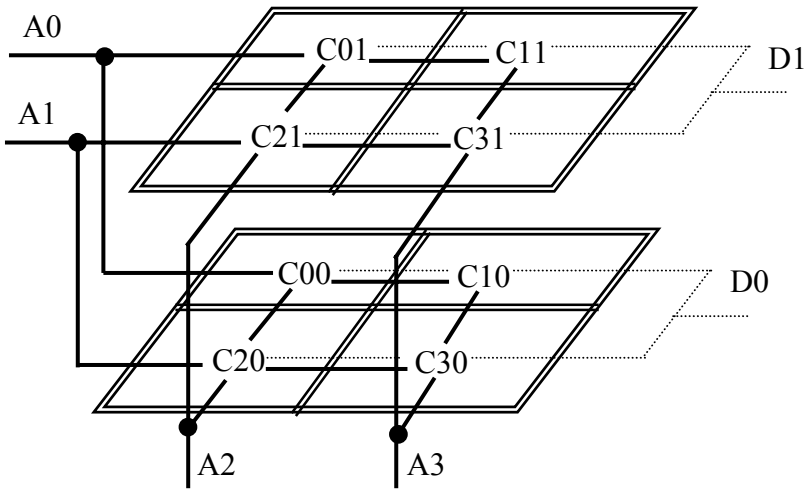


Fig.5.2. Structura unei memorii cu două planuri

Pentru a măsura capacitatea (dimensiunea) memoriei se foloseşte ca unitate de bază ***bitul*** (b) care reprezintă cantitatea elementară de informaţie. Spre exemplu, vom considera experimentul aruncării unei monezi. În urma acestui experiment înlăturăm o incertitudine de 50% pentru că înainte de aruncarea monezii existau numai două posibilităţi ale rezultatului (una din feţele monezii). Astfel, probabilitatea de obţinere a unui rezultat este de 1/2 din cauză că din cele două rezultate posibile s-a obţinut unul. Cantitatea de informaţie pe care o primim după efectuarea acestui experiment se exprimă cu relaţia:

$$I = - \log_2 P = - \log_2 ½ = 1 \text{ bit}$$

Dacă asociem celor două feţe ale monedei valorile binare 0 şi 1, cantitatea de informaţie stocată într-o celulă elementară de memorie (care poate memora una din valorile 1 sau 0) este de un bit. Bitul are ca multiplii ***kilobitul*** (kb), ***megabitul*** (Mb), ***gigabitul*** (Gb) şi ***terabitul*** (Tb) cu următoarele relaţii între ele:

- 1kb = 1024 biţi;
- 1Mb = 1024kb = 1 048 576 biţi;
- 1Gb = 1024Mb = 1 048 576 kb = 1 073 741 824 biţi;
- 1Tb = 1024Gb = 1 048 576 Mb = 1 073 741 824 kb = 1 099 511 627 776 biţi.

Pentru a ne forma o imagine asupra cantităţii de informaţie care poate fi stocată, putem spune că dacă un bit stochează informaţia corespunzătoare unui eveniment echiprobabil cu două posibilităţi de realizare aşa cum este experimentul aruncării unei monezi, într-un gibabit se poate stoca informaţia dintr-o enciclopedie iar într-un terabit informaţia dintr-o bibliotecă de 500 de cărţi.

Întorcându-ne la figura 5.2 putem spune că această memorie are o capacitate de 4x1bit din cauză că sunt patru celule elementare de memorie care pot stoca informaţie,

la un moment dat având acces la o singură celulă. Similar, memoria din figura 5.2 are o capacitate de 4x2 biți din cauză că, la un moment dat, avem acces la două celule elementare de memorie simultan.

Aşa cum s-a arătat în capitolul 4, unitățile centrale pot lucra cu 8, 16, 32, 64, ... biți de date simultan. Din acest motiv un grup de opt biți se numeşte ***octet*** (byte, 1 byte=8 bits). Daca se utilizează 16 biți simultan adică doi octeți, aceştia formează un ***cuvânt*** (word) iar două cuvinte formează un ***pointer***. În tabelul 5.2 sunt sintetizate modurile de notare a grupurilor de biți.

TABELUL 5.2.

1 octet (byte) = 8 biți (bites)
1 cuvânt (word) = 2 octeți (bytes) = 16 biți (bites)
1 pointer (poiner) = 2 cuvinte (words) = 4 octeți (bytes) = 32 biți (bits)

Memoriile calculatoarelor se construiesc în general cu celule de opt biți de date iar capacitățile acestora sunt exprimate în kiloocteți (kilobytes), notat cu ko sau kb, un ko fiind egal cu 1024 octeți. Capacitățile uzuale folosite pentru memoriile calculatoarelor numerice variază de la 1ko şi pot ajunge până la zeci sau sute de Mo (megaocteți).

Calculul capacității unei memorii poate fi făcută în felul următor: sa presupunem că avem o memorie de 5Mo; rezultă că această memorie va avea:

5 x 1 048 576 x 8 biți = 41 943 040 biți

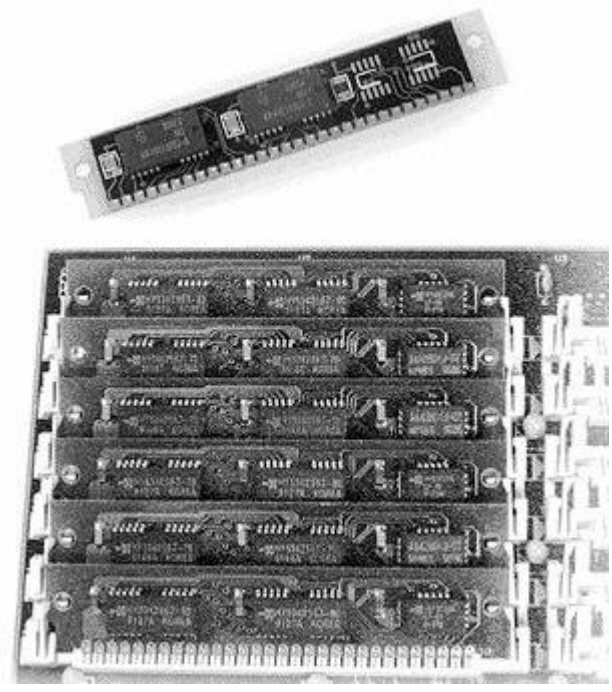


Fig. 5.3. Memorie SIMM şi modul de montare a acesteia

ceea ce înseamnă că o astfel de memorie va conține 41 943 040 celule elementare de memorie. În figura 5.3 se prezintă o memorie de 4 Mo şi modul de montare a acesteia.

În afară de capacitatea memoriei, o altă caracteristică importantă a acesteia este ***timpul de acces.*** Timpul de acces reprezintă timpul necesar memoriei de a răspunde unei comenzi. Cu alte cuvinte, la o comandă de citire, memoria nu depune instantaneu conținutul celulei de memorie (o celulă de memorie fiind formată din una sau mai multe celule elementare) pe magistrala de date ci după un anumit timp care depinde atât de tehnologia în care este realizată memoria cât şi de dimensiunea acesteia. La fel, la scrierea unei memorii, datele de înscris în memorie trebuie menținute

un anumit timp pe magistrala de date, pentru ca memoria să fie capabilă să le transfere în celula de memorare corespunzătoare. Este evident faptul că este de dorit ca timpul de acces al unei memorii să fie cât mai mic în aşa fel încât ea să poată răspunde suficient de repede solicitărilor unității centrale. În prezent memoriile utilizate în calculatoare au timpi de acces între 70ns şi 10ns. Din păcate cu cât timpul de acces al unei memorii este mai mic prețul acesteia este mai mare şi din acest motiv, în cele mai multe cazuri, memoria unui sistem este realizată dintr-o combinație de memorii rapide şi memorii lente. Astfel, memoria RAM într-un calculator este realizată pe cel puțin trei nivele. Pe primul nivel este memoria cea mai rapidă, care se află chiar în interiorul unității ceantrale. Această memorie se numeste memorie ***cache*** pe nivelul 1 (cache level 1). În această memorie se aduc porțiunile de program care sunt rulate la un moment dat, unitatea centrală lucrând practic numai cu memoria cache de pe nivelul 1. Atunci când este necesară execuția unei instrucțiuni care nu este în memoria cache de pe nivelul 1, un circuit de control al memoriei aduce porțiunea respectivă de program, din memoria externă mai lentă, în memoria cache de pe nivelul 1. Microprocesoarele actuale au o memorie cache internă între 128ko şi 512ko, iar la microprocesoarele foarte puternice destinate stațiilor de lucru, memoria cache poate ajunge până la 2Mo. În exteriorul unității centrale există un al doilea nivel al memoriei cache (cache level 2) cu acelaşi rol ca şi memoria cache level 1. Mecanismul memoriei cache este prezentat 1n figura 5.4. Controlerul de memorie are sarcina de a transfera datele de pe un nivel pe altul în aşa fel încât în memoria cache de pe nivelul 1 să se afle întotdeauna porțiunea de program necesar unității centrale.

Performanțele unui calculator vor fi cu atât mai bune cu cât dimensiunea memoriei cache este mai mare.

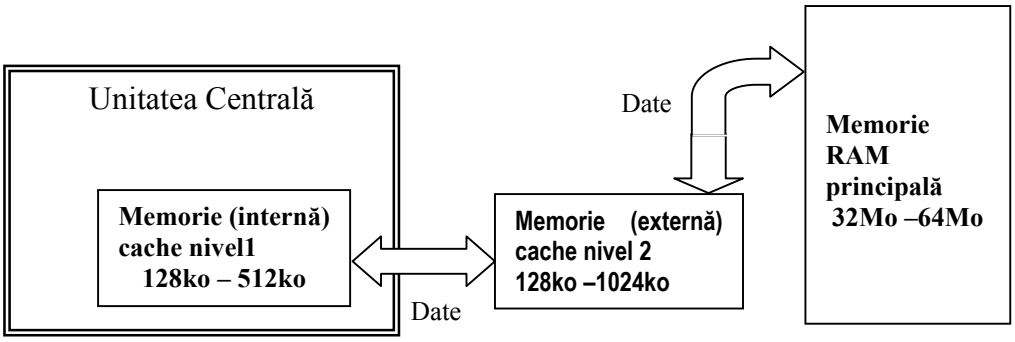


Fig. 5.4. Organizarea memoriei RAM la un calculator

În continuare se vor prezenta pe scurt principalele tipuri tehnologice de memorii ROM şi RAM existente în prezent şi caracteristicile acestora.

Memoriile ROM sunt realizate în următoarele variante:

- memorii ROM propriu-zise care nu pot fi înscrise decât o singură dată, înscrierea fiind făcută cel mai adesea la fabricant sau la utilizator cu ajutorul unor dispozitive de programare speciale, aceste memorii fiind denumite şi memorii PROM (Programmable Read-Only Memory);

- memorii EPROM (Erasable Programmable Read-Only Memory) care sunt memorii de tip ROM programabile. Ele pot fi sterse si înscrise (programate) de mai multe ori. Stergerea memoriei se face cu ajutorul unui flux de lumină ultravioletă, în acest scop capsula circuitului integrat fiind prevăzută cu un geam din sticlă de cuarț (vezi figura 4.1), iar înscrierea se face cu ajutorul unor dispozitive speciale prin metode electrice la tensiuni mai mari decât cele de funcționare normală;
- memorii de tip EEPROM (Electrically Erasable Programmable Read-Only Memory) care sunt memorii ROM programabile ce pot fi şterse şi reînscrise exclusiv prin mijloace electrice. Avantajul acestui tip de memorii este reprezentat de faptul că ele nu trebuie scoase din circuitul în care au fost montate pentru a fi şterse şi reînscrise. Acest tip de memorie necesită tensiuni mai mari decât cele de funcționare obişnuită, pentru ştergere şi programare;
- memorii de tip ***flash*** (flash memory), ultima generație de memorii de tip ROM care deşi păstrează informația şi după întreruperea tensiunii de alimentare, pot fi sterse si programate similar cu memoriile RAM. Singura particularitate este reprezentată de faptul că memoriile flash necesită algoritmi speciali de ştergere şi programare, această operațiune fiind făcută pe blocuri, în trei faze.

Memoriile RAM sunt realizate în următoarele variante:

- memorii RAM ***dinamice*** (DRAM - Dynamic Random Access Memory) care sunt memorii de tip RAM ce necesită reîmprospătarea periodică (la 20 ms) a conținutului, în caz contrar informația pierzându-se chiar dacă sunt alimentate. Reîmprospătarea se face simplu prin citirea periodică a mai multor blocuri simultan din memorie. Principalul avantaj al acestui tip de memorii este reprezentat de faptul că pot avea d densitate foarte mare a celulelor elementare, putând fi realizate memorii cu capacități foarte mari iar principalul dezavantaj este reprezentat de faptul că memoriile RAM dinamice sunt elativ lente (necesită timpi de acces mari). Din acest motiv memoriile RAM dinamice sunt folosite de regulă la memoria principală a calculatorului.
- memorii RAM ***statice*** (SRAM - Static Random Access Memory) care sunt memorii de tip RAM ce nu necesită reîmprospătare. Aceste memorii prezintă avantajul că au timpi de acces foarte mici în schimb au dezavantajul că nu pot fi realizate de capacități foarte mari. Memoriile RAM statice sunt utilizate de regulă la realizarea memoriilor cache.

Pentru cresterea performanțelor de viteză ale memoriilor RAM, în prezent sunt utilizate o serie de tehnici care au dus la apariția mai multor tipuri de memorii RAM dinamice:

- memorii de tip EDO DRAM (Extended Data Output Dynamic Random Access Memory) care sunt memorii de tip DRAM dar mai rapide decât memoriile DRAM obişnuite. Spre deosebire de memoriile DRAM obişnuite, o memorie EDO DRAM nu poate fi accesată decât pe blocuri de date. Pe durata de timp cât memoria EDO RAM trimite la unitatea centrală datele corespunzătoare unui bloc, poate căuta datele blocului următor. Datorită acestor mecanisme timpul de regăsire a datelor este mult redus;
- memorii de tip BEDO DRAM (Burst EDO DRAM). Memoria BEDO DRAM este mai rapidă decât memoria EDO DRAM. Acest tip de memorie se sincronizează cu viteza unității centrale pe durate scurte de timp (***burst***). Pe durata unei astfel de sincronizări o memorie BEDO DRAM poate prelucra patru adrese de memorie simultan. Aceste memorii se pot conecta la procesoare ce lucrează pe magistrale cu viteza de până la 66 MHz;
- memorii de tip SDRAM (Synchronous DRAM). Acest tip de memorii sunt capabile să se sincronizeze cu unitatea centrală şi funcționează la frecvențe de până la 200MHz (memorii **DDR SDRAM – double-data-rate synchronous dynamic random access memory)**.

În figura 5.5 este prezentat modul de realizare a circuitelor de memorie.

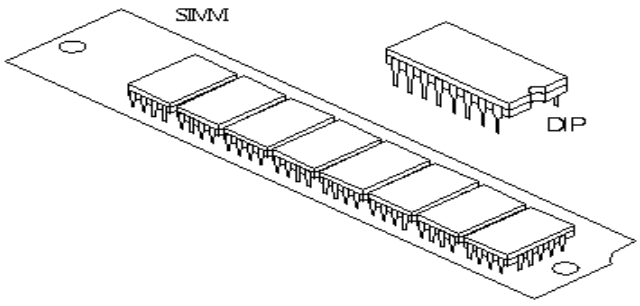


Fig. 5.5. Realizarea fizică a circuitelor de memorie

5.2. Aplicarea principiului "cache" în sistemele de calcul

5.2.1. Memoria Cache

Numele acestui tip de memorie provine de la termenul din limba franceză "ascuns": cache. Deşi este un cvânt de origine franceză s-a încetățenit pronunția acesuia

în limba engleză deoarece majoritatea termenilor din domeniul arhitecturii calculatoarelor provin din această limbă.

Memoria cache reprezintă mai degrabă un principiu: în sistemul de calcul se vor folosi două tipuri de memorie principală – o memorie lentă de dimensiuni mari şi o memorie rapidă de dimensiuni mai mici. Acest lucru se întâmplă din cauză că memoria rapidă este mult mai scumpă decât cea lentă şi atunci mecanismul memoriei cache se foloseşte în scopul cresterii performantelor sistemului în condiţii de preţ acceptabile.

Functionarea memoriei cache se bazează pe aducerea anumitor secvenţe de program, secvenţe ce sunt necesare la un moment dat, din memoria lentă în memoria rapidă (memoria cache) – figura 5.6.

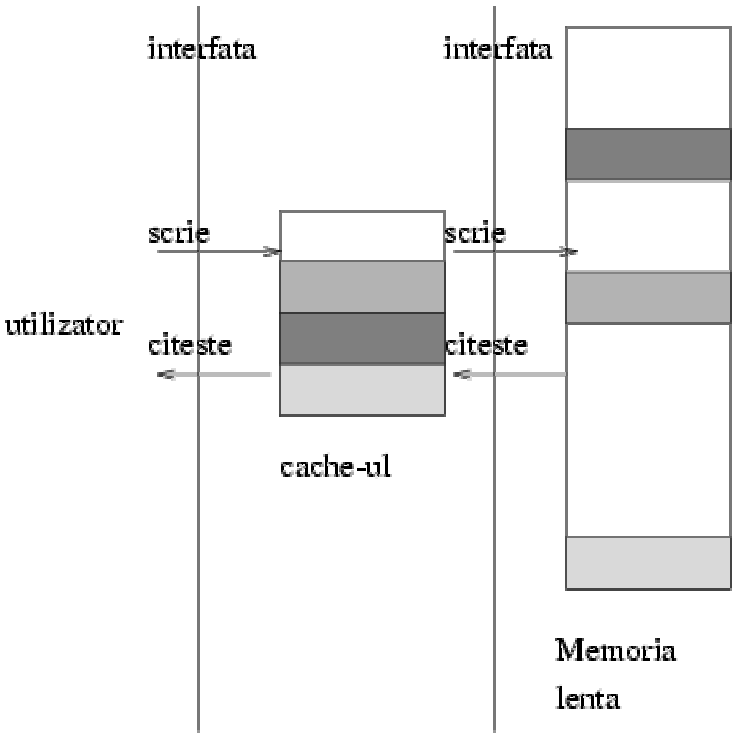


Figura 5.6. Principiul cache-ului

Funcţionarea acestei memrii se bazează pe principiul "localităţii". Există două feluri de localizarea a informaţiei: localizarea spaţială şi localizarea temporală:

- localitate *spaţială*: dacă la un moment dat sunt necesare nişte date, în curînd vor fi necesare probabil de date aflate în apropierea lor în memorie;
- localitatea *temporală*: cînd sunt folosite nişte date, foarte adesea acestea se vor folosi de mai multe ori.

Memoria cache foloseşte acelaşi fel de interfaţă ca şi memoria principală a sistemului. Acest lucru înseamnă că nu ne vom da seama de prezenţa sau lipsa memoriei cache decât prin viteza de calcul a sistemului.

Datorită faptului că eficienţa memoriei cache depinde în bună măsură de programul executat, pentru determinarea performanţelor memoriei cache se fac mai multe determinări cu ajutorul diferitelor tipuri de programe şi apoi se realizează o medie.

Eficienţa unui cache se măsoară în procentajul de situaţii când datele căutate se găsesc în cache H (*hit ratio*): din 100 de accese, cîte găsesc datele în cache? Opusul acestei valori este *miss ratio* M (rateuri). Procentajele astea se măsoară rulînd o mulţime de programe şi făcînd media. Avem desigur, dacă socotim 33% ca 1/3:

$$H = 1 - M$$

Dacă timpul de citire din cache este Tc (``hit time''), iar timpul pe care îl pierdem cînd ratăm este Tm (``miss time'') atunci putem măsura timpul mediu de acces la memoria cu cache cu următoarea formulă:

$$T = Tc \times H + Tm \times M$$

Trebuie observat faptul că timpul unei ratări (Tm) nu este neapărat egal cu timpul de citire din memoria lentă (Tl), pentru că în cazul unei ratări, întîi trebuie să ne dăm seama dacă datele sunt în cache, iar dacă nu sunt să accesăm memoria lentă. Cache-ul va fi eficient dacă T < Tl.

Valoarea mărimii H depinde de mărimea cache-ului: pentru un cache de mărimea memoriei lente (caz limită), toate datele pot fi ţinute în memoria rapidă, şi vom avea H=1. Pentru un cache de mărime 0, H=0, pentru că niciodată datele nu se găsesc în el. Relaţia între mărimea cache-ului, a memoriei lente şi H nu este o linie dreaptă, ci creşte repede la început (figura 5.7). Din cauza asta un cache relativ mic ca mărime are o importanţă mare ca eficienţă.

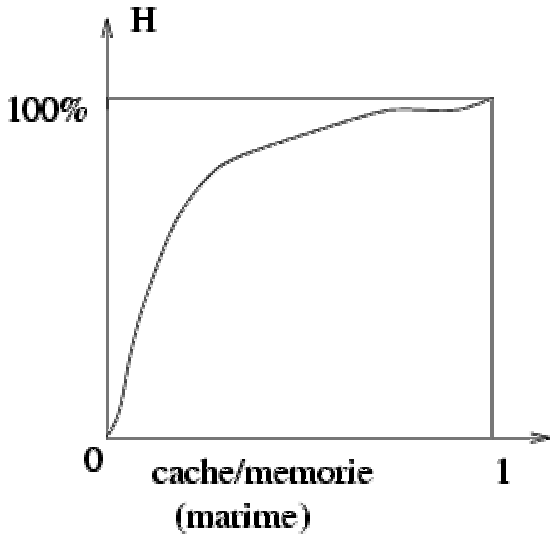


Figura 5.7. Performanţa cache-ului

Eficiența depinde şi de raportul dintre Tc şi Tm; în anumite cazuri Tm este de ordinul a 10000 x Tc, deci chiar un H mic poate să însemne mult.

Pe lîngă dimensiuni şi timpi de acces, există o mulțime de detalii prin care cache-urile diferă între ele, datorate faptului că mediile de stocare a informației nu se comportă chiar la fel. Iată unele dintre posibilele diferențe:

1) Mărimea blocului de date: cîteodată este mai economic să se transporte mai multe date deodată din memoria lentă; cache-urile aduc atunci mai mult decît li se cere (un calup) şi păstrează totul, în ideea (sugerată de principiul de localitate spațială) că şi vecinii obiectului accesat vor fi căutați în curînd. Unitatea de transfer între cache şi memoria lentă e numită *bloc*.
2) Politica de înlocuire: după o vreme cache-ul se va umple cu date, şi totuşi altele vor trebui aduse. Decizia despre care date trebuie scoase afară este foarte importantă pentru eficiență; ea este *politica de înlocuire* (replacement policy). Există o sumedenie de politici; iată-le doar numite pe unele; orice carte de sisteme de operare va descrie cele mai multe dintre ele: politica aleatoare (random), politica circulară (round robin), politica ``cel mai rar folosit'' (least frequently used), politica ``primul intrat - primul ieşit'' (first in, first out), politica ``cel mai demult folosit'' (least recently used), politica ``setul de lucru'' (working set), politica ``optimă'' (optimal), politica ceasului, politica celei de-a doua şanse (second-chance).
3) Politica de scriere: odată cu prezența unui cache, datele efectiv devin duplicate: există o copie în cache. Cînd se fac scrieri, care dintre copii trebuie modificată? Una sau amîndouă? În funcție de circumstanțe există varii răspunsuri la această întrebare.
4) Metoda de identificare: cînd se doreşte ceva din memoria lentă se indică *adresa* la care acel obiect se găseşte. Principiul transparenței (faptul că interfețele sunt identice) implică faptul ca în cache datele să fie căutate tot după această adresă; dar cum cache-ul este mic, adresele din memoria externă (= memoria lentă) nu reprezintă adrese şi în memoria rapidă. Cum găsesc datele? Răspunsul este dat de *metoda de identificare* şi strîns legat de politica de înlocuire, pentru că datele trebuie căutate acolo unde puteau fi aduse.
5) Timpul de viață al informației: dacă dintre copiile pe care le avem (una în cache şi alta în memorie) una se schimbă? Care este cea bună după aceea? Ce trebuie făcut cu cealaltă? Cu ce ocazie trebuie făcută schimbarea?

5.2.2. Cache-ul de disc

Orice sistem de operare modern (mai puțin MS-DOS) are un cache de disc. (Chiar şi pentru MS-DOS există `smartdrive` sau `ncache` de la Norton). Cache-ul de disc este probabil una din cele mai mari surse de eficiență într-un sistem de operare. Acesta se datoreşte faptului că diferența între timpul de acces la disc şi cel la memorie este uriaşă (timpul de acces al unei memorii este de circa 60-70 de nanosecunde, adică $60x10^{-9}$ s, iar timpul de acces al unui disc este de ordinul a 10 milisecunde, adică $10x10^{-3}$ s. Cache-ul de disc este o structură de date care conține un vector de blocuri de

mărime egală. Discul este la rîndul lui împărțit în blocuri de aceeaşi dimensiune. Cînd utilizatorul cere un octet de pe disc, blocul care conține acel octet este încărcat în cache, eventual scoțînd un alt bloc afară.

Din cele 5 puncte de vedere indicate anterior, un cache de disc are următoarele caracteristici:

1) Mărimea blocului: Blocuri mari (512 octeți - 8 Kb).
2) Politica de înlocuire: Politica de înlocuire cea mai frecventă este cea de excludere a datelor nefoşosite de cel mai mult timp (Least Recently Used).
3) Politica de scriere: Cache-urile de disc sunt în general ``write-back''. Asta înseamnă că atunci cînd se *scrie* pe disc, modificările sunt făcute doar în cache. Ele sunt mutate pe disc doar cînd blocul respectiv este dat afară, sau cînd acest lucru este cerut explicit de utilizator.
4) Metoda de identificare: Pentru a găsi un bloc în cache se folosesc algoritmi de *hash*, care sunt foarte eficienți (orice carte elementară de algoritmi descrie hash-urile).
5) Timpul de viață al informației din cache: Pentru a preveni catastrofele, sistemele de operare ``golesc'' (scriu toate blocurile modificate) din cache pe disc periodic (de ex. la 30 de secunde).

5.2.3. Cache-ul microprocesorului

Un microprocesor la 200 de Megahertzi (un Pentium pro, de pildă) are un ciclu de instrucțiune de$1/(200x10^6) = 5$ nanosecunde. O instrucțiune poate dura un număr variabil de cicluri, între 1 şi cîteva zeci. Executarea unei instrucțiuni înseamnă: citirea ei din memorie, decodificarea, executarea, memorarea rezultatelor. Dacă accesul la memorie durează 60 de nanosecunde atunci la fiecare citire procesorul trebuie să piardă 12 cicluri! Din cauza asta între microprocesor şi memoria RAM principală se pune un cache construit din memorie rapidă, cu timp de acces de 5-10 nanosecunde.

Cîteodată designerii pun chiar mai mult decît atît: două nivele de cache între procesor şi RAM: un nivel ceva mai lent, dar mai mare (pentru un PC între 64Kb şi 512Kb de obicei), şi un cache construit chiar în microprocesor, de ordinul a 1-10Kb, mult mai rapid.
Aceste cache-uri se implementează folosind hardware specializat.

1) Mărimea blocului: blocurile sunt mici 1 - 16 octeți.
2) Politica de înlocuire si **4)** Metoda de identificare:

Există două clase mari de cache-uri de microprocesor, şi una intermediară. Ele diferă prin locurile din cache în care un octet din memoria externă poate fi plasat. Cele două mari varietăți sunt: cache-ul cu adresare directă, în care locul fiecărui octet este unul şi precis calculat, şi cache-ul asociativ, în care un octet din memoria externă poate fi plasat în orice loc din cache.

5.2.4. Cache-ul cu adresare directă (direct mapped)

De obicei chiar structura adresei este folosită la căutare. Figura 5.8 arată cum este plasat un anume bloc în cache: biţii de la sfîrşitul adresei blocului dau şi posibila poziţie a blocului în cache. Biţii din începutul adresei blocului constituie verificarea

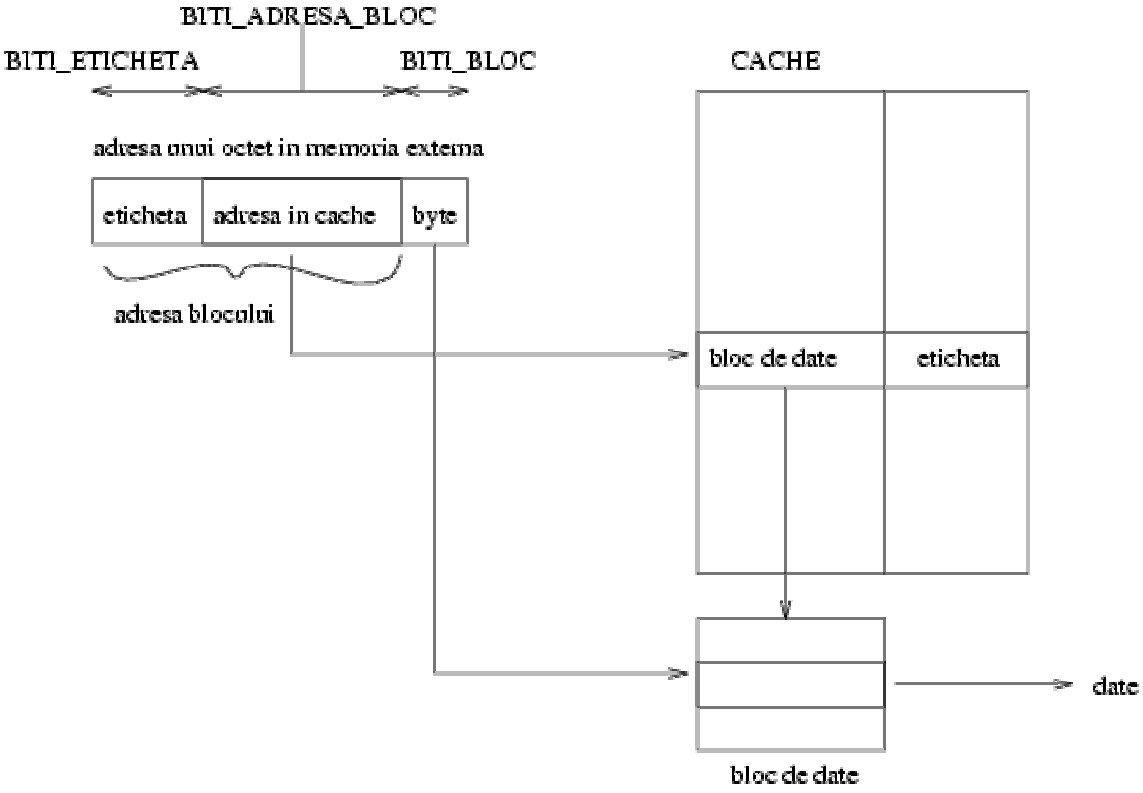


Figura 5.8. Cache-ul cu adresare directă

dacă blocul este cel aflat în cache (mai multe blocuri candidează pentru aceeaşi poziţie; cel care se află înauntru este indicat prin *etichetă* (tag)).

În fine, ultimii biţi din adresă indică poziţia octetului în blocul de date.

Marele avantaj al schemei directe este că dată fiind adresa, poziţia în cache a blocului este unic determinată, şi nu trebuie făcută nici o căutare. Politica de înlocuire nu există din acelaşi motiv: nu poţi alege în ce loc să aduci un bloc. Din cauza asta funcţia de căutare şi cea de înlocuire sunt identice.

5.2.5. Cache-ul cu adresare asociativă (fully associative)

Cache-ul cu adresare asociativă se bazează pe un dispozitiv hardware foarte simpatic, care se numeşte *memorie asociativă* (din cauza prezenţei ei îşi capătă cache-ul numele).

O memorie obişnuită oferă două operaţii: (a) dîndu-se o adresă, citeşte conţinutul şi (b) dindu-se o adresă şi o valoare scrie această valoare acolo.

Pe lîngă aceste operaţii o memorie asociativă mai oferă încă una: dîndu-se o valoare, poate spune la care adresă se găseşte ea. O memorie asociativă nu este tehnologic greu de construit, însă este un dispozitiv relativ costisitor.

Un cache asociativ foloseşte o memorie asociativă pentru a memora *adresele externe* ale blocurilor care corespund fiecărui bloc din cache.

Un bloc poate acum ocupa orice poziție în cache; cînd este căutat memoria asociativă spune unde se află.

Politica de înlocuire va fi însă ceva mai complicată, oricare din schemele înşirate fiind un candidat.

5.2.6. Cache-ul parțial asociativ (set-associative)

Putem să ne imaginăm un cache parțial asociativ ca o colecție de mai multe cache-uri directe care lucrează în paralel. Fie k numărul de astfel de cache-uri directe. (un astfel de cache se numeşte ``associative on k ways" -- asociativ pe k direcții).

Ideea este simplă: cînd caut o adresă folosesc adresare directă în *toate* cele k cache-uri directe simultan. Dacă blocul se găseşte într-unul am rezolvat problema. Daca nu, aleg unul dintre ele pentru înlocuire. Numele este de ``parțial asociativ", pentru că plasamentul în cele k blocuri posibile este oricare, ca la un cache asociativ.

Să revenim la discuția privind cache-urile microprocesoarelor.

[3)] Politica de scriere, **5)** Timpul de viață al informației din cache: Dacă mai multe microprocesoare sunt legate la aceeaşi memorie, există riscul ca fiecare să facă modificări în propriul cache, obținînd astfel rezultate eronate, aşa cum arată şi figura 5.9. Din cauza asta cache-ul se face adesea ``write-through": toate modificările se fac simultan în memorie şi în cache. Cache-urile monitorizează modificările făcute în memorie de celelalte cache-uri şi invalidează copiile datelor pe care le posedă şi care au fost modificate. (Un astfel de cache se numeşte ``snooping cache": cache care trage cu urechea, să vadă dacă altcineva nu modifică memoria externă).

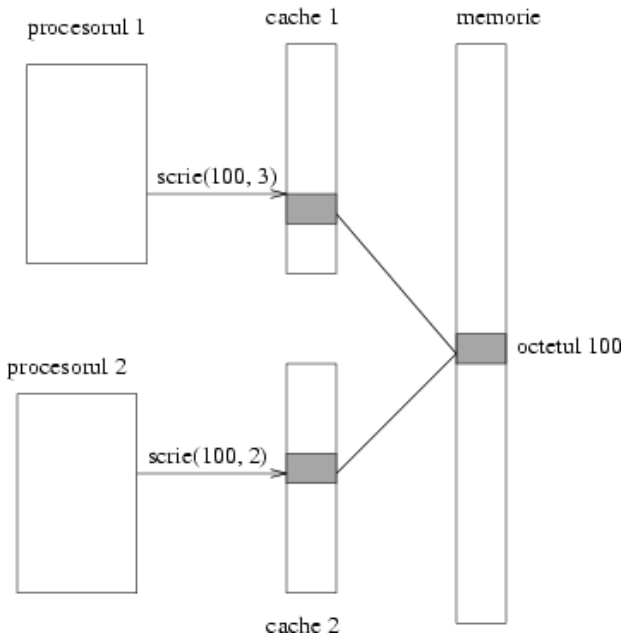


Figura 5.9. Acces prin două cache-uri

5.3. Gruparea memoriilor

Aşa cum s-a arătat, una din caracteristicile importante ale memoriei este reprezentată de capacitatea acesteia. Capacitatea reflectă cantitatea de informaţie ce se poate stoca într-o memorie şi este strâns legată de structura acesteia. Gruparea memoriilor reprezintă metoda de creştere a capacităţii de stocare prin conectarea împreună a mai multor memorii de capacitate mai mică. Pentru a vedea care sunt metodele de grupare a memoriilor vom studia mai întâi semnalele unei memorii.

O memorie are nevoie de mai multe semnale de intrare şi poate furniza mai multe semnale de ieşire. Aşa cum s-a arătat în paragraful 5.1, o memorie este alcătuită din mai multe celule de o anumită dimensiune (un anumit număr de biţi) care sunt identificate prin adresă. Din acest punct de vedere, o memorie necesită un anumit număr de adrese cu ajutorul cărora se identifică fiecare celulă de memorie în parte. Semnalele de adresă sunt semnale binare (pot avea valoarea zero sau unu), fiecare celulă de memorie fiind identificată de o combinaţie unică de pe magistrala de adrese. Astfel, dacă o memorie are, spre exemplu, opt celule de memorie, atunci sunt necesare opt combinaţii distincte de cifre zero sau unu. Cele opt combinaţii distincte pot fi obţinute cu ajutorul a $\log_2 8 = 3$ cifre binare. Altfel spus, cu ajutorul a trei cifre binare se pot obţine $2^3 = 8$ combinaţii distincte: 000, 001, 010, 011, 100, 101, 110 şi 111.

Celula de memorie selectată permite citirea sau scrierea *simultană* a tuturor biţilor pe care aceasta îi conţine. Din acest motiv, o memorie necesită pe lângă liniile de adresă şi un anumit număr de linii de date, numărul liniilor de date fiind egal cu numărul de biţi ai celulei de memorie. Liniile de date trebuie să fie bidirecţionale (ca sa permită citirea sau scrierea biţilor celulei de memorie) cu trei stări (să poată trece în starea de înaltă impedanţă) pentru a permite conectarea în paralel a mai multor memorii. Este evident că în cazul memoriilor ROM liniile de date nu sunt bidirecţionale din cauză că aceste memorii nu pot fi decât citite dar trebuie, de asemenea să aibă starea de înaltă impedanţă pentru a putea fi conectate la magistrala de date a sistemului.

Cea de a treia categorie de semnale necesare unei memorii sunt semnalele de comandă. Semnalele de comandă principale ale unei memoriei sunt:

- semnalul de comandă al selectiei circuitului (Chip Select – notat CS) care este un semnal de intrare în memorie şi care determină starea magistralei de date (acest semnal nu poate lipsi la nici un tip de memorie):
  - CS = 1, pe magistrala de date circulă semnale, fiecare linie a magistralei putând avea un semnal care este în starea unu sau zero logic (în memorie se poate scrie sau din memorie se poate citi);
  - CS = 0, pe magistrala de date nu circulă semnale, aceasta fiind în starea de înaltă impedanţă;
- semnalul de comandă al citirii din memorie sau scrierii în memorie ($\mathrm{Re}ad/\overline{Write}$ - notat $R/\overline{W}$) care este un semnal ce determină direcţia magistralei de date (acest semnal lipseste la memoriile ROM):
  - $R/\overline{W} = 1$, din memorie se citeste, sensul magistralei de date fiind din memorie spre exterior; la un moment dat se citesc biţii celulei

de memorie, selectate de combinația aflată pe magistrala de adrese;

- $R/\overline{W}=0$, în memorie se scrie, sensul magistralei de date fiind din exterior spre memorie; la un moment dat se scriu biții celulei de memorie, selectate de combinația aflată pe magistrala de adrese;

Pe lîngă aceste semnale, mai există o serie de semnale de comandă diverse, cum ar fi semnalele de reâmprospătare, care nu sunt implicate în funcționarea de bază a memoriei.

Memoria, ca matrice, este definită de declarația:

memoria: array (0…p) of byte

unde ***p*** reprezintă numărul de celule de memorie cu dimensiunea de un octet (byte). O astfel de memorie are nevoie de $\log_2 n$ linii de adrese şi opt linii de date.

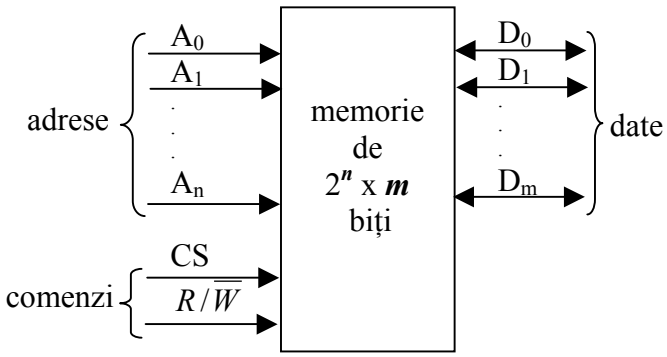


Figura 5.10. Schema bloc a memoriei.

Bineînțeles că celula de memorie poate avea şi dimensiunea de un cuvânt (word) sau pointer sau dimensiuni mai mici cum ar fi 1 bit sau 4 biți.

În continuare vom considera memorii generice care au adrese, date şi semnalele de comandă: CS şi $R/\overline{W}$, aşa cum se arată în schema bloc a unei memorii generice din figura 5.10.

Capacitatea memoriei poate fi exprimată de către o relație similară cu relația (5.1).

capacitatea = $2^{\text{numărul de linii de adresă } (n+1)}$ x numărul de linii de date ($m+1$) (5.1)

De exemplu, capacitatea unei memorii cu zece linii de adresă şi opt linii de date va fi:

$$C = 2^{10} \text{ x } 8 \text{ biți} = 1024 \text{ x } 8 \text{ biți} = 1\text{k octet} = 1\text{ko}$$

Capacitatea exprimată în forma dată de ecuația 5.1 se numeşte capacitate structurală deoarece permite determinarea structurii unei memorii. Pentru exemplul anterior avem: C = 1024 x 1 octet, care ne permite să aflăm structura memoriei: 10 linii de adresă ($\log_2 1024 = 10$), care pot selecta în mod univoc 1024 ($2^{10} = 1024$) celule de memorie care au câte 8 biți (magistrala de date este de un octet).

Capacitatea se poate exprima şi prin valoarea ei absolută, rezultatul înmulțirii numărului de celule de memorie cu dimensiunea acestora, obținându-se numărul de

celule *elementare* de memorie cu dimensiunea de un bit (de exemplu 1ko), valoare care însă nu ne mai dă informații asupra structurii. O memorie de 1ko poate fi: o memorie de 1024 x 1 octet sau o memorie de 2048 x 4 biți.

Gruparea memoriilor poate face în trei moduri:

- gruparea în scopul creşterii numărului de linii de date ale memoriei, numărul de linii de adresă rămânând neschimbat; creşterea numărului de linii de date a memoriei înseamnă de fapt creşterea numărului de biți ai celulei de memorie (numărul de biți citiți sau scrişi din/în memoria respectivă) ai memoriei rezultate. Evident că mărirea capacității celulei de memorie va determina creşterea corespunzătoare a capacității totale a memoriei;
- gruparea în scopul creşterii numărului de linii de adresă a memoriei, numărul liniilor de date rămânând nemodificat; creşterea numărului de linii de adresă înseamnă de fapt cresterea numărului de celule de memorie ai memoriei rezultate, capacitatea celulei (numărul de linii de date) rămânând neschimbat; şi în acest se produce modificarea corespunzătoare a capacității memoriei;
- gruparea mixtă în care se creşte atât numărul de linii de adresă cât şi numărul liniilor de date.

5.3.1. Creşterea capacității memoriei prin creşterea numărului de linii de date

Creşterea numărului de linii de date este necesară, spre exemplu, atunci când dispunem de memorii cu opt linii de date (un octet) şi dorim să le conectăm la un sistem care are o magistrală de date cu 16 linii (un cuvânt – word).

Vom exemplifica modul de grupare pe acest exemplu. Presupunem că avem nevoie de o memorie de 8kw (8 kilo word – 8 kilo cuvinte) şi nu dispunem decât de memorii de 8kb (8 kilo byte – 8 kilo octeți). Se observă faptul că între memoriile disponibile şi cele necesare nu diferă decât numărul de linii de date (necesar 16, existent 8), numărul de linii de adresă fiind acelaşi (8192 de celule înseamnă $\log_2 8192 = 13$ linii de adresă: A_0 la A_{12}). Vom determina mai întâi cîte memorii sunt necesare. Capacitatea absolută a memoriei ce trebuie obținută este:

$$C = 8192 \text{ x } 16 \text{ biți} = 131\,072 \text{ biți}.$$

Capacitatea memoriei existente este:

$$C = 8192 \text{ x } 8 \text{ biți} = 65\,536 \text{ biți}.$$

Rezultă deci că pentru a obține o memorie de 8 kword avem nevoie de 131072/65536 = 2 memorii de 8 kbytes.

Modul de conectare a celor două memorii de 8 kb pentru obținerea unei memorii de 8 kiloword este prezentat în figura 5.11.

Se observă că pentru gruparea memoriilor în vederea extinderii numărului de linii de date, liniile de adresă şi liniile de comandă se leagă în paralel iar liniile de date se pun împreună formând magistrala de date a memoriei rezultate.

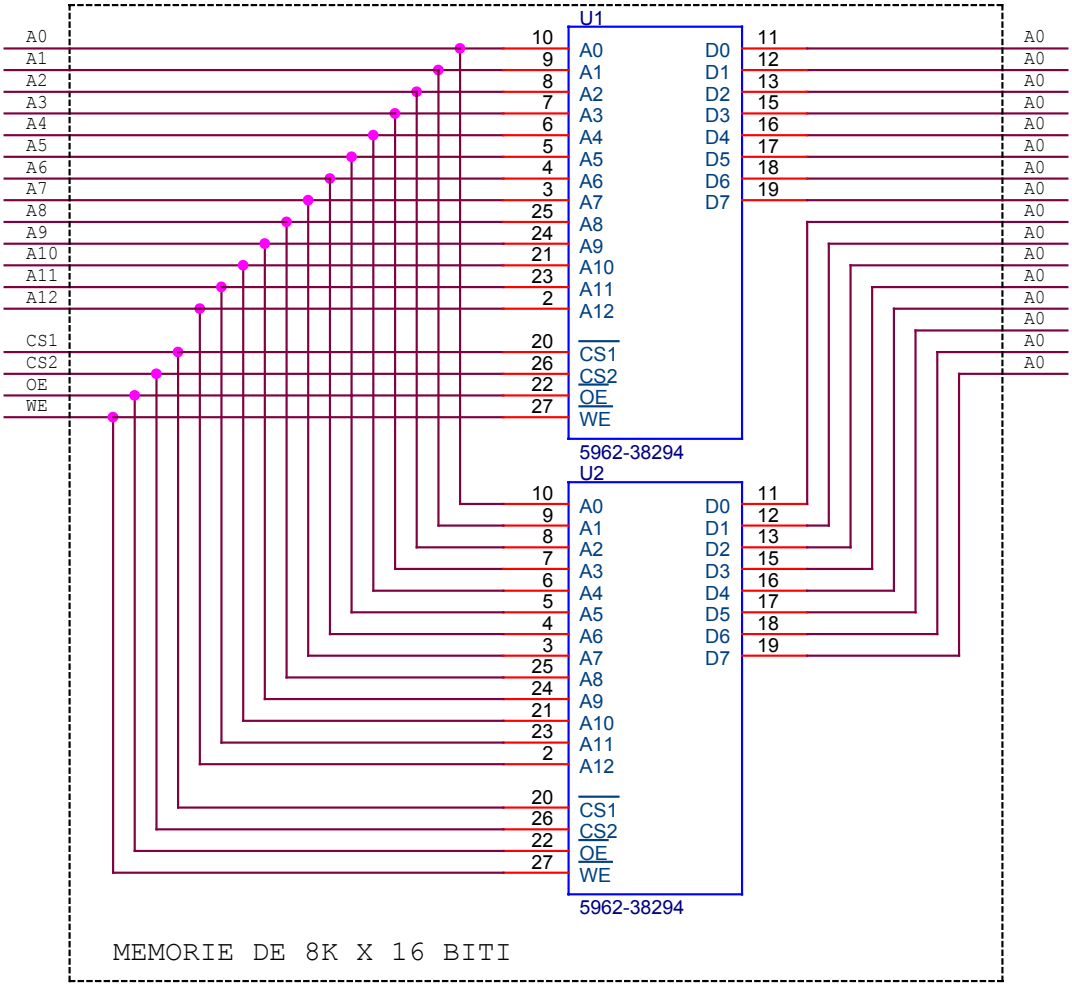


Figura 5.11. Gruparea memoriilor pentru creşterea numărului liniilor de date.

5.3.2. Gruparea memoriilor pentru creşterea numărului de linii de adresă

Creşterea numărului de linii de adresă este necesară pentru creşterea capacităţii unei memorii fără modificarea dimensiunii celulei de memorie.

Pentru a explica principiul vom considera un exemplu simplu. Să presupunem că dorim să obţinem o memorie de 8 x 1 bit cu ajutorul unor memorii de 4 x 1 bit. Atât memoria pe care dorim s-o obţinem cât şi memoriile disponibile au celule de aceeaşi dimensiune (un bit). Ceea ce se doreşte este dreşterea numărului acestor celule. Memoriile disponibile, cele de 4 x 1 bit, necesită două linii de adresă pentru selectarea uneia din cele patru celule iar memoria care se doreşte a fi obţinută, cea de 8 x 1 bit are nevoie de 3 linii de adresă. Rezultă că numărul de linii de adresă trebuie crescut de la două linii la trei.

Vom analiza acum modul de adresare al celulelor de memorie. Fiecare celulă a memoriei este adresată în mod univoc de către o combinație binară de pe magistrala de adrese aşa cum se arată în tabelul 5.3. Din acest tabel se observă faptul că cele patru combinații ale liniilor de adresă A_1A_0 se repetă de două ori la memoria de 8 x 1 bit, o dată pentru $A_2 = 0$ şi o dată pentru $A_2 = 1$. Rezultă că este posibil să realizăm memoria de 8 x 1 bit cu ajutorul a două memorii de 4 x 1 bit este necesar sa selectam una din memoriile de 4 x 1 bit când $A_2 = 0$ şi să selectăm cealalta memorie de 4 x 1 bit când $A_2 = 1$. Rezultă că semnalele de selacție a memoriilor de 4 x 1 bit vor fi folosite pentru crearea liniei de adresă A2 şi va trebui sa creem un nou semnal de selecție pentru memoria rezultata. Vom exemplifica pe acelaşi tip de memorie de 8k x 1 octet. Prin gruparea acestora în scopul creşterii numărului de linii de adresă (creşterea numărului de celule de opt biți) se va crea o noua linie de adresă (A_{13}) memoria obținută având 16k x 1 octet.

TABELUL 5.3.

memorie 4x1bit		memorie 8 x 1bit		
A_1	A_0	A_2	A_1	A_0
0	0	0	0	0
0	1	0	0	1
1	0	0	1	0
1	1	0	1	1
		1	0	0
		1	0	1
		1	1	0
		1	1	1

În acest caz, liniile de date se cuplează în paralel deoarece dimensiunea celulei de memorie rămâne nemodificată (figura 5.12). Memoria inactivă la un moment dat are

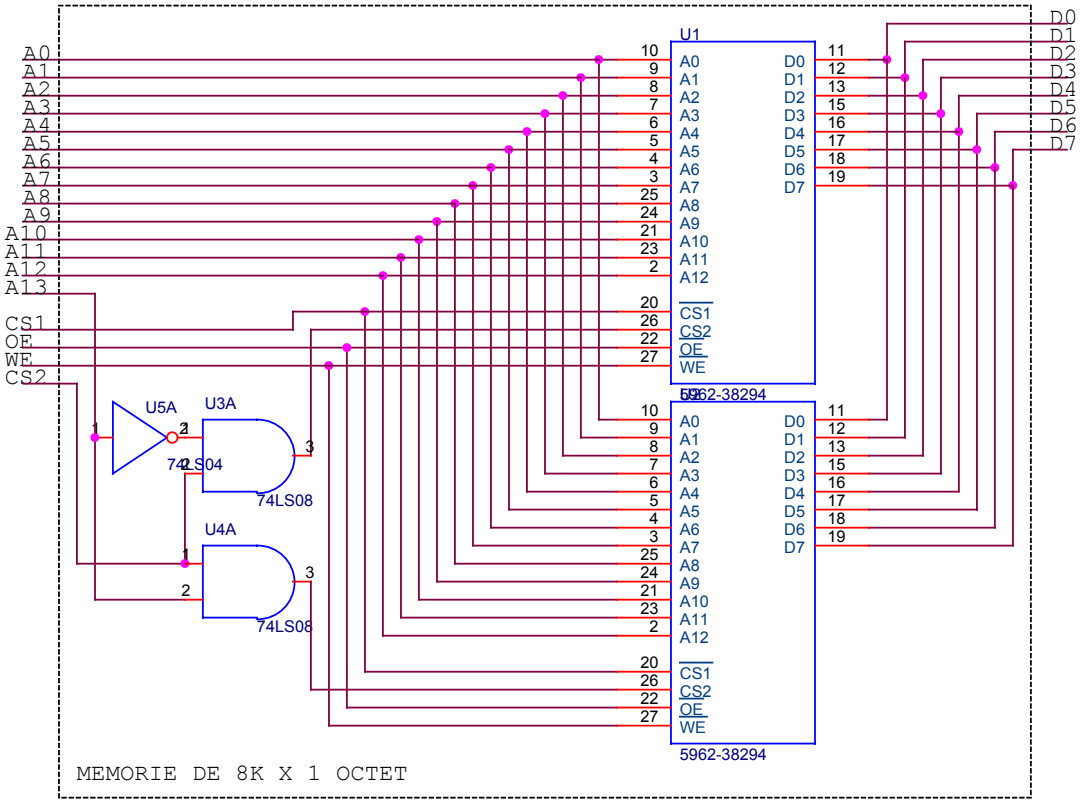


Figura 5.12. Gruparea memoriilor prin extinderea liniilor de adresă.

liniile de date în starea de înaltă impedanță (prin comanda selecției ciruitului respectiv) şi în acest fel liniile de date ale celor două memorii pot fi cuplate împreună.

OBSERVAȚIE: din cele două exemple prezentate anterior rezultă valoarea capacității memoriei finale este multiplu întreg de valoarea capacitîții memoriei finale.

5.3.3. Gruparea mixtă

Gruparea mixtă constă în creşterea atât a numărului de linii de date cât şi a creşterii numărului de linii de adresă ceea ce inseamnă creşterea dimensiunii celulei de memoriei şi a numărului de celule.

Pentru a realiza o astfel de grupare se realizează mai întâi gruparea memoriilor în vederea cresterii numărului de linii de date după care memoriile obținute se grupează prin cresterea numărului de linii de adresă, obținându-se în final memoria dorită.

Vom exemplifica acest lucru prin realizarea unei memorii de 2k octeți cu ajutorul unor memorii de 1k x 4 biți.

Capacitatea absolută a memoriei rezultate este:

$$C = 2 \text{ x } 1024 \text{ x } 8 \text{ biți} = 16384 \text{ biți}$$

Această capacitate trebuie să fie multiplu întreg de capacitatea memoriilor inițiale. Memoria inițiala are capacitatea absolută:

$$C = 1024 \text{ x } 4 \text{ biți} = 4096 \text{ biți.}$$

Rezultă că sunt necesare 16384 : 4096 = 4 memorii de 1k x 4 biți. Aceste patru memorii se vor grupa astfel: mai întâi memoriile se grupează două câte două prin extinderea numărului de linii de date, obținânduse două memorii cu dimensiunea de 1k x 8 biți. Cele două memorii obținute se grupează prin extinderea numărului de linii de adresă, rezultând memoria finală de 2k octeți.

5.4. Adresarea memoriilor

Într-un sistem de calcul, principalul modul master care generează adrese este unitatea centrală. Numărul total de adrese pe care-l poate genera unitatea centrală reprezintă *spațiul maxim de adresare*. Acest spațiu se împarte, conform semnalelor de comandă generate de unitatea centrală, între memorie şi porturi. Există mai multe moduri de adresare a acestora în funcție de utilizarea spațiului maxim de adresare. Adresarea memoriei (sau a porturilor) poate fi *absolută* dacă la formarea adresei se folosesc toți biții de adresă sau relativă (care se mai numeşte şi *redundantă*) dacă se folosesc numai o parte din biții de adresă.

5.4.1. Adresarea absolută

La adresarea absolută se folosesc toți biții de adresă de pe magistrala de adrese pentru selectarea unei locații de memorie (sau a unui port).

valoare în zecimal	adrese pentru selecția memoriei de 1k						adrese pentru selecția unei celule din memoria de 1k				
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	…	A_1	A_0
0	0	0	0	0	0	0	x	x	…	x	x
1	0	0	0	0	0	1	x	x	…	x	x
2	0	0	0	0	1	0	x	x	…	x	x
.	.										
.	.										
.	.										
63	1	1	1	1	1	1	x	x	…	x	x

x x … x x -> orice combinație între 00…00 şi 11 … 11

Figura 5.13. Posibilitățile de adresare absolută a unei memorii de 1k.

Vom explica adresarea absolută printr-un exemplu. Considerăm că într-un sistem magistrala de adrese are 16 linii (A_0, A_1, … , A_{15}) deci spațiul maxim de adresare este de 64 k (2^{16} = 65536). Pe acestă magistrală dorim să conectăm o memorie de 1k (dimensiunea datelor nu are importanță aici). Pentru adresarea celor 1024 de celule de memorie din memoria de 1k avem nevoie de 10 linii de adresă (A_0, A_1, … , A_9) restul de 6 linii de adresă (A_{15}, A_{14}, … , A_{10}) fiind folosiți la selecția circuitului. Astfel adresele A_0, A_1, … , A_9 se leagă la liniile de adresă ale memoriei iar adresele A_{10}, A_{11}, … , A_{15} se leagă printr-un circuit de selecție la intrarea CS a memoriei. În acest fel memoria poate fi aşezată la una din cele 64 de adrese posibile din memorie (figura 5.13).

Prin adresarea absolută, o locație de memorie se găseşte la o singură adresă de memorie.

5.4.2. Adresarea relativă (redundantă)

În cazul adresării redundante, adresele (sau o parte dintre acestea) nu sunt folosite la selecția memoriei. În acest fel o celulă de memorie se poate găsi la mai multe adrese diferite.

Reluând exemplul din paragraful anterior, dacă se leagă semnalul de selecție al memoriei de 1k la valoarea unu logic (memorie permanent selectată) şi în acest fel se face abstracție de adresele A_{15}, A_{14}, … , A_{10}, adresarea unei locații din această memorie este redundantă.

Astfel, celula de memorie de la adresa 0 se va găsi la 64 de adrese distincte: 0, 400h, 800h, C00h şi aşa mai departe, pâna la adresa FFFFh. Acest lucru se întâmplă deoarece numai adresele A_0, A_1, … , A_9 participa la selecția celulei, adresele fiind ignorate.

CAPITOLUL 6

PORTURI (INTERFEȚE)

6.1. Prezentare generală

Porturile sunt module destinate conectării sistemului de calcul cu dispozitivele externe care sunt numite generic dispozitive ***periferice***. La porturi se conectează atât dispozitivele generale de intrare/ieşire (tastatură, monitor) ale calculatorului cât şi cele specifice. De asemenea două porturi de aceeaşi natură se pot conecta între ele în scopul transmiterii informației de la un sistem de calcul la altul. Porturile sunt în general circuite de viteze mică, ele fiind conectate la magistralele de viteză mai redusă ale calculatorului. O parte din porturile sistemului de calcul au destinații specifice şi poartă denumirea de ***controlere***. Dintre acestea mai importante sunt: controlerul de tastatură, controlerul video, controlerul discului dur (hard disc), controlerul discului flexibil (floppy disc) şi controlerul de rețea. Alte porturi au destinații generale, la aceste porturi putând fi conectate diferite dispozitive periferice; dintre porturile de uz general mai importante fiind portul serial, portul paralel, portul SCSI (Small Computer System Interface) şi portul USB (Universal Serial Bus). Datorită faptului că, în general, un port trebuie să realizeze atât conversia semnalelor dispozitivelor periferice la semnale compatibile cu semnalele sistemului de calcul, cât şi invers, asigurând circulația informației în ambele sensuri, acestea se mai numesc şi ***interfețe***. Prin interfață se înțelege un ansamblu format din echipamente şi programe destinat cuplării între două sisteme cu caracteristici diferite. Dispozitivele periferice se conectează la porturi prin intermediul unor conectori, un exemplu fiind prezentat în figura 6.1.

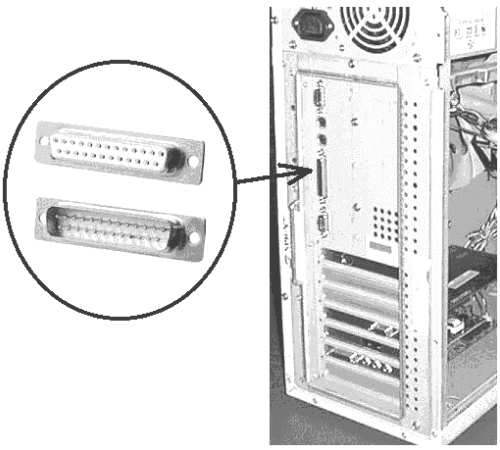


Fig. 6.1. Conectorii pentru porturi în cazul unui calculator personal

Fiecare port are alocată o adresă (un număr) pe care unitatea centrală o foloseşte pentru identificarea portului respectiv în schimbul de date cu acesta. Este foarte important ca toate porturile dintr-un calculator să aibă adrese diferite. Dacă două porturi au aceeaşi adresă, atunci apare un conflict din cauză că unitatea centrală nu va fi capabilă să distingă de la care port sosesc informațiile.

În general activitatea de atribuire a adreselor tuturor porturilor dintr-un sistem de calcul se face în mod automat de către ***sistemul de operare***. Uneori, dacă numărul

perifericelor este mare sau din generații diferite, se poate întâmpla ca sistemul de operare să nu poată realiza alocarea corectă a adreselor perifericelor. În acest caz alocarea adreselor se poate face manual de către operator.

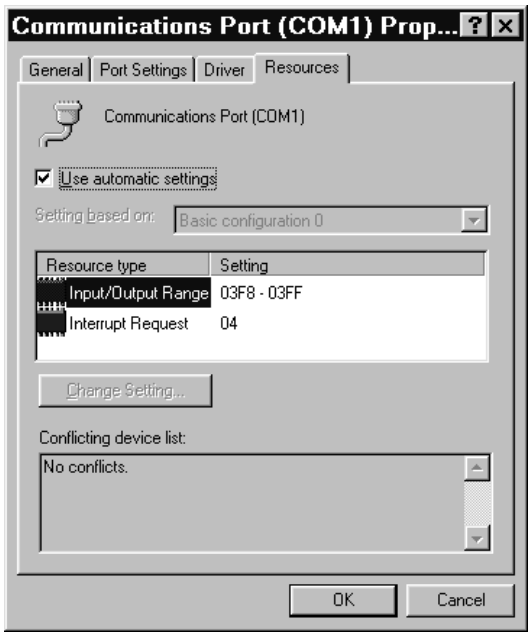


Fig. 6.2. Alocarea adresei şi întreruperii la portul serial COM1

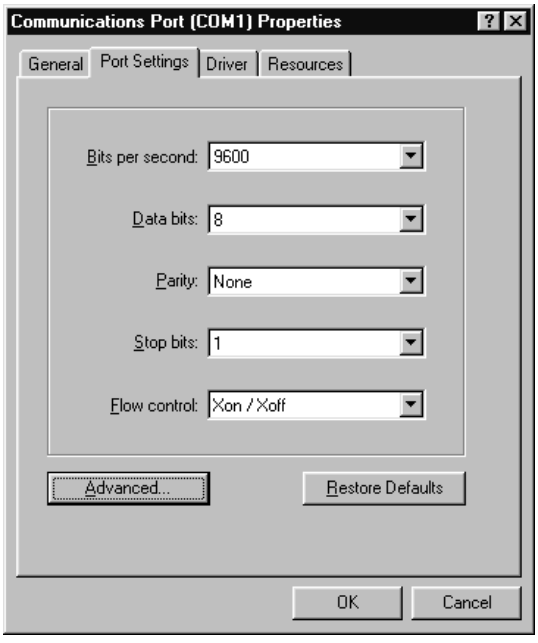


Fig. 6.3. Parametrii portului serial

Porturile fiind în general componente lente ale sistemului de calcul, este neeconomic ca unitatea centrală să lucreze la un moment dat numai cu un port. Modul de lucru cu un port se desfăşoară simplificat în felul următor: unitatea centrală trimite o comandă la un port după care continuă rezolvarea altor sarcini. În momentul în care portul este capabil să răspundă solicitării, întrerupe activitatea curentă a unității centrale, primeşte o nouă comandă, după care unitatea centrală nu se mai ocupă de port până la o nouă întrerupere. În felul acesta oricât de lent este un port el nu scade viteza de lucru a unității centrale. Acest mecanism necesită, pe lângă alocarea unei adrese şi alocarea unei ***întreruperi*** (un număr) fiecărui port în aşa fel încât în momentul lansării unei cereri de întrerupere, unitatea centrală a calculatorului să poată determina care este portul care a cerut întreruperea. Evident că şi în cazul întreruperii este bine ca fiecare port să dispună de o întrerupere separată. Deşi această condiție nu este la fel de restrictivă ca cea în cazul adresei, în unele cazuri mai multe porturi putând avea alocată aceeaşi întrerupere, în marea majoritate a cazurilor numărul întreruperii trebuie să fie diferit de la un port la altul.

Din cele arătate rezultă că un port necesită alocarea din partea sistemului de operare sau a programatorului a două numere: adresa şi numărul întreruperii, fapt ce-l individualizează față de celelalte porturi din sistem. În figura 6.2 este prezentat modul în care sistemul de operare Windows'95 oferă informații despre setările unui port. Din această figură se vede că portul de comunicații COM1 are alocate adresele de la 03F8h la 03FFh şi întreruperea 04.

Portul serial al calculatorului necesită o atenţie specială din cauză că el este în general portul de comunicaţii. Prin intermediul acestui port se pot transmite date la distanţă în mod ***serial asincron***. Transmisia serială este metoda cea mai ieftină de a schimba date între două echipamente numerice aflate la distanţă. Deşi transmisiile seriale nu sunt transmisii de date de viteză mare ele prezintă avantajul că datele sunt transmise pe un singur canal de comunicaţie şi deci nu sunt necesare cheltuieli mari. Modul serial de transmisie presupune ca la un moment dat să se transmită un singur bit. Asta înseamnă că biţii unui octet vor fi transmişi la opt intervale de timp distincte. În cazul transmisiei seriale este necesar ca atât echipamentul care transmite datele (emiţătorul) cât şi echipamentul care primeşte datele (receptorul) să aibă aceeaşi ***parametrii ai transmisiei***. Pentru transmisia serială asincronă parametrii transmisiei sunt: viteza de transmisie care se măsoară în biţi pe secundă (bps), numărul de biţi transmişi într-un cadru, paritatea, numărul de biţi de stop şi protocolul de control al fluxului de date. În figura 6.3 este prezentat un exemplu de stabilire ai parametrilor transmisiei. Standardul adoptat pentru interfaţa seriala este standardul RS 232.

Porturile sunt în general dispozitive programabile. Asta înseamnă că ele acceptă prin înscrierea unor cuvinte de comandă în port. Din acest motiv, pentru funcţionarea corectă a unui port nu sunt suficiente numai alocarea adresei şi a întreruperii ci este necesară şi programarea portului. Pentru simplificarea programării şi utilizării portului se folosesc nişte programe specifice numite ***drivere***. Aceste programe sunt încărcate în memorie de către sistemul de operare şi ele constituie o interfaţă între programele utilizatorului şi port. În figura 6.4 este reprezentat schematic modul de funcţionare a unui driver. Programele driver sunt furnizate de către producătorii de echipamente periferice şi asigură funcţionarea optimă a acestora. Din acest motiv utilizatorul unui sistem de calcul nu are decât sarcina de a obţine de la producătorii de echipamente periferice a versiunilor noi ale driverelor şi să le instaleze pe calculator.

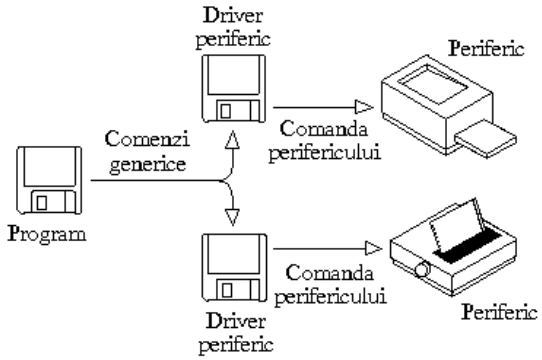


Fig. 6.4. Funcţionarea unui program driver pentru un port

Noua tehnologie de realizare a porturilor inteligente s-a extins tot mai mult. Cu ajutorul acestei tehnologii, utilizatorul este degrevat de sarcina de a mai configura porturile sistemului, acestea fiind recunoscute automat şi programate corespunzător de către programele cu care este înzestrat sistemul de calcul. Această tehnologie numită PnP (Plug and Play) permite utilizatorului să realizeze extinderi ale sistemului de calcul simplu şi comod, aşa cum arată şi numele tehnologiei, prin simpla montare a componentei noi în sistemul de calcul, fără a mai fi necesare alte operaţii suplimentare.

În continuare vor fi prezentate porturile dezvoltate de firma Intel pentru sistemul de calcul prevăzut cu unitatea centrală I8086. Datorită faptului că aceste dispozitive au fost larg răspândite ele au devenit un standard "de facto" pentru circuitele dezvoltate ulterior.

6.2. Interfața serială programabilă 8251

Interfața serială programabilă 8251, cunoscută şi sub numele de Receptor Transmițător Universal Sincron/Asincron (USART), este realizată în tehnologia NMOS, pe o pastilă cu 28 de terminale. Ca interfață serială de I/E acceptă date, sub formă de caractere, transmise în paralel, de către unitatea centrală de prelucrare şi le converteşte într-un format serial, pentru a fi transmise către un echipament de ieşire. În acelaşi timp, poate recepționa date transmise serial, de către un echipament de intrare, convertindu-le în caractere, ce se transferă în paralel, către unitatea centrală de prelucrare.

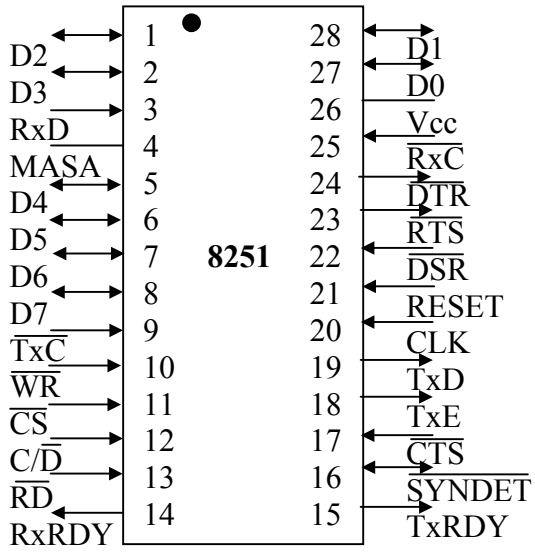


Figura 6.5. Semnificația terminalelor interfeței seriale programabile 8251.

D_7-D_0 - conexiuni la magistrala de date a Microsistemului (bidirecțional);
RESET - aducere în condiții inițiale (intrare);
CLK - cesul dispozitivului (intrare);
C/D - semnal de selecție: comandă/data (intrare);
$\overline{RD}$ - citeşte data sau starea pe magistrala D_7-D_0 (intrare);
$\overline{WR}$ - scrie data sau comanda de pe magistrala D_7-D_0 (intrare);
$\overline{CS}$ - selecție circuit (intrare);
$\overline{DSR}$ - indicator echipament de date pregătit (intrare);
$\overline{DTR}$ - indicator terminal de date pregătit (ieşire);
$\overline{CTS}$ - indicator anulare în vederea Transmisiei (intrare);
$\overline{RTS}$ - indicator cerere în vederea transmisiei (ieşire);
TxD - ieşire serială de date (ieşire);
TxRDY - transmițător pregătit pentru a primi date pe magistrala D_7-D_0 (ieşire);
TxE - transmițător vid, nu are date de transmis (ieşire);
$\overline{TxC}$ - ceasul pentru transmisie serială (ieşire);
RxD - intrare serială (intrare);
RxRDY - un caracter este pregătit pentru a fi transmis pe magistrala D_7-D_0 (ieşire);
$\overline{RxC}$ - ceasul pentru recepție serială (intrare);
SYNDET – forțare sau detecție sincronă de date (bidirecțional);

Interfața semnalizează către unitatea centrală de prelucrare, în cazurile în care poate accepta un nou caracter, în vederea transmisiei sau când a recepționat un caracter, pentru unitatea centrala. Interfața poate fi programată pentru diverse opțiuni privind funcționarea, iar starea ei poate fi citită, în orice moment, de către unitatea centrală de prelucrare. În figura 6.5 sunt prezentate semnificațiile terrninalelor iar în figura 6.6 schema bloc a interfeței seriale prograrnabile 8251.

Comunicația serială, cu echipamente locale sau plasate la distanță, legate prin modemuri, poate fi realizată de către interfața 8251, fie sub forma asincronă, fie sub forma sincronă.

În cazul transisiei asincrone (fig. 6.7) fiecărui caracter manipulat i se adaugă informația de cadru, constând dintir-un bit de START, definit ca zero logic şi unul sau doi biți de STOP, definiți ca unu logic. Frontul negatlv al bitului de START indică receptorului îuceputul procesului de asamblare a unui caracter din fluxul serial de biți, care urmează, precum şi sincrconizarea cu transmițătorul. Sincronizarea are loc pe

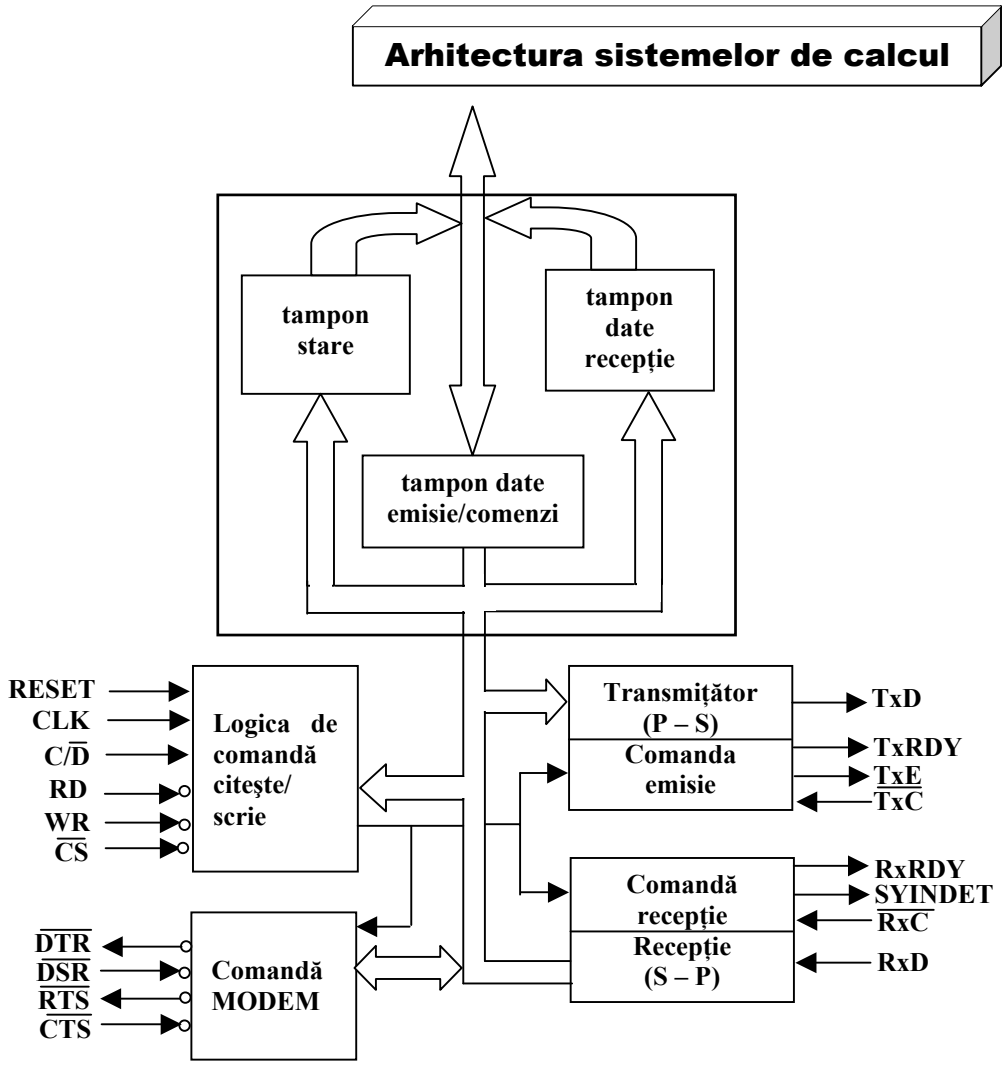


Figura 6.6. Schema bloc a interfeței seriale programabile 8251

durata transmisiei unui singur caracter. Bitul sau biții da STOP, adăugați la sfârşitul caracterului, asigură tranziția necesară pentru bitul de START al unui eventual nou caracter. Aceasta permite adaptarea receptorlui la viteza de lucru a transmițătorului. Dacă ceasul receptorului este puțin mai rapid decât al transmițătorului, primul va recepționa caracterele cu pauze între ele, dar le va recepționa corect.

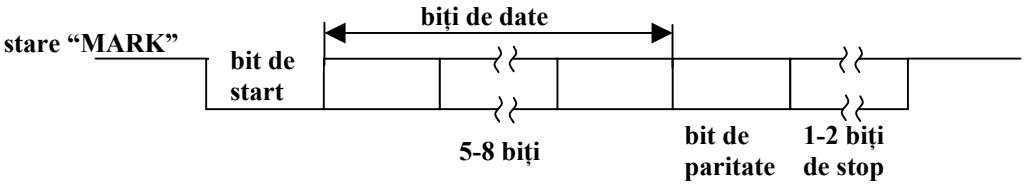


Figura 6.7. Formatul asincron pentru transmisia serială.

În cazul transmisiei sincrone (fig. 6.8), caracterete se asanablează sub forma unor înregistrări, adăugîndu~se caractere de cadru la fiecare început de înregistrare. Caracterele de cadru (SYN) sunt folosite de catre receptor pentru a determina începutul unei noi înregistrări. Întrucât sincronizarea trebuie menținută pe parcursul unui şir destul de lung de caractere, informația referitoarre la aceasta se extrage din canalul de cocnunicație sau de la o sursă externă.

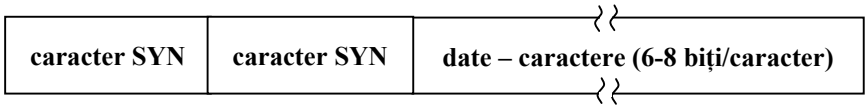


Figura 6.8. Formatul sincron pentru transmisia serială.

Comparând cele două modalități de transmisie se constată că, perntru mesaje care depăşesc 8 caractere, devine mai eficientă transmisia sincronă. Acest lucru poate fi evldențiat prin calculul numărului de biți suplimentari de START, STOP şi respecitv de caractere SYN, ce insoțesc datele pentru a fi transmise.

Transmisia asincronă la distanță se efectuează cu modemuri asincrone, care folosesc semnale de frecvențe diferite pentru unu şi respectiv zero logic.

La transmisia sincronă modemul furnizează semnalul de sincronizare către terminal şi impune ca datele să-i fie livrate sincron cu acest semnal. Modemurile sincrone pot opera numai la frecvențe pcestabilite. Modemul receptor, care are un oscilator, lucrând pe aceeaşi frecvență, cu cel din modemul transmițător, işi ajustează faza dupa cel din transmițător şi interpretează orice modificare de fază ca fiind o informație, o data.

În unele canzuri, când se urmăreşte creşterea vitezei de operare, fără a schimba protocolul, se operează într-un mod hibrid, datele cu format asincron fiind transmise sincron. Această transmisie poartă numele de isosincronă.

Interfața programabilă 8251 poate lucra în modurile sincron, asincrun şi isosincron.

În modul sincron ea manipuleaza caractere de 5, 6, 7 sau 8 biți, cu adaugarea şi respectiv verificarea unui bit de paritate (pară sau impară). Sincronizarea poate fi realizată extern, printr-un hardware adecvat, sau intern, prin detectarea caracterului SYN. Caracterele SYN pot fi diferite. Ele sunt inserate automat, de către interță, la transmisie, pentru a nu pierde sincronizarea, dacă software-ul nu furnizează la timp datele.

La transmisie asincronă, interfața operează cu caractere 5,6,7 sau 8 biți şi cu adaugarea/verificarea bitului de paritate (pară sau impară). Se adaugă 1 bit de START şi 1, l/2 sau 2 biți STOP. Receptorul testează cadrarea corectă şi poziționează un indicator, în cazul unei erori. Interfața poate fi programată sa accepte semnale de ceas, cu frecvența de 16 sau 64 ori mai mare decât cea a semnalelor transmise.

Transmisia isosincronă se consideră ca un caz special de transmisie asincrona cu frecvența programată a interfeței ca fiind egală cu cea a ceasului transmițătorului/receptorului. Formatele asincron, sincron şi iso-sincron pot fi transmise în modurile semiduplex şi duplex, datorită existenței în interfață a unor tampoane duble pentru date.

Interfața 8251 nu asigură toate semnalele de comandă pentru un echipament de transmisie a datelor, conform standardului EIA-RS-232-C. Pentru generarea semnalelor neasigurate, de catre 8251, se poate folosi un port auxiliar, al microprocesorului. Nivelurile de tensiune solicitate de standardul EIA-RS-232-C vor fi realizate prin circuite de atac şi recepție corespunzitoare (1488 şi respectiv 1489).

Schema bloc din figura 6.6 constă din cinci secțiuni, care comunică între ele prin intermediul unei magistrate interne. Cele cinci secțiuni sunt: receptorul, transmițatorul, comanda modemului, comanda pentru citire/scriere şi tamponul de I/E. Acesta din urmă

a fost prezentat mai detaliat şi constă din următoarele subsecţiuni: tamponul de stare, tamponul de transmisie date şi comenzi, tamponul de recepţie date.

Receptorul primeşte datele sub formă sarială, pe terminalul RxD, pe care le asamblează apoi în cuvinte, în conformitate cu un anumit format. În cazul în care este pregătit să primească un caracter, în modul asincron, urmăreşte tranziţia negativă a semnalului RxD. La apariţia unei asemenea tranziţii declanşează un generator intern, pentru a putea genera un interval de timp, egal cu jumătatea perioadei unui bit. În cazul în care testul, după o jumătate de perioadă,de bit, indică un nivel coborât, se consideră că s-a recepţionat bitul de START al unui cuvtnt. În continuare se asamblează cuvântul recepţionat serial, prin testarea lui RxD, la fiecare jumătate de interval de bit. Cuvântului asamblat i se ataşează biţii de paritate şi de STOP, după care este transferat pe magistrala internă, în tamponul datelor recepţionate, activându-se semnalul RxRDY, pentru a indica unităţii centrale de prelucrare disponibilitatea unui caracter. Daca la testarea bitului de START, se constată că la jumătatea intervalului de bit RxD este la nivel ridicat sau daca s-a activat receptorul pe parcursul transmisiei unui caracter, operaţia se anulează şi se reâncepe procesul de testare pentru recepţia unui nou caracter. La recepţia caracterelor având mai puţin de 8 biţi, examinarea se face la dreapta, iar semnalul RxRDY este activat pentru a indica disponibilitatea unui caracter.

La transmisia smcronă receptorul înregistrează un număr specificat de biţi, pe care îi transferă în registrul tampon de recepţie, activând RxRDY. Pentru a grupa corect biţii receptionaţi, receptorul trebuie sincronizat cu emiţătorul ceea ce se realizează în modul HUNT.

In modul HUNT interfaţa citeşte datele serial, bit cu bit, pe linia RxD, comparând, după fiecare bit recepţionat, conţinutul registrului de recepţie cu cel care păstrează caracterul SYN, încărcat prin program. Modul HUNT ia sfârşit când cele doua caractere sunt identice, specificându-se realizarea sincronizării prin activarea semnalului SYNDET. Dacă interfaţa a fost programată să accepte două caractere SYN, sincronizarea se va realiza în momentul în care două caractere recepţionate succesiv sunt identice cu cele doua caractere SYN memorate anterior, prin program, în interfaţă.

Terminalul SYNDET este folosit pentru a sincroniza receptorul, în cazul în care interfaţa foloseşte o sincronizare externă.

Modul de lucru HUNT este stabilit prin cuvântul de comandă (bitul D7) sau la funcţionarea interfeţei în regim sincron.

Receptorul trebuie activat, în vederea operării, prin bitul RxE (D2) al cuvântului de comandă. În acest caz el va furniza semnalul RxRDY activ, dacă are pregatit un caracter pentru a fi transmis pe magisttala DO—D7.

Pe baza figurii 6.9 se poate explica modul în care se efectuează recepţia caracterelor transmise serial.

La recepţie, caracterele se asamblează în RB (fig. 6.9, a), după care sunt transmise în paralel în registrul tampon RA (fig. 6.9, b). Conţinutul lui RA trebuie citit, de către umtatea centrală de prelucrare, în timp ce are loc receptia în RB a unui nou caracter. Dacă nu a avut loc citirea cuvântului lui EA, în timpul prestabilit, noul caracter din RB va fi încărcat în RA, peste vechiul caracter, semnalizându-se o eroare de ritm, prin poziţionarea corespunzătoare a bitului 4, în. registrul de stare.

Transmiţatorul primeşte datele îa paralel, de la unitatca centrală, le adaugă informaţia de cadru, le serializează şi le transmite la ieşirea TxD (fig. 6.9, c).

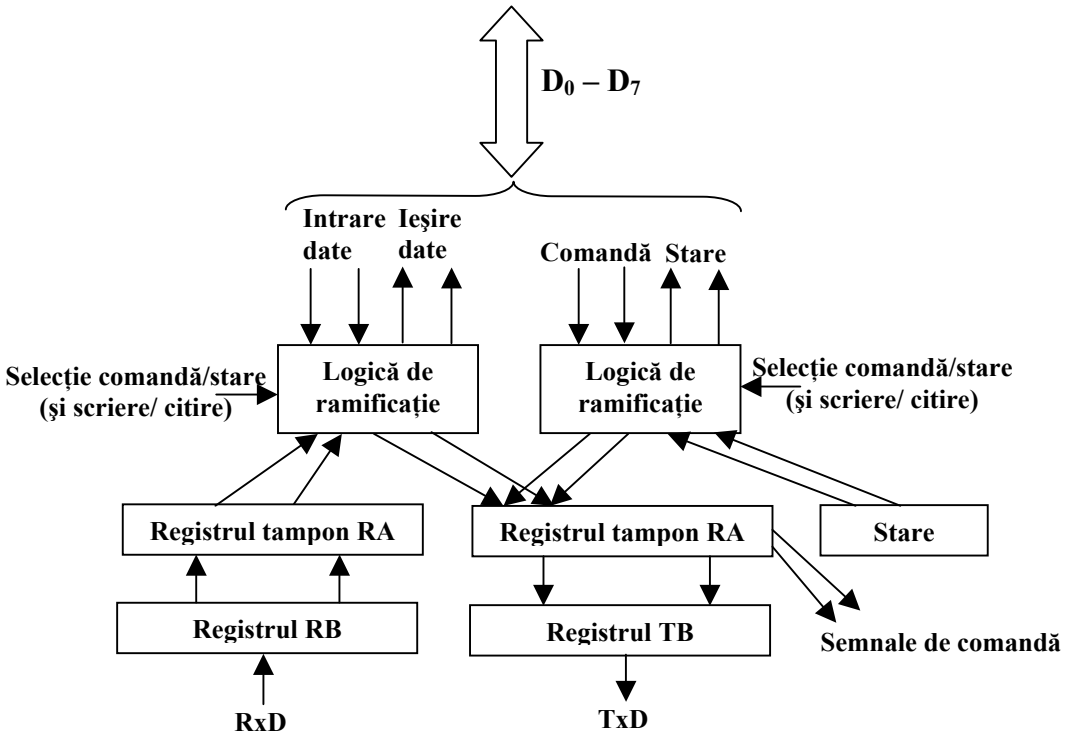


Figura 6.9. Recepția/Transmisia serială (a) Schema bloc.

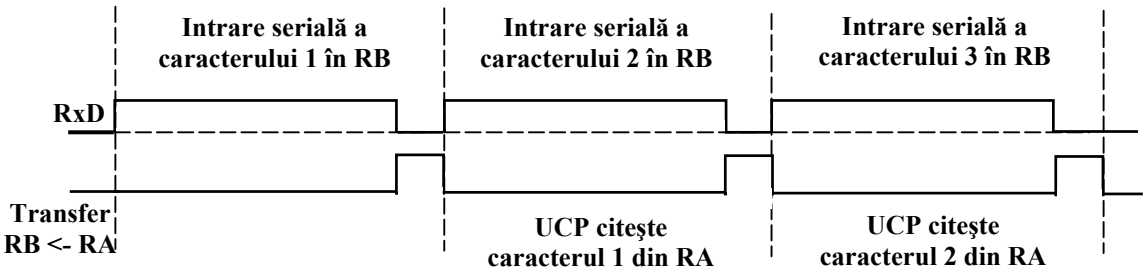


Figura 6.9. Recepția/Transmisia serială (b) Asamblarea caracterelor

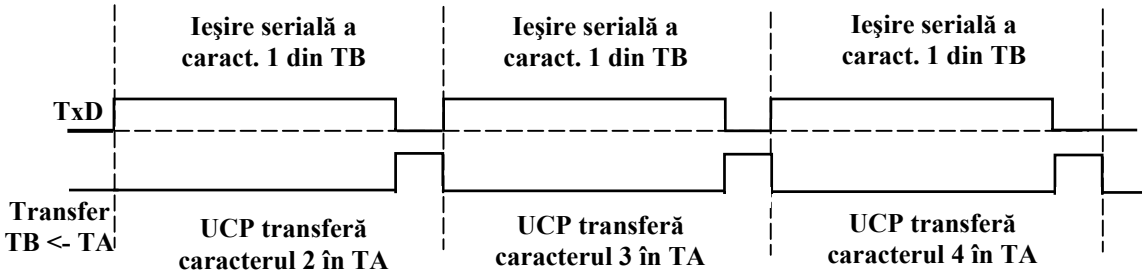


Figura 6.9. Recepția/Transmisia serială (c) Serializarea caracterelor la transmisie

La transmisia asincronă se atşează un bit de START şi, în funcţie de modul de programare, după cei 8 biţi de date, se mai adaugă un bit de paritate (pară sau impară) şi 1, 1/2 sau 2 biţi de STOP.

În cazul transmisiei sincrone nu se introduc biţi suplimentari decât în siutaţia în care calculatorul nu a furnizat caracterul către interfaţă. Aceasta, în mod automat, va insera cacacterul (caracterele) SYN, pentru a asigura o trasmisie continuă a biţilor. Interfaţa nu va iniţializ transmisia decât după ce a primit cel puţin un caracter, din partea unităţii centrale. Caracterele SYN sunt specificate prin software, în cadrul procedurii de iniţializare.

Atât în modul asincron cât şi în cel sincron transmisia este blocată cât timp intrările TxE şi CTS sunt la nivel ridicat. De asemenea, transmiţătorul poate genera informaţia BREAK, reprezentând o perioadă de octeţi, codificând SPACE, pe linia de transrnisie, pentru a întrerupe semnlul care se transmite, în cazul comunicaţiei duplex.

Datele transmise apar, la ieşire, la terminalul TxD, fiind controlate, ca viteză de transmisie de către semnalul de ceas TxC. Acesta poate fi furnizat, fie de unitatea centrală, fie de către o altă sursă. La transmisia asincronă datele suut emise la frecvenţa de 1/16 sau 1/64 din frecvenţa lui $\overline{TXC}$. Datele sunt strobate de tranziţia negativă a semnalului de ceas TxC.

Logica de transmisie generează două semnale de comandă: TxRDY şi TxE.

Semnalul TxRDY trece în 1 logic, atunci când conţinutul registrului TA a fost transferat în, TB, TA putând fi îacărcat cu un nou caracter. TxRDY ia valoarea logică 0, atunci când următorut octet de date este transferat în TA. Starea TxRDY este disponibiă la terminalul TxRDY atunci când interfaţa poate transmite (CTS=0 şi TxE = l). În registrul de stare al interfeţei, TxRDY este poziţionat în 1 logic, când registrul TA este vid, indiferent de valorile semnalelor CTS şi TxE.

Semnalul TxE este adus în 1 logic, când data din TB a fost serializată şi transmisă, rămânând în această stare până la încărcarea lui TB cu conţinutul lui TA.

La transmisia sincronă caracterele SYNC sunt încărcate în TB prin intermediul lui TA. Dacă se va genera o comandă, în. timp ce caracterul SYNC este îcărcat în TB, se va obţine un caracter eronat, care nu reprezintă nici cod de comandă, nici cod de SYNC.

Secţiunea referitoare la comanda modemului (fig, 6.6} generează semnalul, $\overline{DTR}$, indicând faptul că interfaţa este pregătită şi rccepţioneaza, semnalul $\overline{DSR}$, indicând faptul că modemul este pregatit. Semnalul $\overline{DTR}$ este generat pe baza poziţionării în 1 a bitului 2 din cuvântul de comanda, iar $\overline{DSR}$ poate fi testat prin examinarea bitului 7 din cuvâtul de stare.

Tot secţiunea de comandă a modemului recepţionează semnalul $\overline{CTS}$ şi generează semnalul $\overline{RTS}$, indicînd anularea în vederea transmisiei şi respeciv - ceirere în vederea transmisiei.

Comanda intrărilor/ieşirilor este asigurată prin logica de comandă citeşte/scrie, din figura 6.6, care decodifică semnalele de comandă, furnizate de unitatea centrală. În tabelul 6.1 sunt prezentate semnalele de comandă şi funcţiile îndeplinite de ele. Semnalele de comandă de citlre ($\overline{RD}$) si scrlere ($\overline{WR}$) pot apărea în orice moment, în raport cu intrarea de ceas, deoarece logica de comandă citeşte/scrie posedă circuite proprii de sincronizare.

TABELUL 6.1.

$\overline{CS}$	$C/\overline{D}$	$\overline{RD}$	$\overline{WR}$	Funcția
0	0	0	1	Unitatea centrală citeşte data de la interfață.
0	1	0	1	Unitatea centrală siteşte starea interfeței.
0	0	1	0	Unitatea centrală scrie data în interfață.
0	1	1	0	Unitatea centrală scrie o comandă în interfață.
1	x	x	x	Interfață neoperaşională (magistrala în starea de mare impedanță)

Blocul tampon de I/E conține tamponul de stare, tamponul de date pentru recepțe şi tamponul de date pentru emisie/comenzi, Tamponul de emisie date şi de comenzi este comun. Este important ca, în acest tampon, să nu se înscrie o comandă, dacă el deja conține o dată. În cazul în care TxRDY este pe nivelul coborât, nu se vor transfera date sau comenzi în interfață deoarece, în acest caz, se vor transmite date eronate.

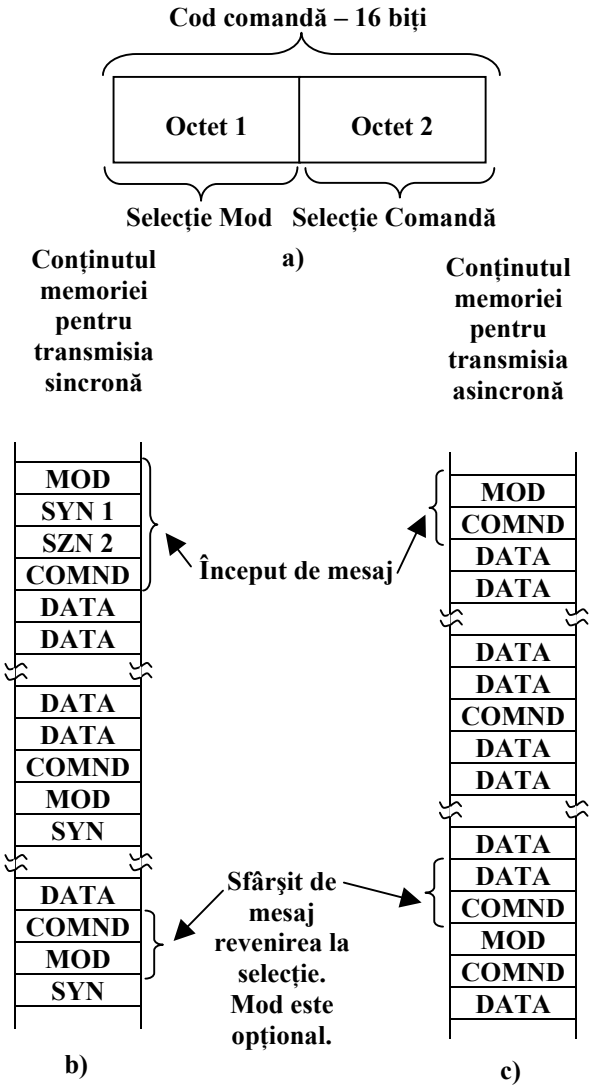


Figura 6.10. Cuvântul de comandă pentru interfața 8251.(a), conținutul zonelor de memorie la transmisiile: sincronă (b) şi asincronă (c).

Codurile de comandă şi stările interfeței sunt formate într-un registru de 16 biți şi respectiv citite, dintr-un registru de 8 biți. Aceste două registre sunt manipulate folosind aceeaşi adresă. Întrucât cele doua coduri de comandă sunt înscrise folosind aceeaşi adresă, ele sunt deosebite prin "Selecție Mod" şi respectiv "Selecție Comanda" (figura 6.10, a). La aplicarea tensiunii de alimentare interfața trece în „Selectie Mod". Dacă octetul de comandă, forțat în interfață, specifică modul sincron, logica interfeței aşteaptă unul sau doua coduri SYN (fig. 6.8). După recepționarea codului (codurilor) SYN, interfața trece automat în „Selecție Comandă". Această situație se menție pâna la aplicarea unui semnal RESET sau forțarea unei comenzi, care aduce interfața în "Selecție Mod". În figurile 6.10, b, c sunt prezentate conținuturile pentru transmisiile sincrona şi respectiv, asincronă.

În cazul în care două programe independente comandă aceeaşi interfață, poate apărea situaţa în care, o comandă internă de RESET este forţată spre interfață, în timp ce ea aşteaptă un caracter SYN. Acastă comandă va fi acceptată în calitate de SYN, fără ca operaţia internă de RESET să aibă loc. Pentru a evita o asemenea situaţie, comanda internă de RESET trebuie precedată de trei octeți de comandă, cu conținutul egal cu zero.

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0

Primul cuvânt de comandă

modul asincron — **modul sincron**

biţii 1,0 – rata de transmisie

00 – invalid; — **00 – semnalează modul sincron**

01 – rata de transmisie x1;

10 – rata de transmisie x 16

11 – rata de transmisie x 64

biţii 3,2 – lungimea caracterului

00 – caracter de 5 biți; — **00 – caracter de 5 biți;**

01 – caracter de 6 biți; — **01 – caracter de 6 biți;**

10 – caracter de 7 biți; — **10 – caracter de 7 biți;**

11 – carecter de 8 biți; — **11 – carecter de 8 biți;**

biţii 5,4 – controlul de paritate

x0 – dezactivat; — **x0 – dezactivat;**

01 – paritate para; — **01 – paritate para;**

10 – paritate impară; — **10 – paritate impară;**

biţii 7,6 – comanda cadrării — **Comanda SYN**

00 – invalid; — **x0 – SYN intern;**

01 – 1 bit de stop; — **x1 – SYN extern;**

10 – 1 1/2 biți de stop; — **0x – două caractere SYN;**

11 – 2 biți de stop. — **1x – un singur caracter SYN.**

a) **b)**

Al doilea cuvânt de comandă

bit 0 – transmisie activată TxEN

0 – dezactivare transmisie;

1 – activare transmisie;

bit 1 – terminal de date pregătit DTR

în cazul unui nivel ridicat se va forța în zero ieşirea DTR;

bit 2 – recepție activată RxE

0 – dezactivează RxRDY;

1 – activează RxRDY;

bit 3 – transmite caracterul BREAK:SBRK

0 – operare normală;

1 – forțare TxD la nivel coborât;

bit 4 – anulare ER

1 – anularea tuturor erorilor din registrul de stare (PE,OE,FE);

bit 5 – cerere de transmisie RTS

nivel ridicat – se forțează ieşirea $\overline{RTS}$ în zero;

bit 6 – RESET intern IR

nivel ridicat – se forțează interfața în mod instrucțiune;

bit 7 – intrare în modul HUNT:EH

1 – intrare în modul HUNT

Figura 6.11. Semnificația cuvintelor de comandă. Primul cuvânt de comandă în modul (a) asincron, (b) sincron, (c) al doilea cuvânt de comandă.

Interpretarea primului cuvânt de comandă, de către logica din interfață, este prezentată în figura 6.11. Se deosebesc două interpretări ale acestui cuvânt, în funcție de modul de lucru asincron (fig. 6.11, a) şi respectiv sincron (fig. 6.11, b). Primul cuvânt de comandă, transmis interfeței, defineşte modul de operare, în timp ce al doilea cuvânt de comandă (Selectie Comandi) defineaşe acțiuni instantanee, dupa cum se arata în figura 6.11, c. Biții din acest cuvânt de comandă sunt poziționați în unu sau zero, în general, sub controlul programului, Trebuie menționat faptul ca biții ER, IR si EH sunt anulați în următoarele condiții: EH — când sunt anulate erorile (PE, OE, FE) din registrul de stare, IR — când interfața se află în modul instrucțiune şi EH — când a fost detctat caracterul SYN.

Comanda RxE (Recepție Activată), din cel de-al doilea cuvânt de comandă, nu afectează logica de recepție, ci numai semnalul RxRDY, Dacă, în continuare, datele sunt recepționate de către interfață, după primirea acestei comenzi, ele vor fi asamblate în RA şi RB, fară a se genera semnalul RxRDY, spre unitatea centrală, pentru a semnala caracterele asamblate. În cazul în care se generează din nou comanda RxE, este posibilă citirea caracterelcr asamblate anterior. Pentru a evita această situație, caracterele respective trebuie citite şi înlăturate imediat ce s-a activat RxEN, în modul asincron, sau EH, în modul sincron.

Este important de menționat faptul că semnalele de comandă $\overline{DTR}$ şi $\overline{RTS}$ fiind coatrolate prin biții 1 şi respectiv 3, din cuvântul de comandă, necesită, pentru o funcționare corectă a interfeței, ca la modificarea lor, ceilalți biți să nu fie schimbați sau să corespundă comenzilor necesare.

La citirea stării interfeței seriale programabile biții cuvântului respectiv sunt interpretați conform figurii 6.12. Bții de eroare: (PE - eroare de paritate, OE - eroare de ritm, FE - eroare de cadrare) se poziționează în unu la apariția condițiilor de eroare respective. Corectarea erorilor se face prin program.

Cuvântul de stare

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
DSR	SYNDET	FE	OE	PE	TxE	RXRDY	TXRDY

biții 0, 1, 2, 6, 7 – condiții pentru semnalele indicate: TxRDY, RxRDY, TxE, SYNDET, DSR;

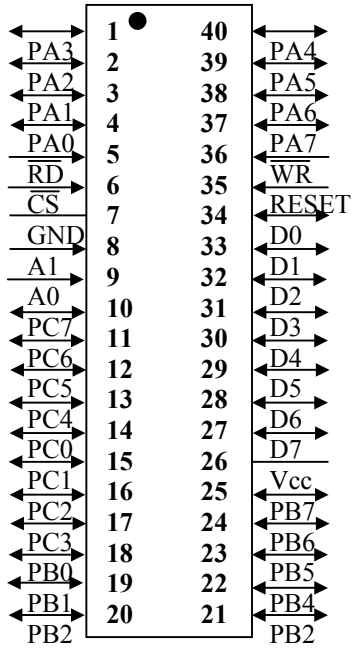
bitul 3 – eroare de paritate PE
1 – eroare de paritate;
bitul 4 – eroare de ritm OE
1 – RA nu a fost citit înainte ca RB să fie încărcat cu un nou caracter;
bitul 3 – eroare de cadrare (în modul asincron)
1 – nu s-a detectat un bit corect de stop la sfârşitul fiecărui caracter.

Figura 6.12. Structura cuvântului de stare.

6.3. Interfața logică programabilă 8255

Circuitul 8255 realizat în tehnologia NMOS, pe o pastilă cu 40 de terminale, ale căror semnificații sunt prezentate în figura 6.13, reprezintă o interfață programabilă de intrare/ieşire (I/E).

Deşi a fost proiectat ca un circuit din familia microprocesorului 8080, el poate fi utilizat şi la alte microprocesoare.



$D_0 – D_7$	- magistrală de date (bidirecțional);
$PA_0 – PA_7$	- terminale I/E Port A (bidirecțional);
$PB_0 – PB_7$	- terminale I/E Port B (bidirecțional);
$PC_0 – PC_7$	- terminale I/E Port C (bidirecțional);
RD	- comandă citire (intrare);
WR	- comandă scriere (intrare);
RESET	- comandă RESET (intrare);
CS	- selecție circuit (intrare);
A_0, A_1	- intrări selecție porturi (intrări);

Figura 6.13. Terminalele circuitului 8255 şi semnificațiile lor.

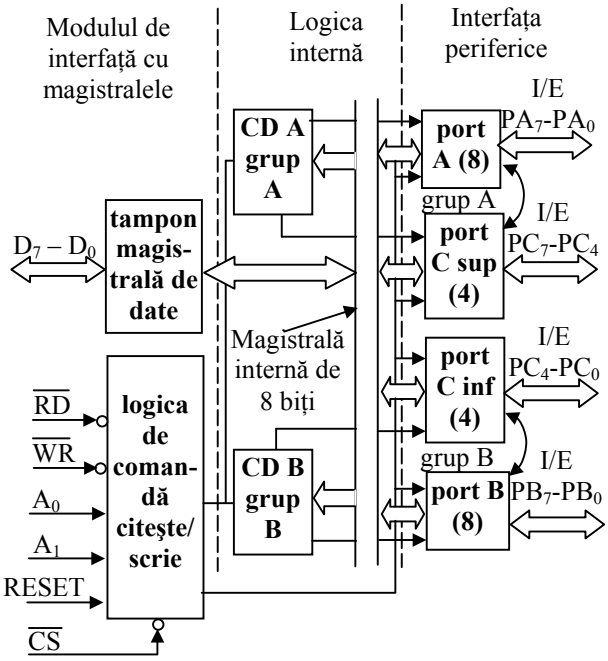


Figura 6.14. Schema bloc.

Circuitul posedă 24 de terminale de I/E, care pot fi configurate ca unul, două sau trei porturi de I/E. Dintre cele 24 de terminale, 16 sunt prevăzute cu posibilități de memorare (latch-uri), iar celelalte 8 posedă tampoane (buffer-e).

Alimentarea circuitului se face de la o sursă de 5V; toate intrările şi ieşirile sunt compatibile TTL.

În figura 6.14 se prezintă schema bloc a circuitului 8255, care constă din: modulul de interfață cu magistrala microprocesorului, interfața cu periferia şi logica internă.

Modulul de interfață cu magistrala microprocesorului conține tamponul bidirecțional, care face legătura între magistrala bidirecționată de date a microprocesorului şi magistrala internă a interfeței paralele, precum şi logica de comandă pentru operațiile de scriere/citire. Aceasta din urmă are ca intrări semnalele de RESET, selecție a circuitului ($\overline{CS}$), scriere ($\overline{WR}$), citire ($\overline{RD}$) şi selecție a porturilor şi a registrului cuvântului de comandă (A_0, A_1).

Semnalul RESET anulează conținuturile tuturor registrelor din circuitul 8255.

Semnalul CS activează comunicația între magistrala microprocesorului şi circuitul 8255.

TABELUL 6.2. Selecţia porturilor ABC şi a registrului cuvântului de comandă.

$\overline{CS}$	A_0	A_1	Se selectează
0	0	0	Portul I/E A
0	0	0	Portul I/E B
0	1	0	Portul I/E C
0	1	1	Registrul cuvântului de comandă (numai pentru scriere)
1	x	x	Nu se selectează 8255

În tabelul 6.2 se prezină efectul semnalelor $\overline{CS}$, A_0 şi A_1 privind selectarea porturilor şi a registrului cuvântului de comandă. În cazul în care se foloseşte o tehnică simplă de selecţie, numită selecţia liniară (o variantă a selecţiei redundante), adresa de 8 biţi a unei instrucţuni IN sau OUT, se poate utiliza ca în tabelul 6.3. Pe baza selecţei liniare, biţii A_0, A_1, din adresă, se folosesc pentru selecţia porturilor, din cele 6 circuite 8255, selectabile cu ajutorul rangurilor A_7 … A_2.

TABELUL 6.3. Selecţia liniară a circuitului 8255.

A_7	A_6	A_5	A_4	A_3	A_2	**Selecţie ($\overline{CS}$)**	A_1	A_0	**Selecţie**
1	1	1	1	1	0	Circuit 1	0	0	Port A
1	1	1	1	0	1	Circuit 2	0	1	Port B
1	1	1	0	1	1	Circuit 3	1	0	Port C
1	1	0	1	1	1	Circuit 4	1	1	Registrul cuvântului de comandă
1	0	1	1	1	1	Circuit 5			
0	1	1	1	1	1	Circuit 6			

Interfaţa cu periferia conţine 24 de linii de interfaţă, tampoane şi logica de comandă. Caracteristicile şi funcţiile liniilor din interfaţă sunt derminate de modul de operare selectat, prin program. Sub controlul software-ului pot fi selectate trei moduri de lucru diferite, pentru interfaţa programabila 8255 (figura 6.15, a, b, c).

Modul 0, denumit modul de bază de intrare/ieşire, asigură mai multe posibilităţi:

- două porturi de câte 8 biţi (PA_7 … PA_0, PB_7 … PB_0);
- două porturi de câte 4 biţi (PC_3 … PC_0, PC_7 … PC_4), cu capabilitata de poziţionare individuală în unu sau zero.

Porturile folosite pentru ieşire sunt prevăzute cu elemente bistabile de memorare, porturile folosite pentru intrări nu dispun de elemente de memorare (fig. 6.15, a).

Modul 1 asigură posibilităţi de strobare pentru intrare/ieşire. Astfel, unul sau două porturi, organizate pe 11 biţi, conţin 8 biţi de date, 3 biţi de comandă şi logica de suport pentru întreruperi.

Oricare port poate fi folosit pentru intrare sau ieşire. Daca în *Modul 1* se foloseşte un singur port, ceilalţi 13 biţi pot fi configuraţi în *Modul 0*. Dacă în *Modul 1* sunt programate două porturi, cei doi biţi rămaşi pot fi utilizaţi pentru intrare sau ieşire cu capabilitate de poziţionare în unu sau zero (figura 6.15, b).

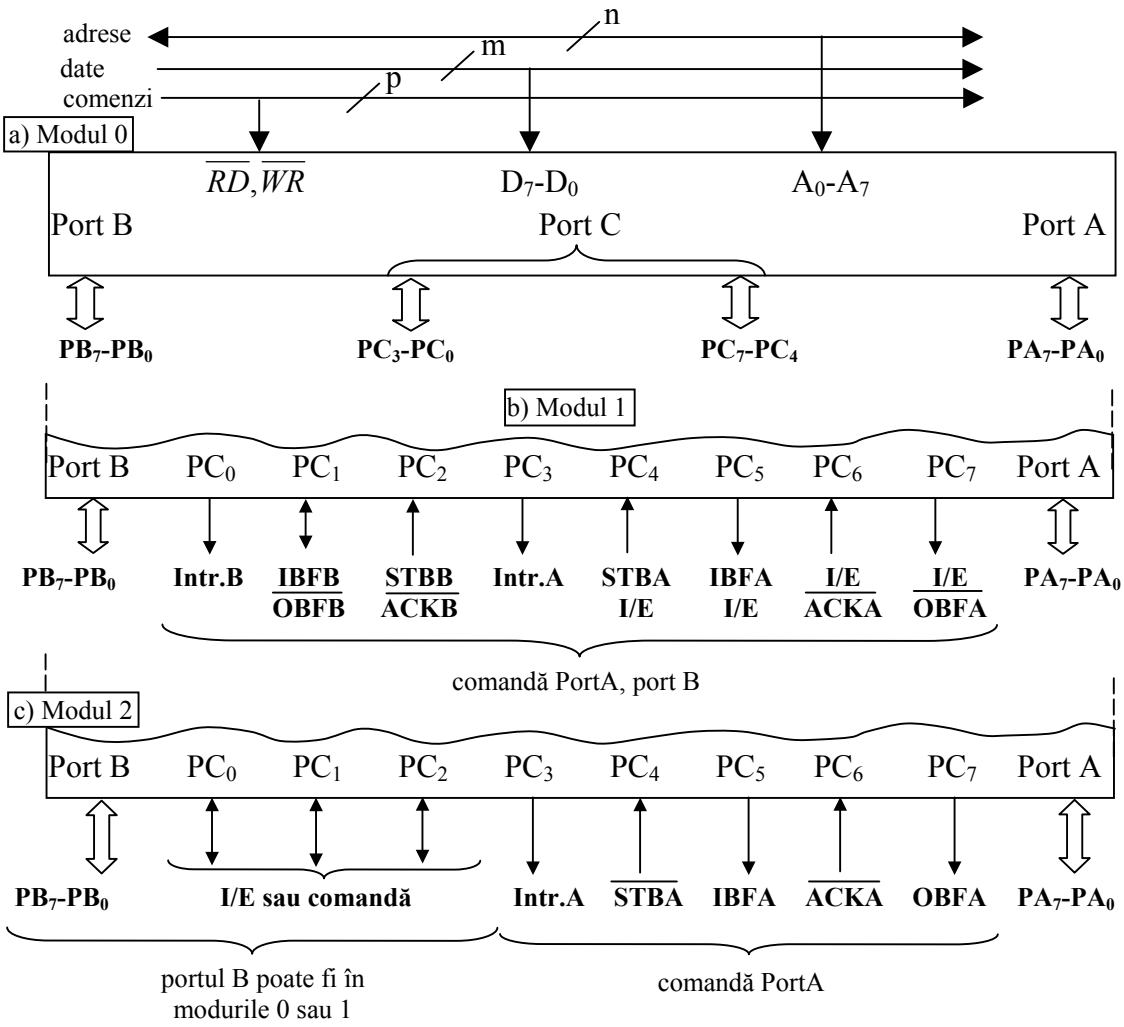


Figura 6.15. Modurile de lucru pentru interfața programabilă 8255: (a) modul 0, (b) modul 1, (c) modul 2,

În figura 6.16 sunt date configurațiile porturilor A, B şi C ale interfeței 8255 şi cuvintele de comandă corespunzatoare, pentru operația de intrare (a) şi operația de ieşire (b).

Pentru intrare, semnalele de comandă au semnificațiile de mai jos:

- $\overline{\text{STBA(B)}}$, strob de intrare, activ pe nivel coborât, încarcă data în registrul de intrare;
- IBFA(B), indicator tampon de intrare încărcat, este activ pe nivel ridicat. El este activat de $\overline{\text{STBA(B)}}$ - pe nivel coborât şi dezactivat de către frontul crescător al comenzii $\overline{\text{RD}}$;
- INTRA(B), cerere de întrerupere pentru microprocesor, activă pe nivelul ridicat, atunci când datele sunt deja în registrul de intrare al interfeței 8255. INTRA(B) este activat dacă STBA(A) este pe nivel ridicat şi dacă IBFA(B) şi INTEA(B) sunt, de asemenea, active. INTRA(B) este dezactivat pe f rontul căzător al comenzii $\overline{\text{RD}}$;

- INTEA(B), bistabile interne asociate cu intreruperile, sunt activate/dezactivate prin terminalele PC_4/PC_2.

Pentru iesire semnalele de comanda au urmatoarele semnificatii:

- $\overline{OBFA(B)}$, indicator tampon de iesire incarcat, este activat pe nivel coborit de catre Jrontul negativ al comenzii WR si dezactivat pe frontul negativ al semnalului ACKA(B);
- $\overline{ACKA(B)}$, acceptare date de catre periferic, este furnizat când datele au fost preluate de periferic. Este activ pe nivel coborât;
- INTRA(B), cerere de întrerupere pentru microprocesor, activă pe nivel ridicat, specifică faptul că echipamentul periferic a preluat data. INTRA(B) este activat când $\overline{ACKA(B)}$ este pe nivel ridicat, $\overline{OBFA(B)}$ este pe nivel ridicat şi INTEA(B) este, de asemenea, pe nivel ridicat. Semnalul este dezactivat pe frontul crescător al comenzii $\overline{WR}$;
- INTEA(B), bistabile interne asociate cu intreruperile, sunt activate/dezactivate prin terminalele PC_6/PC_2.

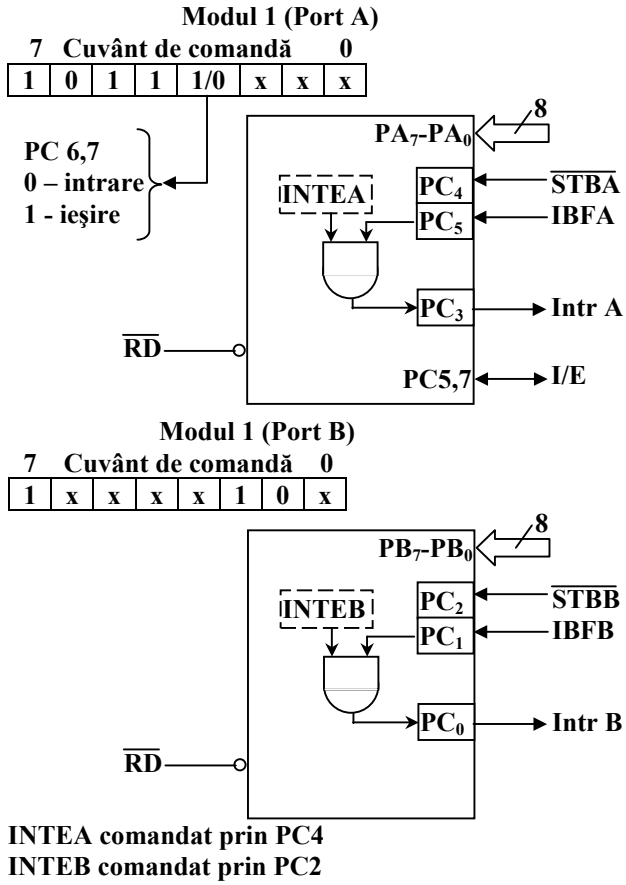


Figura 6.16. a. Configurațiile porturilor şi cuvintele de comandă corespunzătoare pentru operația de intrare.

După cum se constată, portul C este folosit pentru manipularea cuvintelor de comanda/stare, pentrti porturile A, B.

Dacă portul A (sau portul B) a fost programat ca port de intrare, logica externă trebuie să indice faptul că o nouă dată a fost plasată la portul de intrare. Această se realizează prin aplicarea unor semnale de strob $\overline{STBA}$, $\overline{STBB}$ la intrările PC_4 şi respectiv PC_2. Interfața 8255 va aduce terminalul IBFA (IBFB) la un nivel ridicat, atunci când data este încărcată în portul de intrare şi se menține la acest nivel, cât timp portul conține data respectivă. Semnalul IBFA(IBFB) este anulat la dezactivarea comenzii de citire ($\overline{RD}$), emisă de microprocesor, indicând citirea datelor din portul de intrare. Pentru a specifica microprocesorului faptul că data este stabilă în portul de intrare, interfața 8255 generează un semnal

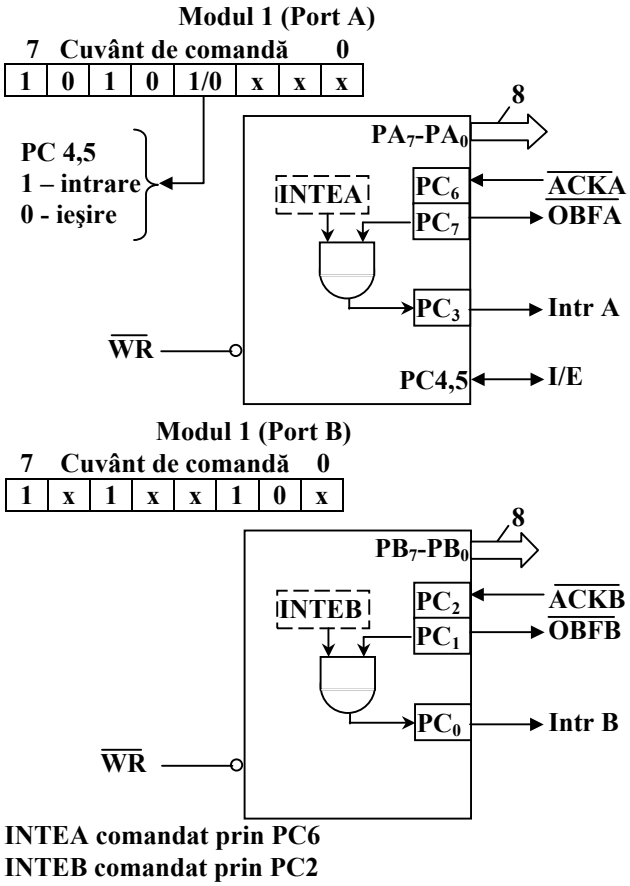


Figura 6.16. b. Configuraţiile porturilor şi cuvintele de comandă corespunzătoare pentru operaţia de ieşire.

de întrerupere (INTR) pe frontul crescător al semnalului de strob ($\overline{STBA}$, $\overline{STBB}$). Acest semnal, INTR, este dezactivat pe frontul căzător al comenzii $\overline{RD}$. Terminalele PC_7 şi PC_6 nu manipulează semnale de comandă privind intrarea şi pot fi utilizate pentru a transfera bidirecţional date.

Dacă portul A (sau portul B) a fost programat ca port de ieşire, în momentul în care microprocesorul îl încarcă cu o dată, interfaţa 8255 generează un semnal de comanda $\overline{OBFA}$ ($\overline{OBFB}$), activ pe nivel coborât. Logica externă poate fi astfel informată asupra posibilităţii de preluare a datelor din portul de ieşire corespunzator. Logica externă semnalează interfeţei faptul că a preluat informaţia, din portul de ieşire, prin activarea semnalului ACKA(ACKB). Pentru ca microprocesorul să poată plasa o nouă dată în portul de ieşire, el trebuie să aştepte un semnal de întrerupere INTRA, (INTRB), care este activat pe frontul crescător al semnalului $\overline{ACKA}$ ($\overline{ACKB}$). Terminable PC_4 şi PC_5 nu manipulează semnale de comandă în cadrul operaţiei de ieşire, în *Modul 1*. Ele pot fi folosite pentru transferul bidirecţional de date.

Modul 2, definit ca magistrală de I/E, bidirecţională strobată, asigură posibilităţi de comunicare cu un periferic, pe o magistrală de 8 biţi. Semnalele de dialog sunt folosite pentru a menţine disciplina de transfer a informaţiei pe magistrală.

Modul 2 este folosit cu grupul A de terminale, care se referă la portul A (PA_0 , PA_1) şi la biţii PC_3 … PC_7, din portul C (figura 6.14).

Configurarea şi cuvântul de comandă ale circuitului în acest mod sunt prezentate în figura 6.17. Semnificaţiile semnalelor sunt prezentate mai jos:

- INTRA, cererea de întrerupere, activă pe nivel ridicat şi emisă de interfaţa 8255, la termiualul PC_3, întrerupe microprocesorul atât pentru operaţia de intrare, cât şi pentru cea de ieşire;

- $\overline{\text{OBFA}}$, în cadrul unei operații de ieşire, va fi activ pe nivel coborât, pentru a indica faptul că microprocesorul a înscris date în portul de ieşire A ;
- $\overline{\text{ACK}}$, în cadrul unei operații de ieşire, activ pe nivel coborât, va comanda tamponul de ieşire, cu trei stări, al portului A, pentru a furniza data în exterior. $\overline{\text{ACK}}$, la nivel ridicat, va aduce tamponul de ieşire în starea de mare impedanță;
- INTE1, bistabil intern de activare/dezactivare a întreruperilor, asociat cu semnalul OBF, este poziționat în unu/zero, prin PC_6;
- $\overline{\text{STBA}}$, în cadrul operației de intrare, activ pe nivel coborât, va forța data de intrare în registrul portului de intrare A (PA);
- IBFA, în cadrul operației de intrare, activ pe nivel ridicat, va specifica faptul că data a fost încărcată în registrul portului de intrare A;
- INTE2, bistabil intern de activare/dezactivare a întreruperilor, asociat cu semnalul IBF, este poziționat în unu/zero, prin PC_4.

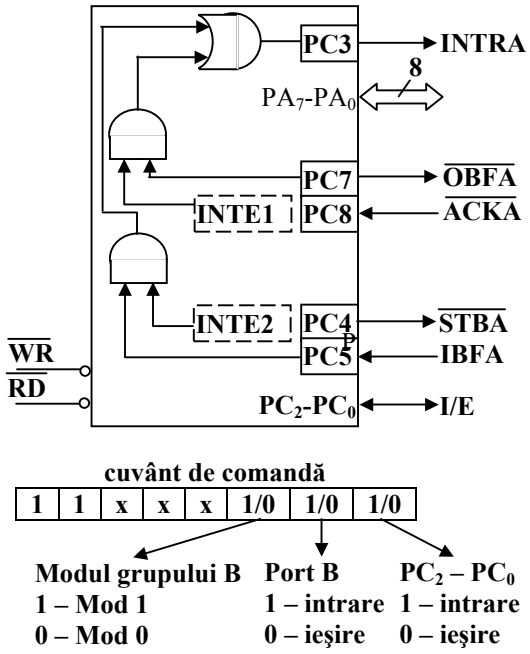


Figura 6.17. Configurația terminalelor interfeței 8255 în modul 2 şi cuvântul de comandă corespunzător.

Operația de intrare, în *Modul 2* are loc prin furnizarea dalelor de către periferic, la intrarea portului A şi foțarea lor în portul respeciv, folosind strobul STBA. La încărcarea registrului portului A se generează semnalul IBFA, activ pe nivel ridicat. Semnalul de întrerupere pentru microprocesor, INTRA, este generat când semnalele $\overline{\text{RD}}$, $\overline{\text{STBA}}$ sunt inactive şi IBFA activ. INTRA este dezactivat prin generarea comenzii de citire $\overline{\text{RD}}$, de către microprocesor. Semnalul IBFA este dezactivat odatS cu dezactivarea eomenzii $\overline{\text{RD}}$.

Operația de ieşire, în *Modul 2* se realizează prin furnizarea datelor la intrarea D_7 … D_0 şi a semnalului $\overline{\text{WR}}$, de către microprocesor. Datele se transferă în registrul portului de ieşire A, fapt care va conduce la activarea semnalului $\overline{\text{OBFA}}$. Acest semnal poate fi testat de către echipamentul periferic, pentru a putea stabili disponibilitatea datelor la portul de ieşire. Preluarea lor de către echipamentul periferic este asigurată prin semnalul $\overline{\text{ACKA}}$, activ pe nivel coborât. Semnalul $\overline{\text{ACKA}}$ dezactivează $\overline{\text{OBFA}}$ şi activează cererea de intrerupere INTRA, pentru a anunța microprocesorul că datele au fost preluate de către periferic.

Logica internă de selecţie gestionează transferul datelor şi al informaţiei de comanda pe magistrala internă. Prin selecţia porturilor A, B şi C se efectuează operaţii de I/E. Logica internă va asigura transferul datelor între magistrala microprocesonilui şi porturile de I/E ale interfeţei 8255.

Configuraţia funcţională a fiecărui port şi poziţionarea în unu sau zero a biţilor portului C sunt controlate prin software, folosind cuvinte de comandă corespunzatoare. Selectarea registrului de comandă. şi încărcarea lui cu un cuvânt de comandă permit logicii interne să efectueze operaţiile descrise în cadrul acestui cuvânt. Cuvântul de comandă conţine un câmp de cod de operaţie, care defineşte modul de operare sau poziţionarea unor biţi ai portului C, în funcţie de valoarea 1, respectiv 0 a bitului 7, din cuvlntul de comandă.

Definirea modului de operare se realizează prin încărcarea în interfaţa 8255 a unui cuvânt de comandă cu bitul 7 egal cu unu. În figura 6.18 se prezintă codificarea cuvântului de comandă pentru a specifica configuraţia celor 24 de linii de legatură a interfeţei programabile cu periferia. Astfel, pot fi specificate independent modurile porturilor A şi B, în timp ce portul C poate fi tratat independent sau separat ca două porturi de cite 4 biţi solicitate de modurile de definire ale porturilor A şi B.

Cuvântul de comandă

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	1/0	1/0	1/0	1/0	1/0	1/0	1/0

Grup B
bitul 0 – port C ($PC_3 – PC_0$);
1 – intrare;
0 – ieşire;
bitul 1 – port B;
1 – intrare;
0 – ieşire;
bitul 2 – Selecţie Mod;
1 – Mod 0;
0 – Mod 1;
Grup A
bitul 3 – port C ($PC_7 – PC_4$);
1 – intrare;
0 – ieşire;
bitul 4 – port A;
1 – intrare;
0 – ieşire;
biţii 5,6 – Selecţie Mod;
00 – Mod 0;
01 – Mod 1;
1x – Mod 2.

Figura 6.18. Structura cuvântului de comandă pentru definirea modului de operare al interfeţei 8255.

Încărcarea unui cuvânt de comandă cu bitul 7 egal cu zero (figura 6.19) permite comanda individuală a biţilor portului C, prin poziţionarea lor în zero sau unu. Selecţia

bitului din portul C se realizează prin biţii 3 … l, din cuvântul de comandă. Bitul 0, al cuvântului de comandă definite forţază în zero sau unu a bitului selectat, din portul C. Biţii 6 … 4 ai cuvântului de comandă nu sunt folosiţi.

Cuvântul de comandă

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	0	0	0	1/0	1/0	1/0	1/0

bitul 0 – bitul selectat se poziţionează în unu / zero;
1 – poziţionare în unu;
0 – poziţionare în zero;
biţii 3,2,1 – selecţie bit;
000 – bit 0;
001 – bit 1;
010 – bit 2;
011 – bit 3;
100 – bit 4;
101 – bit 5;
110 – bit 6;
111 – bit 7;

Figura 6.19. Structura cuvântului de comandă pentru poziţionarea în unu/zero a biţilor portului C.

6.4. Interfaţa USB (Universal Serial Bus)

6.4.1. Introducere

Studiul portului USB poate părea puţin înspăimntător datorită specificaţiilor USB care au 650 de pagini dar şi listei impresionante de standarde asociate (USB Class Standards). Dintre standardele asociate face parte şi HID Class Specifcation care detaliază operaţiile obinuite cu dispozitivele (tastatură, maus etc.) şi care provine din HID (Human Interface Devices) Class. Dacă proiectaţi un port USB (USB Host) atunci va trebui să alegeţi între trei standarde (Hot Controller Interface Standard). Nici unul dintre aceste standarde nu este specificat în USB 2.0.

În continuare se vor prezenta succint informaţiile esenţiale legate de interfaţa USB. Standardele interfeţei USB cât şi specificaţiile conexe trebuie studiate în funcţie de omeniul activităţii desfăşurate (proiectare hardware sau software).

Mai jos (tabelul 6.4) este prezentată structura standardului USB 2.0 şi informaţiile conţinute de capitolele acestuia precum şi recomandări privind parcurgerea informaţiilor pentru înţelegerea interfeţei USB.

TABELUL 6.4.

Capitol	Nume	Descriere	Nr. de pagini
1	Introducere	Conține scopul şi motivația USB. Informația cea mai importantă prezentată în acest capitol este referirea la Universal Serial Bus Device Class Specifications. Nu este necesară citirea acestui capitol.	2
2	Termeni şi notații	Capitol explicativ necesar oricărui standard.	8
3	Informații de bază	Capitolul prezintă scopul interfeței USB care este o interfață Plug and Play simplă (pentru utilizator nu şi pentru proiectant). Sunt prezentate noțiunile de viteză scăzută, completă şi înaltă (Low, Full and High Speed) şi o listă a funcțiilor interfeței. Nici acest capitol nu este necesar a fi citit.	4
4	Structura arhitecturii	Din acest punct trebuie început studiul. Capitolul furnizează noțiunile de bază a sistemului USB: topologia, viteza de transfer a datelor, tipurile fluxurilor de date, specificații electrice de bază etc.	10
5	Modelul fluxului de date USB	În acest capitol se prezintă caracteristicile fluxlui de date al portului serial universal (Universal Serial Bus). Sunt prezentați termeni ca: endpoints şi pipes şi este analizat fiecare tip de flux de date (Control, Interrupt, Isochronous şi Bulk). Acest capitol poate fi destul de dificil începătorilor dar trebuie citit dacă se doreşte cunoaşterea tipurilor de transfer a interfeței USB şi a proprietăților acestora.	60
6	Caracteristici mecanice	În acest capitol se prezintă doi conectori USB standard. Este prezentat conectorul de tip A destinat transferului în aval (downstream) şi conectorul de tip B sestinat transferului în amonte (upstream). În acest fel este imposibil să se conecteze un cablu între două porturi upstream. Toate cablurile detaşabile trebuie să fie de viteză full/high pe când cablurile de viteză scăzută (Low) trebuie conectate (fixate) la aplicație. În afara de faptul că trebuie să aruncați o privire asupra conectorilor, acest capitol poate fi sărit (dacă nu intenționați să fabricați conectori sau cabluri USB). Proiectanții PCB pot găsi aici dimensiunile standard pentru conectori.	33

TABELUL 6.4. (continuare)

7	Caracteristici electrice	Capitolul 7 prezintă semnalele electrice la nivelul de bază, impedanţa de linie, timpii de creştere şi de descreştere, specificaţiile driver/receptor şi codificarea la nivel de bit, structura biţilor etc. Cea mai importantă parte a acestui capitol este identificarea vitezei dispozitivului prin utilizarea unui rezistor pentru interferenţa liniei de date sau a magistralei dispozitivelor alimentate versus dispozitivelor autoalimentate. În afară de cazul când proiectaţi trranscivere USB la nivel de bază puteţi trece superficial prin acest capitol. Schemele corecte a dispozitivelor USB trebuie să arate ce valoare a rezistorului terminal este necesară pentru adaptarea impedanţei magistralei.	75
8	Protocolul	Se explorează protocolul USB prezentându-se pachetele la nivel de bit şi discutându-se despre sync, pid, address, endpoint, CRC fields. Cei mai mulţi dezvoltatori nu trebuie să cunoască aceste detalii deoarece circuitul integrat rezolvă problema protocolului. În orce caz, înţelegerea modului de raportare a stării şi dialogul între componente este necesară.	45
9	USB Device Frame Work	Acesta este capitolul cel mai folosit deoarece descrie enumerarea pe magistrală (bus enumeration) şi cererea codurilor (request codes): set, address, get descriptor etc, care reperezinta noţiunile cele mai folosite din protocolul USB. Capitolul trebuie citit în detaliu.	
10	USB Host Hardware and Software	Se descrie gazda (host) ceea ce presupune: generarea frame şi microframe, cerinţiele host controller şi modelul driver universal serial bus. Dacă nu se proiectază un Host, capitolul poate fi sărit.	23
11	Hub Specification	Se detaliază modul de lucru al USB hub (hub – punct cetral): configurarea hub, split transactions, standard descriptors for hub class etc. Dacă nu se proiectază un Hub, capitolul poate fi sărit.	143

Pentru realizarea driver-elor (software) pentru perifericele USB, atunci trebuie citite numai capitolele:

- 4 - Architectural Overview
- 5 - USB Data Flow Model
- 9 - USB Device Frame Work, and
- 10 - USB Host Hardware and Software.

Proiectarea hardware (electronica) se poate face citind capitolele:

- 4 - Architectural Overview
- 5 - USB Data Flow Model
- 6 - Mechanical, and
- 7 - Electrical.

6.4.2. Prezentarea Universal Serial Bus

Standardul USB 1.1 a fost suficient de complex înainte ca High Speed să fie introdus în USB 2.0. În scopul înțelegerii principiilor fundamentale a USB se vor omite noțiunile legate de dispozitivele High Speed.

USB versiunea 1.1 suportă două viteze, viteza completă (full) de 12 Mbiți/s şi viteza scăzută (low) de 1,5 Mbiți/s. Modul de 1,5 Mbiți/s datorită faptului că este mai scăzut este mai puțin influențat de perturbațiile electromagnetice (EMI) şi acest lucru reduce costul mărgelelor de ferită şi a calității componentelor. De exemplu, cristalele de cuarț pot fi înlocuite reyonatori mai ieftini. USB 2.0 utilizat la majoritatea calculatoarelor de birou are şi viteza înaltă de 480 Mbiți/s (high) şi completează Firewire Serial Bus.

6.4.2.1. Vitezele USB

- High Speed – 480 Mbits/s
- Full Speed – 12 Mbits/s
- Low Speed – 1.5 Mbits/s

Magistrala serială universală (Universal Serial Bus) este de tip host controlled şi nu poate fi decât un host pe magistrală. De asemenea specificațiile nu permit nici un fel de aranjament multimaster. În orice caz, specificațiile On-The-Go care reprezintă o adaptare a standarului USB 2.0 a introdus Host Negotiation Protocol care permite la două dispozitive să negocieze rolul de host. Acest lucru se face în scopul limitării la o singură conexiune punct la punct cum ar fi cu un telefon mobil sau un personal oragniser şi nu hub multiple sau configurații multiple ale dispozitivelor desktop. USB host este răspunzător de toate tranzcțiile şi perogramarea lățimii de bandă. Datele pot fi transmise prin diferite metode utilizând token-based protocol.

În opinia multor specialişti, topologia magistralei USB este limitativă. Una dintre intențiile inițiale a fost ca USB să reducă numărul de cabluri de conexiune. Proiectanții firmei Apple susțin că ideea provine de la tehnologia Apple Desktop Bus prin care tastatura mausul şi alte câteva periferice pot fi conectate împreună în serie (daisy chained) folosind un singur cablu.

În orice cazn tehnologia USB foloseşte topologia stea extinsă, similară celei 10BaseT Ethernet. Aceasta impune utilizarea unui hub undeva ceea ce duce la creşterea preţului de cost, creşterea numărului de cutii pe birou şi creşterea numărului de cabluri. Multe dispozitive au hub USB integrate. De exemplu tastatura poate conţine un hub conectat la calculator. Mausul şi alte dispozitive cum ar fi camera digitală pot fi conectate uşor în locul tastaturii (sau: după tastatură – înseriat cu aceasta). Monitoarele reprezintă un alt periferic dintr-o lungă listă de periferice care au hub-uri incluse.

La topologia stea extinsă, spre deosebire de conexiunea daisy chain, dispozitivele conectate au anumite beneficii. Mai întâi alimentarea fiecărui dispozitiv poate fi monitorizată şi chiar oprită dacă se produce o anumită condiţie fără ca celelalte dispozitive USB să fie întrerupte. Toate dispozitivele cu viteza high, full sau low pot fi acceptate, prin filtrarea de către hub a tranzacţiilor cu viteză high şi full pe care dispozitivele de viteză scăzută nu le vor recepţiona.

Pâna la 127 de dispozitive pot fi conectate la orice magistrala USB la orice moment. Dacă sunt necesare mai multe dispozitive se adaugă un nou port/host. La început, primele USB host aveau două porturi; cei mai mulţi producători au considerat acest lucru restrictiv şi au început să introducă 4 sau 5 porturi (port host card) cu un port intern pentru hard disk. Primele host-uri aveau un singur controler USB şi atunci cele două porturi împărţeau aceeaşi bandă USB disponibilă. O dată cu creşterea necesarului de bandă vedem cartele multiport cu două sau mai multe controlere fiecare cu canalul său.

Controlerele host USB au propriile lor specificaţii. La USB 1.1 sunt două specificaţii Host Controller Interface: UHCI (Universal Host Controller Interface) dezvoltată de Intel care a pus o încărcătură software (Microsoft) mai mare permiţând un hardware mai ieftin şi OHCI (Open Host Controller Interface) dezvoltat de Compq, Microsoft şi National Semiconductor care are o mai mare încărcătură hardware (Intel) şi un soft mai simplu.

Prin apariţia USB 2.0 noi specificaţii Host Controller Interface au fost necesare pentru descrierea la nivel de registru specifice USB 2.0. A apărut EHCI (Enhanced Host Controller Interface). Mai multe firme au cooperat pentru realizarea unei interfeţe standard şi a unui singur driver necesar acesteia.

USB aşa cum sugerează şi numele său este o magistrală serială. Se folosesc 4 fire ecranate din care două sunt pentru alimentare (+5V şi GND) iar celelalte două sunt fire răsucite cu semnal diferenţial de date. Se foloseşte schema de codare NRZI (Non Return to Zero Invert) pentru transmiterea datelor cu un câmp de sincronizare pentru sincronizarea ceasului la host şi la receptor.

USB suportă plug and plug (conectarea şi reconectarea) cu încărcarea şi descărcarea dinamică a driver-elor. Utilizatorul conectează dispozitivul la magistrală iar host-ul va detecta acest lucru, va interoga dispozitivul nou inserat şi va încărca drive-ul corespunzător şi afişează pe ecran faptul că driver-ul a fost instalat. Înmomentul în care utilizatorul a terminat, acesta deconectează cablul, host-ul detectează acest lucru şi elimină driver-ul.

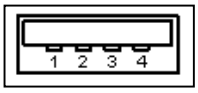
Încărcarea driver-ului potrivit este produsă de folosirea combinaţiei PID/VID (Product ID/Vendor ID). VID este furnizat de forumul implementatorilor USB contra cost şi acesta este un alt puct forte al USB. Ultimile informaţii privind taxele pot fi găsite la USB Implementor's Website.

Alte organizații de standardizare furnizează un extra VID pentru activități necomerciale cum ar fi educația şi cercetarea sau pentru pasionații domeniului. Mulți fabricanți de circuite integrate au propriile lor combinații VID/PID care pot fi folosite la dispozitive necomerciale. Alți producători de circuite integrate pot chiar să furnizeze un PID pentru a fi folosit cu VID-urile acestora în dispozitivele dumneavoastră comerciale.

Altă facilitate notabilă a USB reprezintă modurile de transfer. USB suportă transferuri Control, Interrupt, Bulk şi Isochronous. De exemplu, modul Isochronous permite unui dispozitiv să rezerve o anumită lățime de bandă cu o întârziere (latency) garantată. Acest lucru este ideal pentru aplicațiile audio şi video unde congestionarea magistrale poate produce pierderea datelor sau a frame-urilor. Despre celelalte moduri de transfer se va discuta mai târziu. Fiecare mod de transfer permite proiectantului să lucreze cu diverse domenii cum sunt: detecția şi corecția erorilor, întârzieri şi lățimi de bandă garantate.

6.4.2.2. Conectorii

Toate dispozitivele au o conexiune upstream către host şi toate host-urile au o conexiune downstream către dispozitive. Conectorii upstream şi cele downstream nu sunt interschimbabili din punct de vedere mecanic ceea ce elimină posibilitatea conectării greşite cum at fi conectarea unui hub cu un port downstream la un port downstream. Sunt două tipuri de conectori: conector de tip A şi conectori de tip B.



Conector USB de tip A.

Conector USB de tip B.

Figura 6.20. Conectorii portului USB.

Conectorul de tip A se foloseşte întotdeauna la upstream şi se întâlneşte de obicei la host-uri şi hub-uri. De exemplu conectorul de tip A se întâlneste la calculatoare la main boards şi la hub-uri. Conectorul de tip B se foloseşte la downstream şi în conseciință el se va găsi la dispozitive.

Se gasesc cabluri de conectare de la conectori de tip A la conectori de tip A şi aceasta excepție de la standardul USB se face în scopul conectării a două calculatoare. O altă excepție o reprezintă cablurile prelungitoare care au la un capat un conector tată (de tip A sau B) şi un conector de tip mamă la celălalt capăt.

Specificațiile USB 2.0 introduc o corecție şi definește conectorii mini-USB B. Aceşti conectori sunt necesari dispozitivelor electronice miniaturale cum sun telefoanele mobile sau organisers (agende electronice – PDA Personal Digital Assistant).

Recent au fost elaborate specificațiile On-The-Go care adaugă funcționalitatea peer-to-peer porturilor USB. Acest lucru permite folosirea USB host la telefoane mobile sau la agende electronice şi în acest fel s-au introdus specificațiile prizei mini-A şi a conectorului tată mini-A şi a prizei mini-AB.

TABELUL 6.5.

Numărul pinului	Culoarea cablului	Funcție
1	Roşu	V_{BUS} (5 volți)
2	Alb	D-
3	Verde	D+
4	Albastru	Masă

6.4.2.3. Caracteristici electrice

În afară de situația când realizați circuite integrate pentru dispozitive/transmițătoare USB sau host/hub USB nu trebuie să cunoaşteți în amănunt specificațiile electrice. În continuare se va prezenta esențialul despre acestea.

USB foloseşte o pereche de fire pentru transmiterea diferențială a datelor. Datele sunt codificate folosind NRZI şi biții sunt combinați în aşa fel încât să se asigure trranziția adecvată în fluxul de date. La dispozitivele de viteză low şi full, un "1" diferențial este transmis prin polarizarea liniei D+, legată la masă printr-o rezistență de 15K ohm, la peste 2,8V şi a liniei D-, legată la 3,6V printr-o rezistență de 1,5K ohm, la o tensiune sub 0,3V. Un "0" diferențial se obține cu D- la o tensiune mai mare ca 2,8V şi D+ la o tensiune mai mică de 0,3V, linii conectate cu rezistențele specificate mai sus.

Receptorul defineşte un "1" diferențial ca D+ cu 200mV mai mare ca D- şi "0" diferențial ca D+ cu 200mV mai mic decât D-. Polaritatea semnalului este inversată în funcție de viteza magistralei. Adesea termenii de stare "J" şi "K" sunt folosiți pentru specificarea nivelelor logice. La viteză scăzută (low) o stare "J" reprezintă un "0" diferențial. La viteză înaltă (high) starea "J" reprezintă un "1" diferențial.

Transmițătoarele USB (transceivers) au atât ieşiri diferențiale cât şi single ended. Anumite stări ale magistralei sunt indicate prin semnale single ended cu D+, D- sau amândouă. De exemplu un zero single ended (SE0) poate fi folosit pentru a semnala un reset al unui dispozitiv dacă ține mai mult de 10ms. Un SE0 este generat prin punerea atât a liniei D+ cât şi a liniei D- la nivel scăzut (< 0,3V). Ieşirile single ended şi diferențiale reprezintă informații importante dacă folosiți un transceiver şi un FPGA ca dispozitiv USB. You cannot get away with sampling just the differential output. (Nu puteți merge mai departe cu ieşirea drept ieşire diferențială!?)
Magistralele de viteză scăzută (low)/ completă (full) au o impedanță caracteristică de 90 ohm +/- 15%. Ieste important să se citească schema când se aleg rezistențele serie de adaptare a impedanței pentru D+ şi D-. Orice schemă bună trebuie să specifice aceste valori şi toleranțele acestora.

Viteza înaltă (high) – 480 Mbiți/s) foloseşte un curent constant de 17,78 mA pentru semnal în scopul reducerii zgomotului.

6.4.2.4. Identificarea vitezei

Un dispozitiv USB trebuie să indice vitezele sale prin punerea atât a liniei D+ cât şi a liniei D- la 3,3 volți. Un dispozitiv cu viteză completă (full), prezentat mai jos, va folosi o rezistență conectată la D+ pentru a specifica acest lucru. Rezistorul prezent la dispozitiv este folosit de altfel şi de către host sau hub pentru a detecta prezența dispozitivului conectat la portul său. Fără aceste rezistențe USB consideră că nu s-a conectat nimic la magistrală. Unele dispozitive au aceste rezistențe în circuitul integrat, rezistențe ce pot fi conectate sau deconectate prin program, altele au rezistența în exterior.

De exemplu Philips Semiconductor foloseşte tehnologia SoftConnect™. La prima conectare la magistrală această tehnologie permite microcontrolerului să inițializeze funcțiile dispozitivului USB înainte de a valida conectarea rezistorului de identificare a vitezei şi care indică faptul că dispozitivul este conectat la magistrală.

Dacă rezistența ar fi conectată imediat la V_{bus} atunci acest lucru ar indica host-ului un dispozitiv conectat. Host-ul va încerca să reseteze (inițializeze) dispozitivul şi să ceară descriptorul atunci când microprocesorul nu a început încă să inițializeze funcțiile USB ale dispozitivului.

Alți fabricanți precum Cypress Semiconductor folosesc de asemenea o rezistență programabilă pentru Re-Numeration™ la dispozitivele EzUSB unde un singur dispozitiv poate fi enumerat pentru o funcție cum ar fi In field programming când este deconectat de la magistrală sub controlul programului (firmware) şi enumerat ca un dispozitiv diferit într-o fracțiune de secundă. Multe dispozitive EyUSB nu au nici un fel de memorie Flash sau OTP ROM pentru stocarea codului. Acestea sun bootstraped (pornite) la conectare.

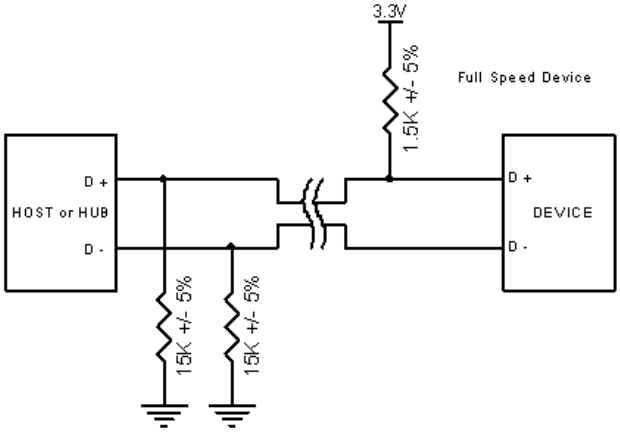


Figura 6.21. Dispozitiv full speed cu rezistență pull up conectată la D+

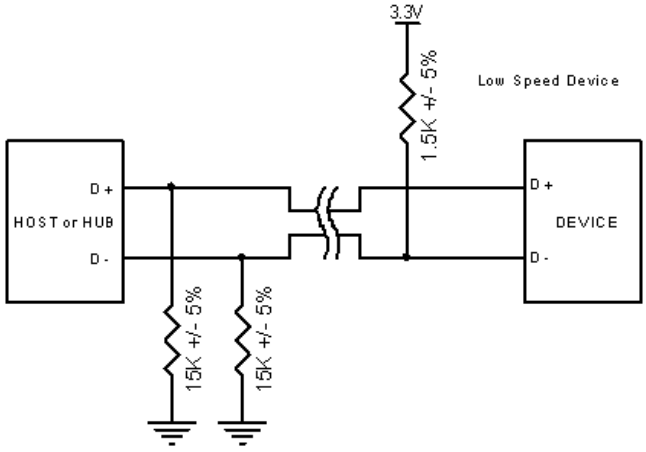


Figura 6.22. Dispozitiv low speed cu rezistență pull up conectată la D-

Trebuie notat faptul că nu a fost prezentată identificarea vitezei înalte (high). Dispozitivele de viteză înaltă vor porni conectându-se ca dispozitive de viteză completă (full) – 1,5k la 3,3V. O dată ce acestea sunt conectate vor transmite un semnal de înaltă

viteză pe durata resetului stabilind o conexiune de înaltă viteză dacă hub-ul suportă modul de înaltă viteză. Dacă dispozitivul lucrează în modul de înaltă viteză, retistența pull up este îndepărtată pentru echilibrarea liniei.

Un dispozitiv compatibil USB 2.0 nu trebuie neapărat să poată suporta modul de viteză înaltă. Aceasta permite producerea dispozitivelor ieftine la care viteza nu este critică. Acesta este de asemenea cazul dispozitivelor de viteză scăzută USB 1.1 care nu trebuie să suporte şi viteza completă (full).

Oricum, un dispozitiv de viteză înaltă poare să nu suporte modul de viteză scăzută. Acesta trebuie să suporte doar modul de viteză completă (full) necesar la conectare după care este negociat modul de viteză high. Un dispozitiv de intrare compatibil USB 2.0 (downstream) – hub sau host – suportă toate cele trei moduri de viteză: high speed, full speed şi low speed.

6.4.2.5. Alimentarea V_{BUS}

Unul din avantajele USB este reprezentat de dispozitivele alimentate de la magistrală – dispozitive care nu mai necesită sursă de alimentare externă.

Un dispozitiv USB specifică consumul său de putere în unități de 2 mA în descriptorul configurației care va fi studiat în detaliu mai târziu. Un dispozitiv nu poate creşte consumul său de putere peste valoarea specificată la enumerare, chiar dacă acesta pierde alimentarea externă. Sunt trei clase de funcții USB:

- Low-power bus powered functions
- High-power bus powered functions
- Self-powered functions

Funcția de alimentare de joasă putere a magistralei ia toată puterea de la V_{BUS} şi nu poate furniza mai mult de o unitate de încărcare. Specificațiile USB definesc o unitate de încărcare ca fiind 100 mA. Magistrala de alimentare de joasă putere trebuie de asemenea să fie proiectată să lucreze pâna la tensiunea cea mai joasă V_{BUS} de 4,40V şi pâna la tensiunea maximă de 5,25V măsurată la conectorul upstream al dispozitivului. Pentru dispozitivele de 3,3V regulatorul este obligatoriu.

Funcția de alimentare de înaltă putere a magistralei ia puterea de la magistrală şi nu poate furniza mai mult de o unitate până când a fost configurată după care poate furniza 5 unități (500 mA max) specificate de către descriptor. Funcția magistralei de înaltă putere trebuie să poată fi detectată şi enumerată la tensiunea minimă de 4,40V. Când se operează la încărcarea maximă (5 unități), o tensiune minimă V_{BUS} de 4,75V este specificată cu un maxim la 5,25. Măsurătorile se fac la conectorul upstream.

6.4.2.6. Protocolul

Spre deosebire de interfața serială la care formatul datelor transmise nu este definit, USB utilizează protocoale pe mai multe nivele.

Orice tranzacție a portului USB constă în:

- Token Packet (un antet ce defineşte ce ne aşteptăm să urmeze);

- Optional Data Packet (conținând datele utile);
- Status Packet (folosit pentru confirmarea tranzacției şi care furnizează mijloace pentru corecția erorilor).

Datele sunt transmise de către USB începând cu cel mai puțin semnificativ bit. Structura unui pachet USB conține următoarele cîmpuri:

- **Sync** – toate pachetele trebuie să înceapă cu un câmp sync. Acest câmp are o lungime de 8 biți la viteza low şi full şi 32 de biți la viteza high şi este folosit la sincronizarea ceasului receptorului cu cel al transmițătorului. Ultimii doi biți semnalează faptul că urmează câmpul PID;
- **PID** – reprezintă identificatorul pachetului (Packet ID). Acest câmp este folosit pentru identificarea pachetului care urmează a fi transmis;
- **ADDR** – câmpul adresă specifică dispozitivul căruia îi este destinat pachetul. Acest câmp are şapte biți şi deci se pot adresa 127 de dispozitive. Adresa zero este folosită pentru dirijarea pachetelor ce au adrese pentru dispozitive nealocate inca si deci această adresă nu poate fi alocată unui dispozitiv;
- **ENDP** – câmpul endpoint care poate avea pâna la 4 biți permițând 16 posibilități endpoint;
- **CRC** – Codul ciclic de eroare (Cyclic Redundancy Checks);
- **EOP** – End of packet – sfârşitul pachetului.

USB foloseşte patru tipuri de pachete: *token packets* care indică tipul tranzacției care urmează, *data packets* care conțin datele de transmis, *handshake packets* care sunt folosite pentru încheierea transmisiei (handshake) coținând informații despre confirmarea transmisiei şi erorile produse şi *start of frame packets* care indică începutul unui nou cadru.

Token Packets

Există trei tipuri de pachete token: **In** – care informează dispozitivul USB care host doreşte să citească informația, **Out** – care informează dispozitivul USB care host doreşte să transmită informația, **Setup** – folosit pentru inițierea controlului transferurilor.

Pachetele token au formatul următor:

Sync	PID	ADDR	ENDP	CRC5	EOP

Data Pakets

Există două tipuri de pachete de date fiecare dintre acestea fiind capabil sa transmită pâna la 1024 octeți de date: Data0 şi Data1.

Modul High Speed defineste încă două PID-uri pentru date: DATA2 şi MDATA.

Pachetele de date au următorul format:

Sync	PID	Data	CRC16	EOP

Dimensiunea maximă a datelor pentru viteza low este de 8 octeți, pentru viteza full este de 1023 de octeți iar pentru viteza high este de 1024 de octeți.

Handshake Packets

Sunt trei tipuri de pachete handshake care constau în esență numai din PID: **ACK** – care semnalează că datele au fost recepționate cu succes, **NAK** – raportează că temporar, dispozitivul nu poate recepționa sau transmite date sau, pe durata tranzacției unei întreruperi informează host-ul că nu sunt date de transmis, **STALL** – este necesară intervenția host-ului.

Pachetele handshake au următorul format:

Sync	PID	EOP

Start of Frame Packets

Pachetele SOF constau din cadre de 11 biți şi sunt transmise de fost la fiecare interval de o milisecundă pe magistralele de viteză full sau la fiecare 125μs pe magistralele cu viteză high.

Pachetele handshake au următorul format:

Sync	PID	Frame Number	CRC5	EOP

6.5. Interfețele microcontrolerelor

Cele două circuite integrate prezentate: interfața serială 8251 şi interfața paralelă 8255 sunt circuite utilizate în realizarea sistemelor de calcul cu microprocesor. Microcontrolerele dețin şi ele, integrate pe acelaşi cip cu unitatea centrală, astfel de interfețe mai simple sau mai complexe. De regulă un microcontroler deține o interfață serială asincronă pentru comunicația cu alte sisteme de calcul, o interfață serială sincronă de mare viteză pentru comunicația cu alte componente integrate care se pot utiliza în sistem împreună cu microcontrolerul şi un sistem de intrări / ieşiri numerice similar cu interfața paralelă prezentată.

Se va prezenta în continuare convertorul analog-numeric al microcontrolerului PIC 16F87x, reprezentativ pentru familia sa.

6.5.1. Modulul convertor analog-digital (A/D)

Convertorul analog-digital are cinci intrăti la dispozitivele cu 28 de pini şi opt intrări la cele cu 40/44 de pini.

Conversia fiecărei intrări se face pe zece biți.

Modului A/D are o referință de tensiune înaltă şi o referință de tensiune scăzută selectabile prin program prin anumite combinații între V_{DD}, V_{SS}, RA2 sau RA3.

Convertorul A/D poate funcționa şi în modul SLEEP dacă se asigură pentru convertorul analog-digital semnal de ceas de la oscilatorul RC intern.

Convertorul analog digital are patru regiştrii:

- registrul rezultat al celui mai semnificativ octet al convertorului analog-digital (ADRESH);
- registrul rezultat al celui mai puțin semnificativ octet al convertorului analog-digital (ADRESL);
- registrul de control 0 (ADCON0);
- registrul de control 1 (ADCON1).

Registrul ADCON0 controlează funcționarea modulului analog-digital.

Registrul ADCON1 configurează pinii portului care pot fi: intrări analogice (RA3 poate fi, de asemenea, rerința de tensiune) sau intrări/ieşiri digitale.

Registrul ADCON0 (adresa 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/~~DONE~~	-	ADON
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

R = bit de citire; W = bit de scriere U = bit neimplementat, citit ca zero; n = valoare la resetul power-on; '1' = bitul este setat; '0' = bitul este resetat; x = valoarea bitului este nu este cunoscută

bit 7-6 ADCS1:ADCS0 biții de selecție a ceasului de conversie A/D

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Ceas de conversie
0	00	$F_{OSC}/2$
0	01	$F_{OSC}/8$
0	10	$F_{OSC}/32$
0	11	F_{RC} (ceas derivat de la ceasul intern A/D RC)
1	00	$F_{OSC}/4$
1	01	$F_{OSC}/16$
1	10	$F_{OSC}/64$
1	11	F_{RC} (ceas derivat de la ceasul intern A/D RC)

bit 5-3 CHS2:CHS0 biți de selecție ai canalului analogic

000 = canal 0 (AN0)
001 = canal 1 (AN1)
010 = canal 2 (AN2)
011 = canal 3 (AN3)
100 = canal 4 (AN4)
101 = canal 5 (AN5)
110 = canal 6 (AN6)
111 = canal 7 (AN7)

Notă: PIC16F873A/876A au implementat numai canalele A/D 0 pâna la 4; combinațiile neimplementate sunt rezervate; nici una dintre aceste combinații nu trebuie trimise la aceste dispozitive.

bit 2 GO/~~DONE~~ bit de stare a conversiei

0 = conversie A/D în desfăşurare (setarea acestui bit porneşte conversia A/D şi bitul este şters automat de către hardware când conversia A/D este completă);
1 = conversia nu este în desfăşurare.

bit 1 Neimplementat. La citire se citeşte zero.

bit 0 ADON bit de alimentare a convertortorului A/D

1 = convertorul A/D este alimentat;
0 = convertorul este nealimentat şi nu consumă curent.

Registrul ADCON1 (adresa 9Fh)

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	-	-	PCFG3	PCFG2	PCFG1	PCFG0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

R = bit de citire; W = bit de scriere U = bit neimplementat, citit ca zero; n = valoare la resetul power-on; '1' = bitul este setat; '0' = bitul este resetat; x = valoarea bitului este nu este cunoscută

bit 7 ADFM bit de selecție a formatului rezultatului A/D

1 = aliniere dreapta. Şase din cei mai semnificativi biți ai ADRESH sunt citiți ca zero;

0 = aliniere stânga. Şase din cei mai puțini semnificativi biți ai ADRESL sunt citiți ca zero.

bit 6 ADCS2 bit de selecţie a ceasului de conversie A/D

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Ceas conversie
0	00	$F_{OSC}/2$
0	01	$F_{OSC}/8$
0	10	$F_{OSC}/32$
0	11	F_{RC} (ceas derivat de la ceasul intern A/D RC)
1	00	$F_{OSC}/4$
1	01	$F_{OSC}/16$
1	10	$F_{OSC}/64$
1	11	F_{RC} (ceas derivat de la ceasul intern A/D RC)

biţii 5-4 Neimplementaţi. La citire sunt citiţi ca zero.

biţii 3-0 PCF3:PCF0 biţii de control ai configurării A/D

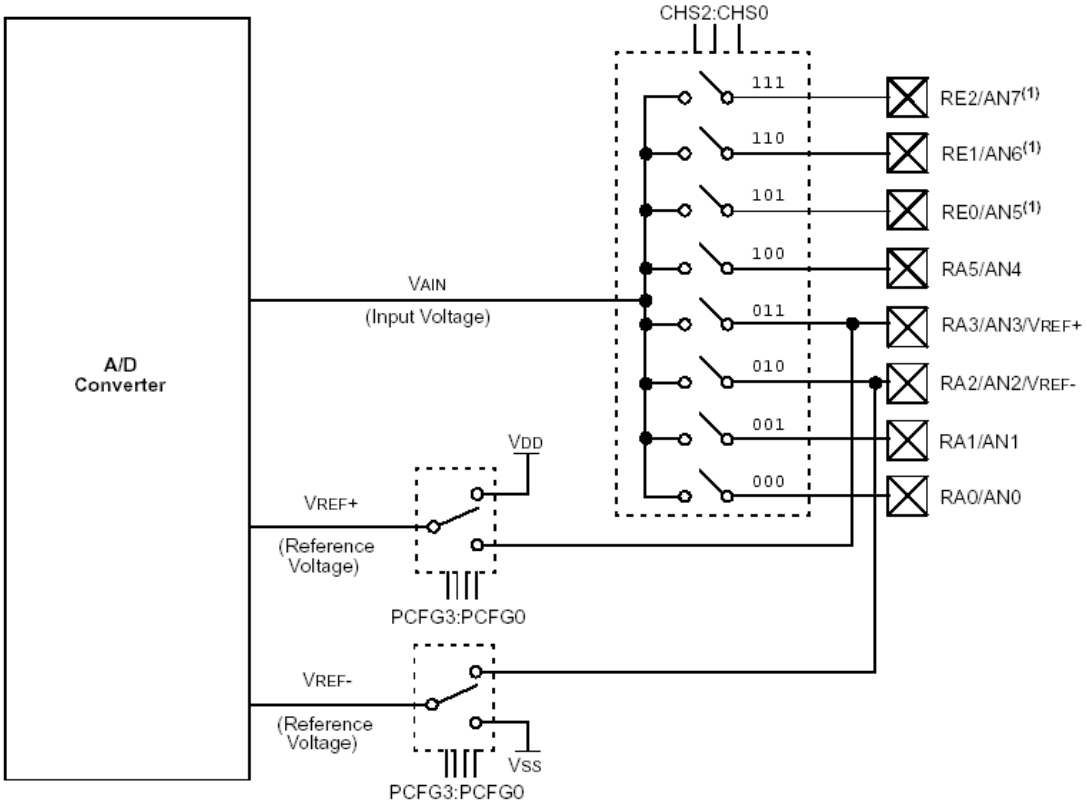
PGF <3:9>	**AN7**	**AN6**	**AN5**	**AN4**	**AN3**	**AN2**	**AN1**	**AN0**	**V_{REF+}**	**V_{REF-}**	**C/R**
0000	A	A	A	A	A	A	A	A	V_{DD}	V_{SS}	8/0
0001	A	A	A	A	V_{REF+}	A	A	A	AN3	V_{SS}	7/1
0010	D	D	D	A	A	A	A	A	V_{DD}	V_{SS}	5/0
0011	D	D	D	A	V_{REF+}	A	A	A	AN3	V_{SS}	4/1
0100	D	D	D	D	A	D	A	A	V_{DD}	V_{SS}	3/0
0101	D	D	D	D	V_{REF+}	D	A	A	AN3	V_{SS}	2/1
011x	D	D	D	D	D	D	D	D	-	-	0/0
1000	A	A	A	A	V_{REF+}	V_{REF-}	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	V_{DD}	V_{SS}	6/0
1010	D	D	A	A	V_{REF+}	A	A	A	AN3	V_{SS}	5/1
1011	D	D	A	A	V_{REF+}	V_{REF-}	A	A	AN3	AN2	4/2
1100	D	D	D	A	V_{REF+}	V_{REF-}	A	A	AN3	AN2	3/2
1101	D	D	D	D	V_{REF+}	V_{REF-}	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	V_{DD}	V_{SS}	1/0
1111	D	D	D	D	V_{REF+}	V_{REF-}	D	A	AN3	AN2	1/2

A = intrare analogică D = I/O digitală
C/R = numărul canalelor de intrare analogice/numărul tensiunilor de referinţă pentru conversia A/D

Notă: După RESET pinii dispozitivului care sunt multiplexaţi cu funcţii analogice (Anx) sunt forţaţi să fie intrări analogice.

Regiştrii ADRESH:ADRESL conţin cei 10 biţi ai rezultatului conversiei A/D. Când conversia A/D este completă, rezultatul este încărcat în perechea de regiştrii ai rezultatului, bitul GO/~~DONE~~ (ADCON0<2>) este şters şi fanionul întreruperii A/D este setat.

Schema bloc a modulului convertorului A/D este prezentat în figura următoare.



Schema bloc a convertorului analog-digital

După ce modulul A/D a fost configurat, canalul selectat trebuie achiziționat înainte de a porni conversia analog-digitală. Canalele de intrare analogice trebuie să aibă biții corespunzători din registrul TRIS selectați ca intrări.

După ce timpul de achiziție s-a scurs, conversia A/D poate să înceapă.

Paşii care trebuie urmați pentru realizarea unei conversii analog-digitale sunt:

1. Configurarea modulului A/D:

   - configurarea pinilor de intrare a semnalului analogic/referință de tensiune şi a I/O digitale (ADCON1);
   - selectarea canalelor de intrare A/D (ADCON0);
   - selectarea ceasului de conversie (ADCON0);
   - alimentarea modulului A/D (ADCON0).

2. Configurarea intreruperii A/D (dacă este necesară):

   - ştergerea bitului ADIF;
   - setarea bitului ADIE;
   - setarea bitului PEIE;
   - setarea bitului GIE.

3. Se aşteaptă timpul de achiziţie necesar.
4. Se porneşte conversia:

   - se setează bitul GO/~~DONE~~ (ADCON0).

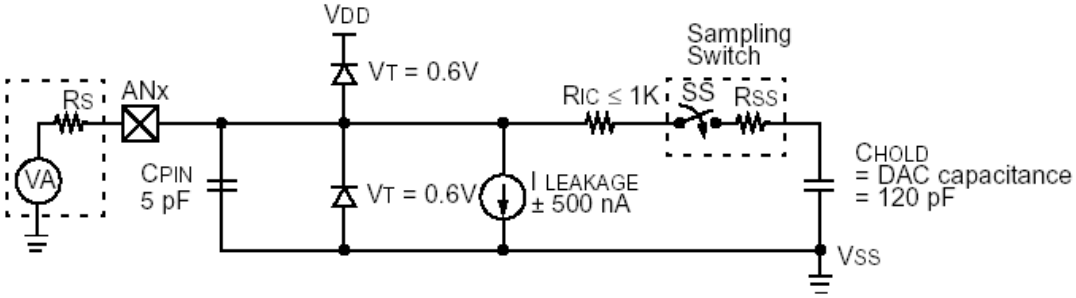
5. Se aşteaptă terminarea conversiei A/D prin:

   - testarea prin program a bitului GO/~~DONE~~;
   - sau (dacă întreruperile au fost validate) se aşteaptă întreruperea A/D.

6. Se citeşte rezultatul conversiei din perechea de regiştrii ADRESH:ADRESL, se şterge bitul ADIF, dacă este necesar.
7. Pentru următoarea conversie se face salt la pasul 1 sau 2 după necesităţi. Timpul de conversie A/D este definit ca T_{AD}.

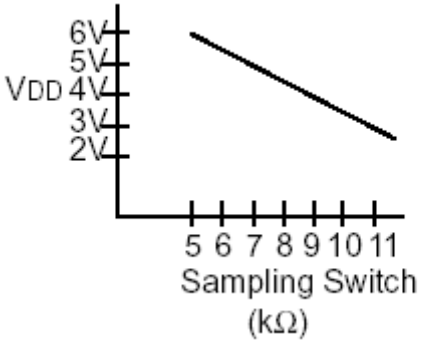
6.5.1.1. Cerinţele achziţiei analog-digitale

Pentru convertorul A/D trebuie îndeplinite anumite condiţii pentru obţinerea unui rezultat precis al conversiei analog-digitale. În primul rând încărcarea condensatorului de reţinere (C_{HOLD}) trebuie să fie completă pâna la tensiunea prezentă la intrarea analogică. Modelul intrării analogice este prezentat în figura următoare:



Legendă:

C_{PIN}	= capacitate de intrare
V_T	= tensiune de prag
$I_{LEAKAGE}$	= curent de scurgere la pin datorat conexiunii
R_{IC}	= rezistenţa de interconectare
SS	= comutatorul de eşantionare
C_{HOLD}	= capacitatea de eşantionare/reţinere (pentru DAC)



Impedanța sursei (R_S) şi impedanța comutatorului de eşantionare intern (R_{SS}) afectează în mod direc timpul necesar încărcării capacitîții C_{HOLD}. Impedanța comutatorului (R_{SS}) variază în funcție de tensiunea de alimentare a dispozitivului (V_{DD}).

Impedanța maximă recomandată pentru sursa analogică este de 10 kΩ. Dacă impedanța scade, timpul de achiziție poate fi scăzut.

După ce canalul intrării analogice este selectat (schimbat), trebuie să se facă achiziția semnalului înainte de pornirea conversiei analog-numerice.

Pentru a calcula timpul minim de achiziție se poate folosi ecuația:

T_{ACQ} = Timpul de setare a amplificatorului + Timpul de încărcare a capacității de reținere + coeficientul de temperatură

$= T_{AMP} + T_C + T_{COFF}$
$= 2\mu s + T_C + [\text{Temperatura} - 25^0C)\ (0{,}05\mu s/^0C)]$
$TC = C_{HOLD}\ (R_{IC} + R_{SS} + R_S)\ \ln(1/2047)$
$= -\ 120pF\ (1k\Omega + 7k\Omega + 10k\Omega)\ \ln(0{,}0004885)$
$= 16{,}47\mu s$
$T_{ACQ} = 2\ \mu s + 16{,}47\ \mu s + [50^0C - 25^0C)(0{,}05\ \mu s/^0C)$
$= 19{,}72\ \mu s$

În această ecuație se foloseşte eroarea maximă permisă de 1/2 LSB (1024 paşi pentru A/D).

6.5.1.2. Selecția ceasului conversiei analog-digitale

Timpul de conversie pe bit este definit ca T_{AD}. Conversia A/D necesită cel puțin $12T_{AD}$ pentru conversia pe 10 biți. Sursa ceasului de conversie A/D este selectatabilă prin program. Cele şapte variante posibile pentru T_{AD} sunt:

- $2T_{OSC}$
- $4T_{OSC}$
- $8T_{OSC}$
- $16T_{OSC}$
- 32_{OSC}
- $64T_{OSC}$
- oscilatorul RC intern al modulului A/D (2-6μs).

Pentru o conversie corectă A/D ceasul de conversie T_{AD} trebuie astfel ales încât să asigure timpul minim de conversie de 1,6 μs.

Tabelul următor arată timpul T_{AD} rezultat din frecvențele de operare ale dispozitivului şi din sursa de ceas A/D selectată.

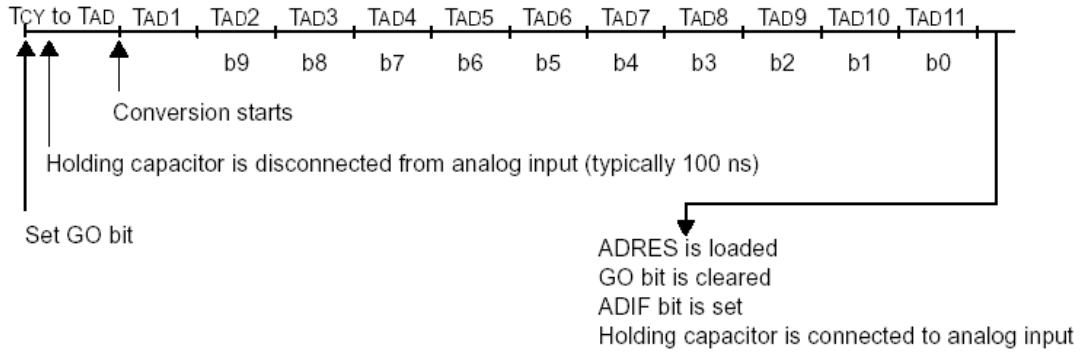
Sursa ceasului AD (T_{AD})		Frecvența maximă a dispozitivului
Operare	ADCS2:ADCS1:ADCS0	Max.
$2T_{OSC}$	000	1,25MHz
$4T_{OSC}$	100	2,5MHz
$8T_{OSC}$	001	5MHz
$16T_{OSC}$	101	10MHz
$32T_{OSC}$	010	20MHz
$64T_{OSC}$	110	20MHz
$RC^{(1,2,3)}$	x11	(Nota 1)

Nota: 1. Sursa RC are un T_{AD} tipic de 4µs, dar vatriază între 2-6µs.
2. Când frecvența dispozitivului este mai mare decât 1 MHz, sursa de ceas RC A/D este singura recomandată pentru operarea SLEEP.
3. Pentru dispozitive cu tensiune de alimentare extinsă se vor studia caracteristicile electrice.

6.5.1.3. Conversia A/D

Dacă se şterge bitul GO/~~DONE~~ pe timpul unei conversii atunci acea conversie este abandonată iar perechea de regiştrii cu rezultatul conversiei nu vor fi modificați, După abandon, o nouă achiziție de date este pornită automat pentru canalul selectat. Bitul GO/~~DONE~~ poate atunci fi setat pentru a porni conversia.

În figura de mai jos, după ce bitul GO este setat, primul segment de timp este minimum T_{CY} şi maximum T_{AD}.



Ciclii T_{AD} ai conversiei A/D

T_{cy} reprezintă un ciclu maşină ($f_{osc}/4$)

Regiştrii asociați convertorului analog-digital sunt: INTCON, PIR1, PIE1, ADRESH, ADRESL, ADCON0, ADCON1, TRISA, PORTA, TRISE, PORTE.

CAPITOLUL 7

CIRCUITE SPECIALE

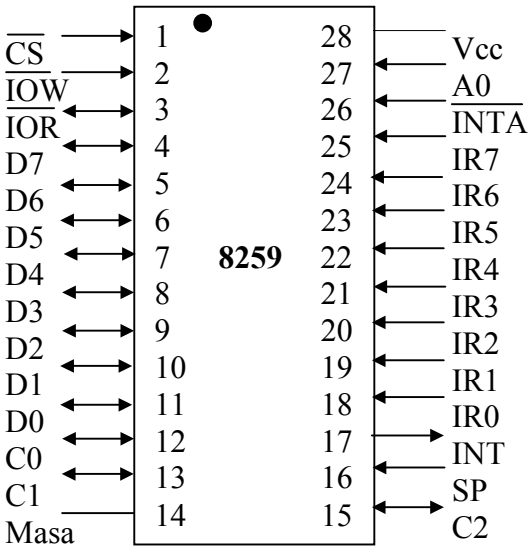
7.1. Introducere

Sistemele de calcul sunt prevăzute suplimentar cu circuite destinate unor funcţii care care duc la creşterea versatilităţii arhitecturii prin adăugarea unor funcţii noi. Astfel, printre circuitele suplimentare adăugate unei arhitecturi, se pot menţiona: circuitul pentru gestionarea prioritară a întreruperilor, circuitul pentru controlul accesului direct la memorie (Direct Access Memory – DMA) şi circuitul timer (un circuit destinat măsurării intervalelor de timp şi contorizării evenimentelor).

7.2. Controlerul de întreruperi programabil 8259

Unitatea 8259 este realizată în tehnologia NMOS, pe un singur circuit integrat, cu 28 terminate ale căror semnificaţii sunt date în figura 7.1. O singură unitate poate manipula până la 8 cereri de întreruperi externe şi asigură o varietate de maodalităţi programabile, pentru arbitrarea priorităţilor acestora. Datorită posibilităţilor de conectare în cascadă, implementate în unitatea 8259, se pot interconecta, la o unitate 8259-master, până la opt unităţi-slave, asigurlndu-se astfel facilităţl de tratare a 64 niveluri prioritare de întrerupere. Modalităţile de arbitrare pot fi programate diferit la unitatea master şi la fiecare unitate slave.

Schema bloc funcţională este prezentată în figura 7.2. Ea constă din mai multe registre şi reţele logice grupate în jurul unei magistrale interne.



D_7-D_0	- magistrala de date (bidirecţional);
$\overline{CS}$	- selecţie circuit (intrare);
A0	- identifică unul din cele două porturi ale unităţii (intrare);
$\overline{IOR}$	- semnal de comandă citire (intrare);
$\overline{IOW}$	- semnal de comandă scriere (intrare);
INT	- cerere de întrerupere trimisă de unitatea centrală (ieşire);
$\overline{INTA}$	- recunoaştere întrerupere (intrare);
SP	- identifică unitatea fie ca master, fie ca slave (intrare);
C0-C2	- linii de selecţie a unităţilor slave în sistemele cu unităţi multiple (ieşire la master, intrere la slave);
VCC	- tensiune de alimentare pozitivă;

Figura 7.1. Semnificaţia terminalelor unităţii pentru comanda prioritară a întreruperilor 8259.

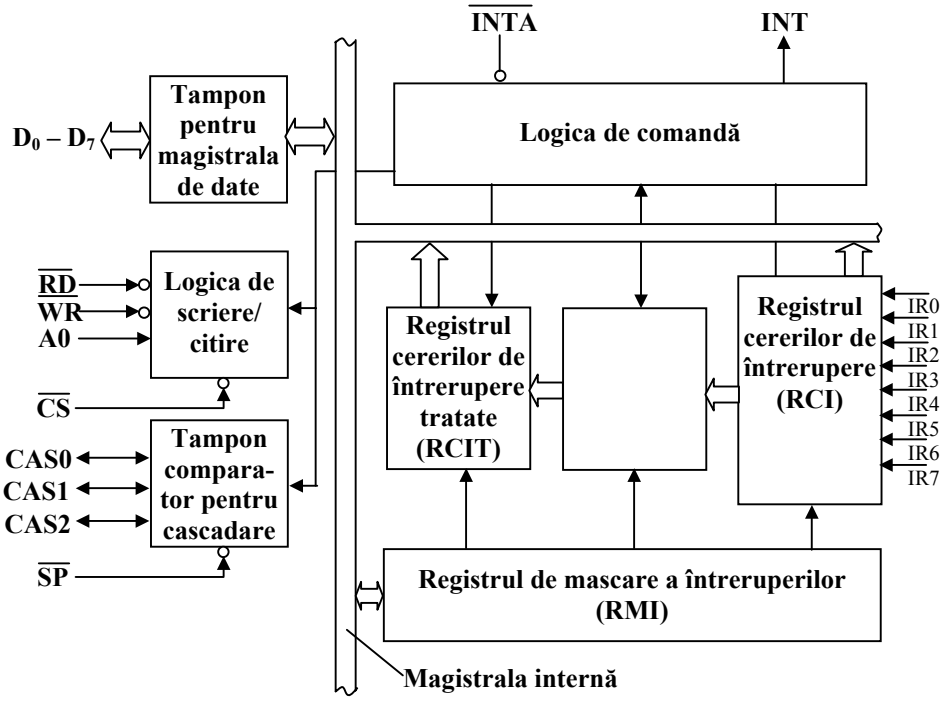


Figura 7.2. Schema bloc a unității 8259.

Cererile de întrerupere IR_0-IR_7, de la echipamentele periferice, poziționează în unu, pe fronturile pozitive, când devin active, bistabilii corespunzătoi ai Registrului Cererilor de Intrerupere (ROI). Poziționarea în unu, a unuia sau mai multor bistabili din registrul RCI, are ca efect generarea unui semnal (nivel ridicat) pe linia INT. Conținutul lui RCI, corespunzător cererii de întrerupere considerate, este forțat în zero prin secvența INTA.

Noua cerere de întrerupere considerată va fi înregistrată într-un bistabil corespunzător, din Registrul Cererilor de Întrerupere Tratate (RCIT), odată cu forțarea în zero, a bistabilului asociat întreruperii respective, în RCI. Bistabilul din RCIT va fi readus în zero prin program, la sfârştul operației de tratare a întreruperii, printr-o comandă adecvată (OCW2 - a se vedea programarea unitatii 8259), dată înaintea revenirii în prograraul prindpal.

Circultul pentru rezolvarea prioritaților cererilor de întrerupere stabileşte prioritățile cererilor, care au poziționat în unu bistabilii RCI. Pe durata impulsului $\overline{\text{INTA}}$, bitul cu prioritatea cea mai mare este forțat din RCI în RCIT.

Semnalul de întrerupere INT, generat de unitatea 8259, este forțat direct la microprocesor.

Sernnalul de recunoaştere a întreruperii ($\overline{\text{INTA}}$), este generat de către unitatea centrală. Primul semnal $\overline{\text{INTA}}$, va face ca 8259 sa forțeze, pe magistrala de date D_0-D_7, codul instrucțiunii CALL. Ca urmare a acestui cod , unitatea centrală va furniza, în ciclurile maşină care urmează, încă două semnale $\overline{\text{INTA}}$, la care unitatea 8259 va forța, pe magistrala de date, cei doi octeți de adresă ai instrucțiunii CALL. În acest mod,

unitatea centrală de prelucrare va începe execuția rutinei corespunzatoare, de tratare a întreruperii, a cărei adresă de start a fost furnizată de instructiunea CALL.

Registrul de Mascare a Întreruperilor (RMI) conține biții cererilor de întrerupere care trebuie mascate. Conținutul său are efect asupra informației din RCIT şi RCI.

Tamponul pentru magistrala de date este bidirecțional, fiind prevăzut cu circuite cu trei stări, ceea ce permite conectarea directă a unității 8259 la magistrala de date a unității centrale de prelucrare. Prin acest tampon sunt transferate cuvintele de comandă şi informația de stare.

Logica pentru comanda scrierii/citirii acceptă comenzi de la unitatea centrală de prelucrare şi transferă către aceasta cuvântul de stare al unității 8259. Această logică conține registrele pentru Cuvântul de Comanda de Inițializare (CCI) şi Cuvântul de Comandă al Operării (CCO).

Intrarea de Selectie a circuitului ($\overline{CS}$) activează unitatea 8259, când este pe nivel coborât.

Intrarea comenzii de scriere ($\overline{WR}$), activă pe nivel coborât, permite să se înscrie cuvinte de comandă (CCI, CCO) în unitatea 8259.

Intrarea comenzii de citire ($\overline{RD}$), activă pe nivel coborât, permite transmiterea, de către unitatea 8259, spre unitatea centrală de prelucrare, pe magistrala de date, a conținuturilor RCI, RCIT, RMI sau a codului binar zecimal al nivelului de întrerupere.

Semnalul de intrare A_0, folosit împreună cu comenzile $\overline{RD}$ şi $\overline{WR}$ permite scrierea de comenzi în diferite registre de comandă sau citirea conținuturilor unor registre. Intrarea A_0 se poate conecta direct la una din liniile de adrese.

În vederea extinderii schemei sistemului de întrerupere pentru a putea manipula pâna la 64 de cereri de întrerupere, se pot folosi mai multe unități 8259, dintre care una va juca rol de master, iar celelalte de slave. Unitatea master va fi desemnată prin conectarea terminalului SP la un nivel pozitiv de tensiune. Conectarea terminalului SP la un nivel coborât de tensiune va desemna o unitate de tip slave.

Tamponul comparator pentru cascadare memorează şi compară identitățile tuturor unităților 8259 folosite în sistem. Terminalele de intrări/ ieşiri (CAS_0-CAS_2) sunt folosite ca ieşiri de către unitatea master, care transmite în exterior identitatea unității slave luată în considerație. Aceleaşi terminale se folosesc ca intrări, pentru unitățile slave, în vederea selecției. Unitatea slave selectată va forța pe magistrala de date, pe durata ultimelor două semnale $\overline{INTA}$, adresa preprogramată a subrutinei, începând cu octetul cel mai puțin semnificativ.

Interacțiunea unității 80259 cu unitatea centrală de prelucrare se desfăşoară conform următoarei secvențe:

- una sau mai multe linii IR_7-IR_0 sunt activate de fronturile crescătoare ale cererilor de întrerupere;
- unitatea 8259 transmite un semnal INT, catre unitatea centrală, ca urmare a acccptării cererilor de întrerupere şi rezolvare a priorităților;
- unitatea centrală, recunoaşte semnalul INT şi generează un semnal $\overline{INTA}$;
- la recepționarea semnalului $\overline{INTA}$, unitatea 8259 răepunde cu forțarea pe magistrala de date (D_7-D_0) a codului instrucțiunii CALL;

- unitatea centrală de comandă va genera două semnale succesive $\overline{INTA}$, care vor activa forțarea pe magistrala de date, sub forma a doi octeți succesivi, a adresei rutinei chemate prin CALL, începând cu octetul cel mai puțin semnificativ;
- după terminarea tratării cererii de întrerupere se va genera comanda Sfârşit de Intrerupere (EOI), care va poziționa în zero bitul corespunzător întreruperii tratate în RCIT.

Programarea unității 8259 se realizeză prin comenzi generate de catre unitatea centrală de prelucrare, care pot fi comenzi de inițializare şi respectiv de operare.

Cuvintele de comandă pentru inițializare sunt două, pentru cazul unei singure unități 8259 şi respectiv trei, pentru cazul mai multor unități 8259, în sistem. În figura 7.3, a, b, c, d se prezintă structurile celor trei cuvinte de comandă ale inițializării (CCI1,

a) Cuvântul de comandă al inițializării CCI1

A0	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0				1	x			x

bit 1
1 – o singură unitate 8259;
0 – o unitate master şi mai multe slave;

bit 2
1 – între vectorii de adresă se lasă 4 octeți;
0 – între vectorii de adresă se lasă 8 octeți;

biții 7,6,5
biții $A_7A_6A_5$ ai vectorului de adresă de întrerupere.

b) Cuvântul de comandă al inițializării CCI2

A0	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1								

biții 7,6,5,4,3,2,1,0
biții $A_{15}A_{14}A_{13}A_{12}A_{11}A_{10}A_9A_8$ ai vectorului de adresă de întrerupere.

c) Cuvântul de comandă al inițializării CCI3

la master

A0	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1								

biții 7,6,5,4,3,2,1,0
oricare bit poziționat în unu semnifică nivelul la care a fost ataşată o unitate 8259 slave.

la slave

A0	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	x	x	x	x	x			

biții 2,1,0
identifică nivelul cererii la unitatea master 8259, la care s-a conectat unitatea slave dată.

x – valoare oarecare (indiferentă).

Figura 7.3. Structura cuvintelor de comandă ale inițializării.

CCI2, CCI3).

Primul cuvânt de comandă al inițializării CCI1 (fig. 7.3, a) este generat când A_0 = 0 şi D_4 = l. El are ca efect: aducerea în condițiile inițiale a circuitului de sesizare a unei cereri de întrerupere (tranziție pozitivă a semnalului cerere), anularea RMI, atribuirea intrării IR_7 a priorității 7, poziționarea în zero a bistabilului asociat cu masca şi a bistabilului asociat cu citirea stării. CCI1 specifică dacă se foloseşte o singură unitate sau mai multe unități 8259, indică distanța în octeți între vectorii adreselor de întrerupere (4 octeți sau 8 octeți), furnizează biții A_7-A_5, ai vectorului adresei de întrerupere, pentru instrucțiunea CALL.

Al doilea cuvânt de comandă al inițializării CCI2 (fig. 7.3, b) este forțat după CCI1, în unitatea 8259 şi specifică biții A_{15}-A_8, ai vectorului adresei de întrerupere. Se menționează că unitatea 8259 forțează automat biții A_4, A_3, A_2, corespunzător codului nivelului cererii de întrerupere, în timp ce biții A_l, A_0 sunt forțați în zero.

Al treilea cuvânt de comandă al inițalizării CCI3 se foloseşte numai în cazurile sistemelor de întrerupere cu unități 8259 multiple. Semnificațiile CCI3 diferă în funcție de unitatea căreia îi este adresat: master (fig. 7.3, c) şi slave (fig. 7.3, d).

Cuvintele de comandă a operării (CC0). După programarea unității 8259 cu cuvintele de comandă de inițializare, pot fi luate în considerație eventualele cereri de întrerupere. Modurile, disciplinele de tratare a cererilor de întreruperi, se pot programa prin cuvintele de comandă ale operării, selectate corespunzator CCOl, …, CCO3 (fig. 7.4).

Primul dintre cuvintele de comandă ale operării, CC01, se referă la programarea registrului de mascare a întreruperilor (RM1). Acest registru va opera asupra RCI şi RCIT. Dacă o întrerupere a fost recunoscută de către unitatea 8259, nivelul respectiv, deşi mascat prin CCOl, va inhiba nivelurile mai puțin prioritare. Pentru a evita acest lucru se poate emite un CCO2, reprezentând comanda de terminare a întreniperii (EOI) pentu a anula bitul corespunzător din RCIT sau se va utiliza un CCO3, reprezentând un mod special de mascare.

Modul de tratare cu imbricare completă, a cercrilor de întrerupere, se realizează fară transmterea vreunui CCOi, după inițializare. În acest mod prioritățile sunt atribuite în ordinea 0, …,7, prioritatea cea mai mare având-o nivelul 0. La recunoaşterea unei întreruperi, se determină cererea curentă cu prioritatea cea mai mare, se forțează vectorul adresă corespunzător pe magistrală şi se poziționează în unu bitul asociat din RCIT. Acest bit rămâne poziționat în unu până la comanda terminării întreruperii, înaintea revenirii în rutina principală întreruptă. În acest timp sunt inhibate toate cererile de întrerupere situate pe niveluri mai puțin prioritare decât nivelul tratat. Sunt acceptate cererile de întrerupere corespunzătoare unor niveluri superioare.

Modul de tratare cu rotirea priorității este folosit în aplicațiile în care echipamentele, care solicită întreruperi, au priorități egale (de exemplu: canalele de comunicație). Acest mod are două variante: modul cu auto-rotire şi modul special. În cazul auto-rotirii, dispozitivul care a fost tratat primeşte nivelul de prioritate cel mai coborât. Modul special permite modificarea priorităților, de către programator, în sensul de a stabili nivelurile cu prioritatea minimă şi respectiv maximă.

Terminarea normală a întreruperii (EOI) şi terminarea specială a întreruperii (SEOI) se referă la modalitățile de anulare a bitului, din RCIT, corespunzător cererii de întrerupere care s-a tratat. În cazul modului cu imbricare completă, unitatea anulează

bitul cu prioritatea cea mai mare (EOI), în timp ce, în modul cu rotirea priorității, trebuie să se specifice prin comandă (SEOI) nivelul care urmează să fie anulat (vezi CCO2 din fig. 7.4, b).

d) Cuvântul de comandă al operării CCO1. Masca de întrerupere.

A0	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1								

biții 7,6,5,4,3,2,1,0

1 – poziționare mască;
0 – anulare mască.

e) Cuvântul de comandă al operării CCO2. Comanda terminării întrerupere (EOI).

A0	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	R	SEOI	EOI	0	0	L2	L1	L0

R - rotirea priorității (1 – rotește, 0 – fără rotire); SEOI - terminare specială a întreruperii; EOI - terminare normală a întreruperii.

biții 2,1,0

000 – se selectează nivelul 0 ca având prioritate minimă;
001 – se selectează nivelul 1 ca având prioritate minimă;
010 – se selectează nivelul 2 ca având prioritate minimă;
011 – se selectează nivelul 3 ca având prioritate minimă;
100 – se selectează nivelul 4 ca având prioritate minimă;
101 – se selectează nivelul 5 ca având prioritate minimă;
110 – se selectează nivelul 6 ca având prioritate minimă;
111 – se selectează nivelul 7 ca având prioritate minimă;

biții 7,6,5

000 – neoperațional;
001 – terminare simplă a întreruperii, se ignoră biții 2,1,0;
010 – neoperațional;
011 – terminare specială a întreruperii, se anulează bitul RCIT specificat de biții 2,1,0;
100 – neoperațional;
101 – terminare întrerupere şi se execută rotirea priorității în Mod A;
110 – execută rotirea priorității în Mod B. Nivelul cu prioritate minimă este stabilit de biții 2,1,0;
111 – terminare întrerupere şi execută rotirea priorității în Mod B. Nivelul cu prioritate minimă este stabilit de biții 2,1,0;

Figura 7.4. a. b. Cuvintele de comandă al operării.

Modul special de tratare cu mascare este stabilit prin CCO3 şi se referă la situația în care unii biți sunt mascați prin conținutul RMI, stabilit cu ajutorul CCO1. Dacă, din diferite motive, se execută subrutina care are nivelul mascat (fie datorită recepționării semnalului $\overline{\text{INTA}}$ după mascare, fie datorită automascării), folosind acest mod, nivelurile mai puțin prioritare nu vor fi afectate până la anularea acestui mod. Nivelurile prioritare, superioare nivelului tratat, de asernenea, nu vor fi afectate.
Modul de tratare cu interogare se bazează pe dezactivarea, de către unitatea centrală, a Intrărilor de întrerupere. Servirea echipamentelor se realizează prin comanda de interogare, la inițiativa programatorului, folosind CCO3 cu bitul P=l, pe durata

impulsului $\overline{WR}$. Cu ocazia generării de către unitatea centală a unui impuls $\overline{RD}$, unitatea 8259 îl tratează ca semnal de recunoaştere a întreruperii, poziționând în unu, în cazul unei cereri de întrerupere, bistabilul corespunzator din RCIT şi citind nivelul de prioritate asociat. Unitatea 8259 va forța pe magistrala de date un cuvânt, care va avea în biți D_2, D_l, D_0, codul cererii curente de întrerupere cu prioritatea cea mai mare, iar în bitul D_7 se va indica prezența (1) sau absența (0) unei cereri de întrerupere.

c) Cuvântul de comandă al operării CCO3. Mod special de mascare.

A0	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	x	CSMM	SMM	0	1			

biții 1,0

00 – nepermis;
01 – nepermis;
10 – citeşte RCI la următoarea comandă $\overline{RD}$;
11 – citeşte RCIT la următoarea comandă $\overline{RD}$.

bitul 2

1 – comandă pentru modul interogare;
0 – comandă pentru modul lipsă interogare (normal)

biții 6,5

00 – neoperațional;
01 – neoperațional;
10 – anulează modul special de tratare cu mascare;
11 – activează modul special de tratare cu mascare;

x – indiferent.

Figura 7.4. c. Cuvintele de comandă al operării.

Citirea stării unității 8259 se realizează prin generarea unui impuls $\overline{RD}$, după ce, în prealabil, s-a transmis o comandă corespunzătoare CCO3. Astfel, se pot citi RCI, RCIT, poziționând în mod corespunzător biții D_l, D_0, din CCO3. Pentru citirea RMI nu este necesar un CCO3. Informația citită, de pe magistrala de date, va corespunde lui RMI, dacă $\overline{RD}$ este activ şi $A_0 = 1$.

7.3. Întreruperile microcontrolerului TMS320F240

Microcontrolerul TMS320F240 dispune de un mecanism evoluat de gestiune a întreruperilor prin intermediul unui manager de evenimente.

În continuare se prezintă succint acest mod de tratare a întreruperilor datorită faptului că se consideră reprezentativ pentru microcontrolere.

7.3.1. Întreruperile managerului de evenimente (EV)

Organizarea întreruperilor unității centrale

Sistemul de întreruperi externe ale unității centrale (miezul procesorului C2xx - CPU) este format din şase întreruperi mascabile (INT1-INT6) şi una nemascabilă (NMI). Prioritatea maximă o are întreruperea NMI iar prioritățile întreruperilor mascabile scad de la INT6 la INT1, INT1 având prioritatea minimă.

Fiecărei întreruperi îi corespunde un bit în registrul fanioanelor de întrerupere a unității centrale (IFR) şi fiecărei întreruperi mascabile îi corespunde un bit în registrul măştilor unității centrale (IMR). O întrerupere mascabilă este mascată (nu va genera o întrerupere către miez) când bitul corespunzător în IMR este 0. De asemenea miezul procesorului mai are un bit pentru mascarea generală a întreruperilor (INTM) în registrul de stare ST0. Când INTM este setat la 1 toate întreruperile mascabile sunt dezactivate.

Răspunsurl la întrerupere a miezului C2xx

Când apare o tranziție de la unu la zero pe o intrare de întrerupere a miezului, bitul corespunzător al fanionului din IFR este setat în 1. O întrerupere este generată către miez dacă aceasta nu este mascată - întreruperile globale sunt permise (INTM=0) - şi nu există o altă întrerupere nemascată de prioritate mai mare în aşteptare (acest lucru însemnând că nu există nici un fanion setat al unei întreruperi nemascate de prioritate mai mare). Fanionul este şters de către hardware o dată ce cererea de întrerupere este trimisă către miez. Un fanion de întrerupere poate fi de asemenea şters de către programul utilizator scriind un 1 în bitul corespunzător.

Cererea de întrerupere EV şi servirea acestora

Întreruperile managerului de evenimente sunt organizate în trei grupe: A, B şi C. Grupul A generează o cerere de întrerupere către miez prin INT2 iar B şi C prin INT3 şi respectiv INT4. În tabelul 7.1 sunt prezentate toate întreruperile EV, prioritățile acestora şi modul de grupare.

TABELUL 7.1.

Grup	Întrerupere	Prioritatea în interiorul grupului	Vectorul (ID)	Descriere/sursă
A	PDPINT	1 cea mai mare	0020h	Întrerupere de protecție a acționărilor de putere
	CMP1INT	2	0021h	Întreruperea unității de comparare completă 1
	CMP2INT	3	0022h	Întreruperea unității de comparare completă 2
	CMP3INT	4	0023h	Întreruperea unității de comparare completă 3
	SCMP1INT	5	0024h	Întreruperea unității de comparare simplă 1
	SCMP2INT	6	0025h	Întreruperea unității de comparare simplă 2
	SCMP3INT	7	0026h	Întreruperea unității de comparare simplă 3
	T1PINT	8	0027h	Întrerupere timer GP 1 la perioadă
	T1CINT	9	0028h	Întrerupere timer GP 1 la comparare
	T1UFINT	10	0029h	Întrerupere timer GP 1 la depăşire inferioară

TABELUL 7.1. (continuare)

	T1OFINT	11 cea mai mică	002Ah	Întrerupere timer GP 1 la depăşire superioară
B	T2PINT	1 cea mai mare	002Bh	Întrerupere timer GP 2 la perioadă
	T2CINT	2	002Ch	Întrerupere timer GP 2 la comparare
	T2UFINT	3	002Dh	Întrerupere timer GP 2 la depăşire inferioară
	T2OFINT	4	002Eh	Întrerupere timer GP 2 la depăşire superioară
	T3PINT	5	002Fh	Întrerupere timer GP 3 la perioadă
	T3CINT	6	0030h	Întrerupere timer GP 3 la comparare
	T3UFINT	7	0031h	Întrerupere timer GP 3 la depăşire inferioară
	T3OFINT	8 cea mai mică	0032h	Întrerupere timer GP 3 la depăşire superioară
C	CAP1INT	1 cea mai mare	0033h	Întrerupere a unității de captură 1
	CAP2INT	2	0034h	Întrerupere a unității de captură 2
	CAP3INT	3	0035h	Întrerupere a unității de captură 3
	CAP4INT	4 cea mai mică	0036h	Întrerupere a unității de captură 4

Există câte un registru al fanioanelor de întrerupere EV pentru fiecare grup: EVIFRA, EVIFRB şi EVIFRC. Avem de asemenea câte un registru de mascare pentru fiecare grup: EVIMRA, EVIMRB şi EVIMRC. Un fanion în EVIFRx (x=A,B sau C) este mascat (nu va genera o cerere de întrerupere către miez) dacă bitul corespunzător din EVIMRx este zero.

Există de asemenea câte un registru al vectorilor de întrerupere (EVIVRx, x=A, B sau C) asociat cu fiecare grup de întreruperi EV. Registrul vectorului de întrerupere corespunzător poate fi citit de către o rutină de servire a întreruperii (ISR) atunci când o cerere de întrerupere generată de către un grup de întreruperi este acceptată de către miez. Valoarea (vectorul) din registrul vectorului de întrerupere identifică care este întreruperea nemascată şi în aşteptare din grup ce are cea mai mare prioritate.

Generarea întreruperii

Când apare o întrerupere în modulul EV, fanionul corespunzător din registrul de înreruperi EV este setat în unu. O cerere de întrerupere este generată către CPU de către grupul de întreruperi dacă fanionul de întreruperi este setat şi grupul de întreruperi este nemascat la nivelul EV şi întreruperea corespunzătoare din CPU este nemascată la nivelul CPU.

Vectorii de întrerupere

Vectorul de întrerupere al unui grup de întreruperi al EV poate fi citit după ce o cerere de întrerupere a fost generată de către grupul respectiv către miez. Când acest lucru se întâmplă, vectorul de întrerupere (ID) corespunzător fanionului de întrerupere cu cea mai mare prioritate dintre toate fanioanele setate este încărcat în acumulator. Fanionul este şters când vectorul de întrerupere este citit. De asemenea fanionul întreruperii poate fi şters prin scrierea valorii unu direct în fanionul întreruperii.

O valoare zero este întoarsă atunci când registrul de întreruperi al unui grup al EV este citit şi nu există nici un fanion setat şi nemascat în grupul respectiv. Acest lucru permite identificarea întreruperilor rătăcite (necunoscute) ca întreruperi EV.

Servirea întreruperii

După ce o cerere de întrerupere EV este recepționată, registrul EVIVRx poate fi citit în acumulator şi deplasat la stânga cu unul sau mai mulți biți. După aceea o adresă ofset (adresa de început – de intrare - a tabelei de intrări în rutinele de întreruperi) poate fi adunată acumulatorului. O instrucțiune BACC poate fi utilizată pentru a sări la o intrare în tabel. Un alt salt găsit în tabel ne duce la rutina de servire a întreruperii (ISR) pentru sursa specificată (specifică - cea care a solicitat întreruperea). Această procedură produce o întârziere a servirii întreruperii tipică de 20 cicluri CPU (25 dacă este necesară salvarea unui context minim a stării maşinii) din momentul în care întreruperea a fost generată şi până când prima instrucțiune a ISR pentru sursa specifică este executată. Această întârziere poate fi redusă la 8 cicluri CPU dacă o singură întrerupere este permisă într-un grup EV. Dacă spațiul de memorie nu este o problemă, întârzierea poate fi redusă la 16 cicluri CPU fără a fi necesară folosirea unei singure întreruperi pe grup.

Registrele fanioanelor de întrerupere al EV

Registrele sunt toate tratate ca memorii de 16 biți. Biții neutilizați întorc zero la citire. Scrierea biților neutilizați nu are nici un efect. Din cauză că regiştrii EVIFRx pot fi citiți, apariția unei întreruperi poate fi monitorizată de program prin citirea repetată a registrului EVIFRx adecvat, atunci când întreruperea este mascată.

Registrul A al fanioanelor de întrerupere al EV

EVIFRA – adresa 742Fh

15-11	10	9	8
Rezervat	T1OFINT Flag	T1UFINT	T1CINT
	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
T1PINT	SCMP3INT	SCMP2INT	SCMP1INT	CMP3INT	CMP2INT Flag	CMP1INT	PDPINT
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Biții 15-11 **Rezervați. La citire întorc 1 la scriere nu au efect**

Bit 10 **T1OFINT Flag. Întrerupere timer GP1 la depăşire superioară**
Citire: **0 = Fanionul este resetat**
1 = Fanionul este setat
Scriere: **0 = Fără efect**
1 = Resetează fanionul

Bit 9 **T1UFINT. Întrerupere timer GP1 la depăşire inferioară**
Citire: **0 = Fanionul este resetat**
1 = Fanionul este setat
Scriere: **0 = Fără efect**

1 = Resetează fanionul

Bit 8 T1CINT. Întrerupere timer GP1 la comparare
Citire: 0 = Fanionul este resetat
1 = Fanionul este setat
Scriere: 0 = Fără efect
1 = Resetează fanionul

Bit 7 T1PINT. Întrerupere timer GP1 la perioadă
Citire: 0 = Fanionul este resetat
1 = Fanionul este setat
Scriere: 0 = Fără efect
1 = Resetează fanionul

Bit 6 SCMP3INT. Întrerupere la comparator simplu 3
Citire: 0 = Fanionul este resetat
1 = Fanionul este setat
Scriere: 0 = Fără efect
1 = Resetează fanionul

Bit 5 SCMP2INT. Întrerupere la comparator simplu 2
Citire: 0 = Fanionul este resetat
1 = Fanionul este setat
Scriere: 0 = Fără efect
1 = Resetează fanionul

Bit 4 SCMP1INT. Întrerupere la comparator simplu 1
Citire: 0 = Fanionul este resetat
1 = Fanionul este setat
Scriere: 0 = Fără efect
1 = Resetează fanionul

Bit 3 CMP3INT. Întrerupere la comparator complet 3
Citire: 0 = Fanionul este resetat
1 = Fanionul este setat
Scriere: 0 = Fără efect
1 = Resetează fanionul

Bit 2 CMP2INT. Întrerupere la comparator complet 2
Citire: 0 = Fanionul este resetat
1 = Fanionul este setat
Scriere: 0 = Fără efect
1 = Resetează fanionul

Bit 1 CMP1INT. Întrerupere la comparator complet 1
Citire: 0 = Fanionul este resetat
1 = Fanionul este setat
Scriere: 0 = Fără efect
1 = Resetează fanionul

Bit 0 PDPINT. Întrerupere la protecția circuit de comandă de putere
Citire: 0 = Fanionul este resetat
1 = Fanionul este setat
Scriere: 0 = Fără efect
1 = Resetează fanionul

Registrul B al fanioanelor de întrerupere al EV

EVIFRB – adresa 7430h

15-8	7	6	5	4	3	2	1	0
Rezervat	T3OFINT Flag	T3UFINT	T3CINT	T3PINT	T2UFINT	T2OFINT	T2CINT	T2PINT
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Biții 15-8 **Rezervați. La citire întorc 1 la scriere nu au efect**

Bit 7 **T3OFINT Flag. Întrerupere timer GP 3 la depăşire superioară**
Citire: **0 = Fanionul este resetat**
1 = Fanionul este setat
Scriere: **0 = Fără efect**
1 = Resetează fanionul

Bit 6 **T3UFINT. Întrerupere timer GP 3 la depăşire inferioară**
Citire: **0 = Fanionul este resetat**
1 = Fanionul este setat
Scriere: **0 = Fără efect**
1 = Resetează fanionul

Bit 5 **T3CINT. Întrerupere timer GP 3 la comparare**
Citire: **0 = Fanionul este resetat**
1 = Fanionul este setat
Scriere: **0 = Fără efect**
1 = Resetează fanionul

Bit 4 **T3PINT. Întrerupere timer GP 3 la perioadă**
Citire: **0 = Fanionul este resetat**
1 = Fanionul este setat
Scriere: **0 = Fără efect**
1 = Resetează fanionul

Bit 3 **T2OFINT Flag. Întrerupere timer GP 2 la depăşire superioară**
Citire: **0 = Fanionul este resetat**
1 = Fanionul este setat
Scriere: **0 = Fără efect**
1 = Resetează fanionul

Bit 2 **T2UFINT. Întrerupere timer GP 2 la depăşire inferioară**
Citire: **0 = Fanionul este resetat**
1 = Fanionul este setat
Scriere: **0 = Fără efect**
1 = Resetează fanionul

Bit 1 **T2CINT. Întrerupere timer GP 2 la comparare**
Citire: **0 = Fanionul este resetat**
1 = Fanionul este setat
Scriere: **0 = Fără efect**
1 = Resetează fanionul

Bit 0 **T2PINT. Întrerupere timer GP 2 la perioadă**

Citire: **0 =** **Fanionul este resetat**
1 = **Fanionul este setat**
Scriere: **0 =** **Fără efect**
1 = **Resetează fanionul**

Registrul C al fanioanelor de întrerupere al EV

EVIFRC – adresa 7431h

15-4	3	2	1	0
Rezervat	CAP4INT Flag	CAP3INT	CAP2INT	CAP1INT
	RW-0	RW-0	RW-0	RW-0

Biții 15-4 **Rezervați. La citire întorc 1 la scriere nu au efect**

Bit 3 **CAP4INT Flag. Întrerupere captură 4**
Citire: **0 =** **Fanionul este resetat**
1 = **Fanionul este setat**
Scriere: **0 =** **Fără efect**
1 = **Resetează fanionul**

Bit 2 **CAP3INT. Întrerupere captură 3**
Citire: **0 =** **Fanionul este resetat**
1 = **Fanionul este setat**
Scriere: **0 =** **Fără efect**
1 = **Resetează fanionul**

Bit 1 **CAP2INT. Întrerupere captură 2**
Citire: **0 =** **Fanionul este resetat**
1 = **Fanionul este setat**
Scriere: **0 =** **Fără efect**
1 = **Resetează fanionul**

Bit 0 **CAP1INT. Întrerupere captură 1**
Citire: **0 =** **Fanionul este resetat**
1 = **Fanionul este setat**
Scriere: **0 =** **Fără efect**
1 = **Resetează fanionul**

Registrul A de mascare a întreruperilor EV

EVIMRA – adresa 742Ch

15-11	10	9	8
Rezervat	T1OFINT ENABLE	T1UFINT ENABLE	T1CINT ENABLE
	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
T1PINT ENABLE	SCMP3INT ENABLE	SCMP2INT ENABLE	SCMP1INT ENABLE	CMP3INT ENABLE	CMP2INT ENABLE	CMP1INT ENABLE	PDPINT ENABLE
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Biții 15-11 **Rezervați. La citire întorc 1 la scriere nu au efect**

Bit 10 **T1OFINT ENABLE**
0 = Dezactivare
1 = Activare

Bit 9 **T1UFINT enable**
0 = Dezactivare
1 = Activare

Bit 8 **T1CINT ENABLE**
0 = Dezactivare
1 = Activare

Bit 7 **T1PINT ENABLE**
0 = Dezactivare
1 = Activare

Bit 6 **SCMP3INT ENABLE**
0 = Dezactivare
1 = Activare

Bit 5 **SCMP2INT ENABLE**
0 = Dezactivare
1 = Activare

Bit 4 **SCMP1INT ENABLE**
0 = Dezactivare
1 = Activare

Bit 3 **CMP3INT ENABLE**
0 = Dezactivare
1 = Activare

Bit 2 **CMP2INT ENABLE**
0 = Dezactivare
1 = Activare

Bit 1 **CMP1INT ENABLE**
0 = Dezactivare
1 = Activare

Bit 0 **PDPINT ENABLE**
0 = Dezactivare
1 = Activare

Registrul B de mascare a întreruperilor EV

EVIMRB – adresa 742Dh

15-8	7	6	5	4	3	2	1	0
Rezervat	T3OFINT ENABLE	T3UFINT ENABLE	T3CINT ENABLE	T3PINT ENABLE	T2UFINT ENABLE	T2OFINT ENABLE	T2CINT ENABLE	T2PINT ENABLE
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Biții 15-8 **Rezervați. La citire întorc 1 la scriere nu au efect**

Bit 7 **T3OFINT ENABLE**
0 = Dezactivare
1 = Activare

Bit 6 **T3UFINT ENABLE**
0 = Dezactivare
1 = Activare

Bit 5 **T3CINT ENABLE**
0 = Dezactivare
1 = Activare

Bit 4 **T3PINT ENABLE**
0 = Dezactivare
1 = Activare

Bit 3 **T2OFINT ENABLE**
0 = Dezactivare
1 = Activare

Bit 2 **T2UFINT ENABLE**
0 = Dezactivare
1 = Activare

Bit 1 **T2CINT ENABLE**
0 = Dezactivare
1 = Activare

Bit 0 **T2PINT ENABLE**
0 = Dezactivare
1 = Activare

Registrul B de mascare a întreruperilor EV

EVIMRC – adresa 742Eh

15-4	3	2	1	0
Rezervat	CAP4INT ENABLE	CAP3INT ENABLE	CAP2INT ENABLE	CAP1INT ENABLE
	RW-0	RW-0	RW-0	RW-0

Biții 15-4 **Rezervați. La citire întorc 1 la scriere nu au efect**

Bit 3 **CAP4INT ENABLE**
0 = Dezactivare
1 = Activare

Bit 2 **CAP3INT ENABLE**
0 = Dezactivare
1 = Activare

Bit 1 **CAP2INT ENABLE**
0 = Dezactivare
1 = Activare

Bit 0 **CAP1INT ENABLE**
0 = Dezactivare
1 = Activare

Registrul A al vectorului de întrerupere EV

EVIVRA – adresa 7432h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Biții 15-6 **Rezervați. La citire întorc zero, la scriere nu au efect.**

Biții 5-0 **D5-D0. Vector (ID) al fanionului de întrerupere care are cea mai mare prioritate din toate fanioanele setate şi nemascate din EVIFRA; valoare zero dacă nici un fanion nemascat nu este setat în EVIFRA.**

Registrul B al vectorului de întrerupere EV

EVIVRB – adresa 7433h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Biții 15-6 **Rezervați. La citire întorc zero, la scriere nu au efect.**

Biții 5-0 **D5-D0. Vector (ID) al fanionului de întrerupere care are cea mai mare prioritate din toate fanioanele setate şi nemascate din EVIFRB; valoare zero dacă nici un fanion nemascat nu este setat în EVIFRB.**

Registrul C al vectorului de întrerupere EV

EVIVRC – adresa 7434h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Biții 15-6 **Rezervați. La citire întorc zero, la scriere nu au efect.**

Biții 5-0 **D5-D0. Vector (ID) al fanionului de întrerupere care are cea mai mare prioritate din toate fanioanele setate şi nemascate din EVIFRC; valoare zero dacă nici un fanion nemascat nu este setat în EVIFRC.**

Resetarea fanionului unei întreruperi se face prin scrierea valorii 1 în registrul EVIFRx (x = A, B sau C) sau prin citirea vectorului de întrerupere din registrul EVIVRx (x = A, B sau C). Acest lucru este important pentru că semnalează achitarea întreruperii şi permite generarea întreruperii următoare.

7.4. Accesul direct la memorie (DMA)

Accesul direct la memorie, Direct Memory Access (DMA) oferă o cale avantajoasă de transfer de date între un port de interfață al unui echipament periferic şi unitatea de memorie sau între două zone diferite de memorie. Aceste transferuri au o pondere relativ ridicată în activitatea generală a unui calculator şi reducerea timpului în care se realizează transferurile duce la creşterea performanțelor generale ale calculatorului. Spre exemplu transferul DMA se utilizează atunci când este necesară salvarea (stocarea) programelor şi a rezultatelor acestora din memoria RAM pe un suport extern de memorie (disc flexibil, harddisc, bandă magnetică, etc.). De asemenea mecanismul transferurilor DMA este utilizat pentru reîmproaspătarea conținutului memoriilor DRAM, când practic nu se face un transfer real, dar procesul de citire a memoriei DRAM în vederea transferului, realizează reîmprospătarea acestora.

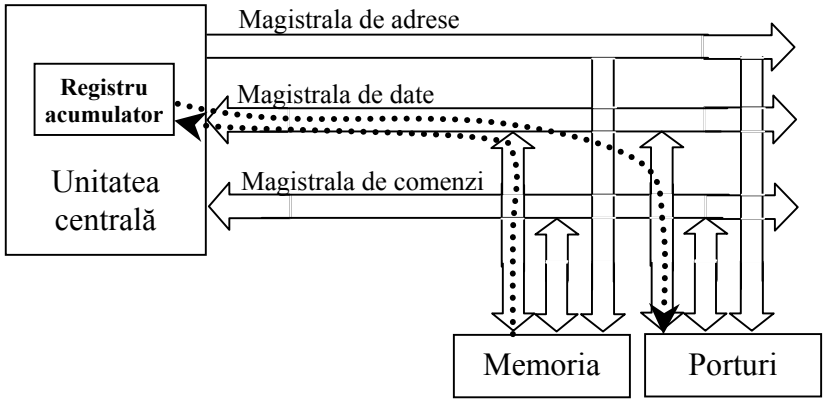


Fig. 7.5. Transferul datelor prin intermediul unității centrale

În principiu, transferul datelor se poate face prin intermediul unității centrale, fără a fi necesară existența unui circuit DMA. În figura 7.5 este prezentat fluxul datelor în cazul în care transferul se face prin intermediul unității centrale. Această metodă este uzuală în cazul sistemelor ieftine dar ea prezintă o serie de dezavantaje care o face nerecomandabilă la sistemele performante. În primul rând, durata transferului este relativ mare din cauză că sunt necesare două activități succesive ale unității centrale: o citire a datei din memorie în unitatea centrală şi o scriere a datei, din memorie în port, iar în al doilea rând este ineficientă ocuparea unității centrale cu o activitate atât de simplă cum este transferul datelor.

În cazul utilizării circuitului DMA performanțele sistemului cresc din mai multe motive: circuitul DMA fiind un circuit specializat pentru astfel de operații, transferul datelor se face mult mai rapid decât în cazul transferul datelor prin intermediul unității

centrale, unitatea centrală este degrevată de astfel de sarcini iar traseul datelor este mai scurt. În figura 7.6 este prezentat fluxul datelor în cazul utilizării circuitului DMA.

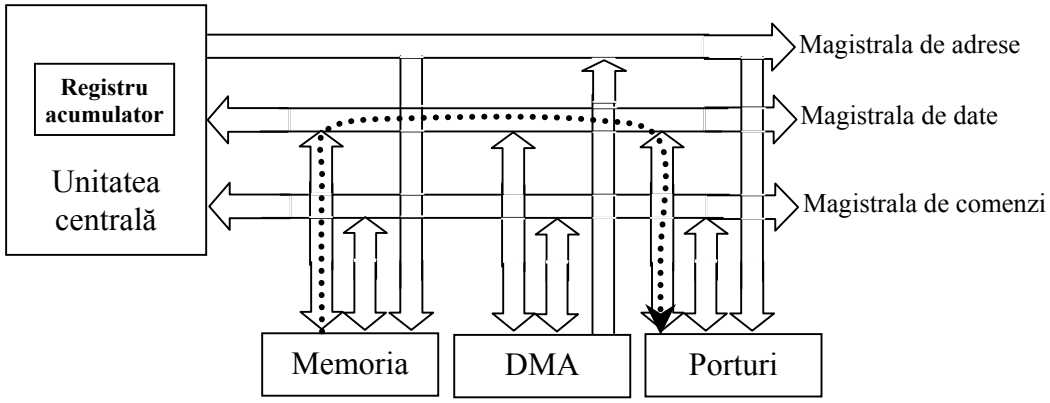


Fig. 7.6. Transferul datelor prin intermediul circuitului DMA

După ce este programat, circuitul DMA generează semnalele de adresă şi de control, pe magistrala de comenzi, în aşa fel încât să se citească datele din memorie şi acestea să fie transferate portului (sau altei zone de memorie). Circuitul DMA este un circuit master ca şi unitatea centrală. Din acest motiv aceste două circuite nu pot lucra simultan ci ele vor prelua pe rând controlul magistralelor sistemului. Motivul pentru care două module master nu pot lucra simultan este că (aşa cum se vede şi din figura 7.6), cele două module generează atât adrese cât şi comenzi. Dacă cele două module ar lucra simultan atunci adresa generată de unitatea centrală în scopul execuţiei programului din memorie va fi diferită de adresa generată de unitatea DMA pentru realizarea transferului şi pe magistrala de adrese (care este comună) ar apărea două adrese diferite ceea ce ar genera un conflict.

Cele două unităţi master (unitatea centrală şi unitatea DMA) îşi suspendă una alteia activitatea, printr-un dialog desfăşurat pe magistrala de comenzi. Dacă circuitul DMA are de făcut un transfer, atunci va solicita unităţii centrale să elibereze magistralele. În momentul în care unitatea centrală poate ceda magistralele semnalează acest lucru modulului DMA şi îşi întrerupe activitatea pe magistrale. La terminarea transferului (sau a unei părţi a acestuia, în funcţie de modul în care este programat circuitul DMA) unitatea centrală este informată printr-un semnal de comandă că poate prelua controlul magistralelor reluându-şi în acest mod activitatea.

În mod aparent, faptul că cele două module master nu pot lucra simultan cu magistralele sistemului, duce la scăderea eficienţei acestora. Unitatea centrală, în mod natural, în execuţia unui program, necesită efectuarea unor activităţi interne (cum ar fi de exemplu executarea unei operaţii matematice) care nu solicită lucrul cu magistralele. În aceste intervale de timp circuitul DMA poate prelua controlul magistralelor fără a scădea viteza de lucru a unităţii centrale. De asemenea, structura ierarhizată a magistralelor şi utilizarea unor magistrale separate, care să permită funcţionarea simultană a circuitului DMA şi a unităţii centrale duc la creşterea eficienţei transferurilor de date. Spre exemplu, controlerul video necesită în general un flux mare de date în mod continuu. Din acest motiv s-a extins un standard de magistrală care să

permită accelerarea transferurilor între memoria principală a sistemului de calcul şi memoria video.

Importanţa acestui mecanism de transfer al datelor a crescut o dată cu creşterea capacităţii memoriilor şi a volumului de date prelucrate în sistem. Din acest motiv, un criteriu de performanţă pentru un calculator este reprezentat şi de numărul modulelor DMA instalate în sistem. Unele circuite utilizează mecanisme DMA fără ca acest lucru să fie specificat explicit, spre exemplu cum este sistemul memoriilor cache.

Un modul DMA are mai multe canale care pot fi programate separat şi care pot lucra cu mai multe periferice simultan. Din acest motiv la unele porturi, în afară de adresă şi de numărul întreruperii alocate se stabileşte şi canalul DMA asociat pentru transferul datelor.

7.4.1. Circuitul 8257 pentru acces direct la memorie – DMA

Accesul direct la memorie. Direct Memory Access, DMA, oferă o cale avantajoasă de transfer de date pe magistrala sistemului între un port de interfaţă al unui echipament periferic şi unitatea de memorie.

În varianta clasică este presupusă intervenţia unităţii centrale, care foloseşte registrele sale interne ca memorie intermediară. Un astfel de transfeer periferic/memorie presupune două faze: transfer port/memorie în acumulatorul unităţii centrale şi apoi încă un transfer din registrul intern al unităţii centrale în memorie/port. Accesul direct la memorie, realizat de un circuit LSI specializat, elimină unitatea centrală de pe magistrala sistemului, dirijând în mod independent transferurile periferic - memorie prin intermediul semnalelor de citire/scriere, semnale generate simultan atât pentru port, cât şi pentru memorie. În plus, circuitul DMA furnizează adresele de memorie la locaţii succesive pentru transferuri pe şir de caractere şi semnalele de selecţie pentru port. Accesul la magistrală se realizează în urma unui dialog cu unitatea centrală folosind semnalele de cerere/cedare magistrală (de exemplu semnalele HOLD/HLDA pentru microprocesorul 8080 sau BUSRQ/BUSAK pentru microprocesorul Z80). Transferurile succesive se pot realiza în pachet, la viteza maximă, sau cu câte o cerere de transfer pentru fiecare octet, în acest din urmă caz, unitatea centrală poate lucra în pauzele dintre transferuri.

Circuitul 8257 oferă funcţia DMA, simultan pe patru canale. Viteza maximă corespunde transferului în pachet a câte unui octet la fiecare 4T unde T este perioada semnalului de sincronizare al unităţii centrale. T nu poate fi mai mic de 0,32 microsecunde (8257) sau 0,25 microsecunde (8257-5).

7.4.1.1. Conexiunile externe

Schema bloc, figura 7.7, evidenţiază aşezarea, în jurul unei magistrale interne, a circuitelor de interfaţă cu unitatea centrală şi a celor pentru dialogul de preluare/eliberare a magistralei acesteia, pe de o parte, şi a registrelor de canal, inclusiv circuitele de dialog cu perifericele şi circuitele pentru stabilirea priorităţii între canale, pe de altă parte. Circuitul, prezentat în figura 7.8, are 40 de conexiuni externe, cu următoarele semnificaţii:

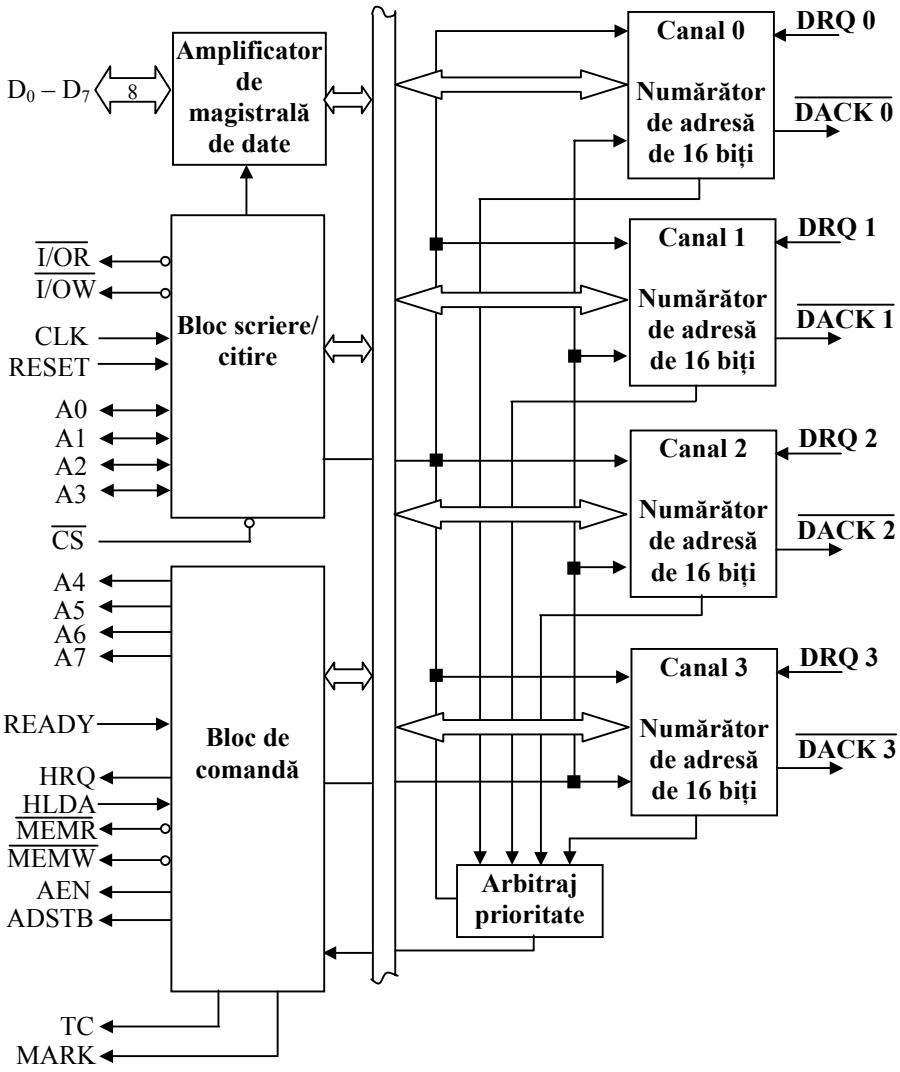


Figura 7.7. Schema bloc a circuitului 8257

$\overline{I/OR}$ (trei stări). Semnal de la/spre magistrala unității centrale pentru eşantionarea datelor citite din port.

$\overline{I/OW}$ (trei stări). Semnal de la/spre magistrala unității centrale pentru cşantionarea datelor scrise în port.

$\overline{MEMR}$ (trei stări). Semnal de la/spre magistrala unității centrale pentru eşantionarea datelor citite din locația de memorie adresată.

$\overline{MEMW}$ (trei stări). Semnal de la/spre magistrala unității centrale pentru eşantionarea datelor scrise în locația de memorie adresată.

MARK (ieşire). Semnal emis la fiecare al 128-lea octet transferat.

READY (intrare). Semnal care permite introducerea unor stări de aşteptare, SW, înaintea încheierii transferului, având port-ul şi locația de memore selectate.

HLDA (intrare). Semnal primit de pe magistrala unității centrale, semnificând că aceasta s-a eliberat, în vederea transferurilor DMA.

ADDSTB (ieşire). Semnal de eşantionare pentru încărcarea unui port adițional, cu octetul cel mai semnificativ al adresei de memorie.

AEN (ieşire). Semnal care semnifică ocuparea magistralei unității centrale pentru accese DMA. Validează emiterea octetului cel mai semnificativ de adresă din portul adițional.

HRQ (ieşire). Semnal de cerere a magistralei, în vederea eliminării unității centrale.

$\overline{CS}$ (intrare). Semnal de selecție pentru porturile circuitului 8257. Invalid în timpul ciclilor DMA.

CLK (intrare). Semnal de sincronizare. Coincide cu semnalul de tact al unității centrale.

RESET (intrare). Semnal de inițializare. 8257 intră în starea S1.

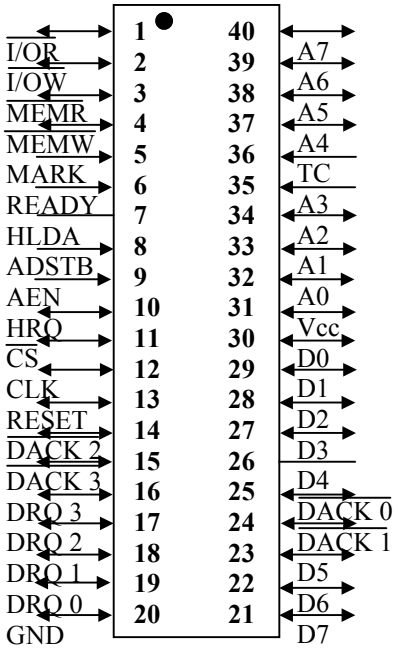


Figura 7.8. Conexiunile externe ale circuitului 8257.

$\overline{DACK\ 2}$ (ieşire). Semnal de selecție pentru portul corespunzator canalului 2. Se emite după obținerea magistralei în vederea selctării portului în vederea efectuării transferului.

$\overline{DACK\ 3}$ (ieşire), Idem, pe canalul 3.

DRQ 3 (intrare). Semnal prin care se semnalizează circuitului DMA că este necesar un transfer pe canalul 3.

DRQ 2 (intrare). Idem, pe canalul 2.

DRQ 1 (intrare). Idem, pe canalul 1.

DRQ 0 (intrare). Idem, pe canalul 0.

GND. Masa tensiunii de alimentare (0V).

(D_7-D_0) (trei stări). Semnale de date pe magistrala unității centrale.

$\overline{DACK1}$ (ieşire) Semnal identic cu $\overline{DACK\ 2}$, pentru canalul 1.

$\overline{DACK0}$ (ieşire). Idem, canalul 0.

VCC Sursa de aUmentare (+5V).

A_0-A_7 (trei stări). Octetul cel mai puțin semnificativ al liniilor de adresare ale magistralei unității centrale.

TC(0). Semnalează sfârşitul transferului sirului de octeți. Se emite în timpul ultimului transfer.

7.4.1.2. Registrele interne ale 8257

8257 conține câte 2 registre bidirecționale de 16 biți pentru fiecare dintre cele 4 canale şi încă două registre unidirecționale, pentru comanda şi examinarea funcționării circuitului, Semnificația fiecărei poziții din registre rezultă din tabelul 7.2.

TABELUL 7.2.

Registru	Octet	Adresa $A_3…A_2$	Bistabil intern	Conținut							
				D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Adreesă canal 0	Cmps Cms	0000 0000	0 1	A_7 A_{15}	A_6 A_{14}	A_5 A_{13}	A_4 A_{12}	A_3 A_{11}	A_2 A_{10}	A_1 A_9	A_0 A_8
Numărător de octeți canal 0	Cmps Cms	0001 0001	0 1	C_7 S	C_6 C	C_5 C_{13}	C_4 C_{12}	C_3 C_{11}	C_2 C_{10}	C_1 C_9	C_0 C_8
Adreesă canal 1	Cmps Cms	0010 0010	0 1	A_7 A_{15}	A_6 A_{14}	A_5 A_{13}	A_4 A_{12}	A_3 A_{11}	A_2 A_{10}	A_1 A_9	A_0 A_8
Numărător de octeți canal 1	Cmps Cms	0011 0011	0 1	C_7 S	C_6 C	C_5 C_{13}	C_4 C_{12}	C_3 C_{11}	C_2 C_{10}	C_1 C_9	C_0 C_8
Adreesă canal 2	Cmps Cms	0100 0100	0 1	A_7 A_{15}	A_6 A_{14}	A_5 A_{13}	A_4 A_{12}	A_3 A_{11}	A_2 A_{10}	A_1 A_9	A_0 A_8
Numărător de octeți canal 2	Cmps Cms	0101 0101	0 1	C_7 S	C_6 C	C_5 C_{13}	C_4 C_{12}	C_3 C_{11}	C_2 C_{10}	C_1 C_9	C_0 C_8
Adreesă canal 3	Cmps Cms	0110 0110	0 1	A_7 A_{15}	A_6 A_{14}	A_5 A_{13}	A_4 A_{12}	A_3 A_{11}	A_2 A_{10}	A_1 A_9	A_0 A_8
Numărător de octeți canal 3	Cmps Cms	0111 0111	0 1	C_7 S	C_6 C	C_5 C_{13}	C_4 C_{12}	C_3 C_{11}	C_2 C_{10}	C_1 C_9	C_0 C_8
Mod	-	1000	-	AL	TCS	EV	EP	EN3	EN2	EN1	EN0
Stare	-	1000	-	0	0	0	UP	TC3	TC2	TC1	TC0

Simboluri folosite:

Cmps = cel mai puțin semnificativ octet;
Cms = cel mai semnificativ octet;
C = citire port, generează $\overline{I/OR}$, $\overline{MEMW}$;
S = scriere port, generează $\overline{I/OW}$, $\overline{MEMR}$;
AX = bit adresă de start transfer, X= 0, …, 15;
CX = bit de inițializare a numărului de octeți, X=0, …, 13.

7.4.1.3. Registrele de canal

Fiecăruia dintre cele 4 canale îi corespunde un registru pentru adresa de memorie RAM de la care începe transferul, într-un spațiu de 64 Kocteți şi încă un registru de 16 biți, ale cărui prime 14 locații au funcția de numărător, pentru până la 16 Kocteți. iar celelalte 2 locații biții 14 şi 15, monitorizează generarea semnalelor $\overline{\text{I/OR}}$, $\overline{\text{I/OW}}$, $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$. Adresa şi numărul de octeți evoluează la fiecare transfer DMA pe canalul respectiv, iar ultimul transfer este semnalat ca închciere a transferului, Terminal Count, TC, pe o conexuine exterioară a circuitului. Semnalele de citire/scriere corespund combinațiilor:

	S	C	$\overline{MEMR}$	$\overline{MEMW}$	$\overline{I/OR}$	$\overline{I/OW}$
Verificare	0	0	1	1	1	1
Citire port	0	1	1	0	0	1
Scriere port	1	0	0	1	1	0
Invalid	1	1	-	-	-	-

Fiecare registru de 16 biți este accesibil printr-o dublă adresare la câte 8 biți, folosind un bistabil intern, care selectează între octetul cel mai puțin semnificativ şi cel mai semnificativ.

7.4.1.4. Registrul de mod

Registrul asigură validarea canalelor în lucru şi a 4 opțiuni, asupra modului de desfăşurare a dialogului pe magistrală.

7	6	5	4	3	2	1	0
AL	TCS	EW	RP	EN3	EN2	EN1	EN0

Registrul este înscris, de regulă, după completarea celor 2 registre de 16 biți ale fiecărui canal cu care se lucrează. La RESET registrul de control este şters, inhibându-se toate canalele şi opțiunile, astfel prevenindu-se transferuri DMA medorite la aplicarea tensiunii.

EN0…3: Enable Channel, corespunde validării canalului pe care se doreşte să se efectueze transferuri. Se pot face transfcniri pe 1…4 canale. Pcntru mai mult de un canal, ordinea de servire corespunde unei scheme de preiorități, stabilite de bitul 4.

RP: Rotating Priority, corospunde stabilirii schemei de priorități, între două variante: prioritate fixă sau prioritate circulară. În varianta fixă se alocă o prioritate maximă canalulni 0, prioritate care descrcşte până la a fi minimă pentru canalul 3. În varianta circulară, se alocă dinamic nivelul de prioritate al fiecărui canal, după fiecare ciclu de transfer. În aecst caz, fiecare canal servit va trece ultimul pe lista de priorități, în arbitrarea ciclului DMA următor.

Alegerea modului RP previne fenomenul de „gîtuire", bottleneck, când unul dintre canale monopolizează transferurile în mod DMA. Dacă sunt prezentate cereri de transfer pe mai multe canale, se vor servi toate canalele, pe rând. Primul transfer se va executa conform schemei fixe de priorități.

EW: Extended Write, contribuie la generarea anticipată a semnalelor de scriere, pentru compensarea unor eventuali timpi de acces mai lungi ai participanților la transfer. În absența acestei opțiuni, ar fi fost necesare stari de aşteptare, folosind semnalul READY de la conexiunea exterioară 6 ceea ce ar fi micşorat viteza de lucru.

TCS: Terminal Count Stop, specifică oprirea sau continuarea transferurilor DMA, din momentul când s-a emis ,,1" pe conexiunea exterioară TC, simultan cu poziționarea bitului corespunzător canalului, în registrul de stare. Bitul de validare a canalului este şters, iar operațiile DMA pe acest canal se pot relua după o rescriere în registrul de control. În cazul în care nu se programează bitul TCS, transferurile DMA continuă, la adrese succcsive, cât timp perifericul emite DRQ.

AL: Autoload, autoîncărcare permite, când este emis să, să se folosească înlănțuirea de comenzi, fără intervenția unității centrale. Opțiunea se aplică numai canalului 2, utilizând canalul 3 ca registru cu parametri de reinițializare. Canalul 2 se programează obişnuit, pentru primul set de parametri, iar canalul 3 se programează cu setul de parametri pentru transferul ulterior. După ce se efectuează ciclurile DMA conform programării inițiale a canalului 2, se efectuează automat o trecere a parametrilor din canalul 3 în canalul 2. Opțiunea TCS nu are efect, dacă a fost prevăzută simultan cu opțiunca AL. În cazul existenței opțiunii AL, programarea canalului 2 este automat duplicata şi în registrele canalului 3, ceea ce permite operații repetate pe bloc, prin programarea unui singur canal. Se pot înscrie parametri separați pentru cele două canale, dacă se programează întâi canalul 2 şi apoi canalul 3. De observat că, dacă este validat şi canalul 3 şi apar cereri DRQ pe acest canal, se pot efectua transferuri simultane pe canalele 2 şi 3, insă, la operația de reîncărcare a canalului 2 se vor încărca parametrii modificați ai canalului 3, cu o diferență dată de numărul de transferuri DMA ce s-au efectuat pe acest din urmă canal. La intrarea într-un ciclu, după reinițializarea canalului 2, se poziționează bitul UP în registrul de stări. Reinițializarea nu distruge parametrii canalului 3.

Reinițializarea se efectuează la următorul ciclu DMA după un ciclu TC, care va fi astfel primul ciclu cu noii parametri. Bitul UP din registrul de stare se va şterge după acest prim ciclu DMA. În operații înlănțuite, bitul UP se poate examina de unitatea centrală pentru a semnala reinițializrea şi a permite înscrierea noilor parametri în canalul 3, pentru urmatorul bloc ce se va transfera pe canalul 2.

7.4.1.5. Registrul de stare

Registrul indică pe care dintre canale s-a terminat transferul numărului de octeți programat, inclusiv semnalarea stării de inceput a unui bloc de transfer cu parametri reinițializați (numai pentru canalul 2).

7	6	5	4	3	2	1	0
0	0	0	UP	TC3	TC2	TC1	TC0

TC0…3: Terminal Count, sfârşit de numărare, corespunde stării conexiunii exterioare TC, numai că aici se specifică şi numărul de ordine al canalului pe care a avut loc cvenimentul. Biții rămîn înscrişi până la prima citire a registrului de stare, sau până la RESET. Deoarcce, între citirea stării şi înscrierea de parametri, poate apărea semnalarea TC, este contraindicată repregramarea canalelor în funcție de semnalările din registrul de stare, cacă între cele două operații succesive nu se blochează transfcrurile DMA.

UP: Update, este emis la primul transfer DMA efectuat conform unor parametri de reinițializare a canalului 2. UP nu se şterge la citire. Se foloseşte pentru a preveni incărcarea unor parametri în canalul 3, înainte ca parametrii precedenți să fie trecuți în canalul 2.

7.4.1.6. Efectuarea transferurilor cu DMA 8257

Funcționarea 8257 corespunde diagramei de stări date în figura 7.9. În urma semnalului RESET se intră în starea S1, în care se aşteaptă lansarea unei cereri de transfer DRQ de la periferice. Ieşirea din S1 şi trecerea în S0 se face la primirea DRQ pe unul din canale şi lansarca cererii HOLD, pentru obținerea magistralei unității centrale. În starea S0 se aşteaptă eliminarea unității centrale de pe magistrală, semnalată prin sosirea semnalului HLDA, după care se intră în starea S1. În acest moment, 8257 emite adresele superioare pe liniile de date, memorate în portul adițional, cu semnalul ADDSTB. Se obțin astfel toate cele 16 linii de adresă pe magistrală şi se intră în starea S2. În S2 începe transferul prin activarea operației de citire şi a unei eventuale scrieri anticipate. Se emite DACK pentru dialogul DRQ-DACK şi se intră în starea S3, în care se activează scrierea şi se fac semnalările MARK şi TC, dacă este cazul. Din această stare se poate intra într-o stare de aşteptare SW, daca nu s-a primit READY. Testarea conexiunii externe READY este eliminată dacă pe canalul respectiv a fost programată o operație de verificare, VERIFY. Din S3 se intră în S4, dacă s-a primit semnalul READY în operațiile de transfer efectiv. Din SW se trece în S4 dacă se primeşte READY. Starea S4 încheie un ciclu DMA, inactivând DACK şi, dacă este cazul, TC şi MARK. În cazul operațiilor cu TCS, se şterge validarea canalului respectiv, dacă a apărut TC. Se stabilesc prioritățile pentru următorul ciclu DMA. Se testeaza DRQ şi HLDA. Dacă DRQ nu se mai emite, se ştergc HOLD după semnalarea căderii HLDA. Daca DRQ se emite în continuare, se păstrează HOLD şi, în cazul în care şi HLDA se menține, se trece în modul de transfer burst, în pachet, prin succesunea de stări S1, S2, S3, S4, S1 etc., evitâdu-se dialogul de conectare/deconectare la magistrale. La pierderea magistralei, situație semnalată de căderea HLDA, sau dacă perifericul nu mai cere transfer, cu DRQ la „0", se sterge HOLD şi se trece în starea S1 pentru ciclul de reconectare S1, S0, S1, S2, S3, S4 etc.

În rezumat, transferul octet cu octet se desfăşoară după cum urmrază (figura 2.25): perifericul cere transfer prin emiterca semnalului DRQ pe unul dintre canale; dacă este prioritar şi validat, canalul începe operațiunea de transfer, lansând HOLD; la recepția HLDA se lansează DACK pe canalul corespunzator, pentru selectarea registrului de date al perifericului; se generează apoi perechea de semnale de cşantionare: $\overline{IOR}$, $\overline{MEMW}$ sau $\overline{IOW}$, $\overline{MEMR}$, în funcție de sensul transferului, dacă nu este specificată operația VERIFY; memoria este adresată la locația specificată în registrul de adrese al canalului, direct de 8257 pe octetul cel mai puțin semnificativ şi prin intermediul portului adițional 8212, pe octetul cel mai semnificativ. După transferul octetului, semnalele de eşantionare şi achitarea cererii, DACK, se inhibă, iar dacă DRQ nu se mai emite înainte de S4, se eliberează magistrala şi se inhibă HOLD.

Funcționarea în mod burst se alege păstrând DRQ şi HLDA după efectuarea transferului, caz in care se incrementează adresa memoriei şi se reiau operațiile pe octet, fără deconectare/reconectare intermediară.

Dacă se cer transferuri simultan pe mai multe canale, procedura corespude transferului în pachet, cu comutarea canalului pe care se emit semnalele DRQ, DACK, în funcție de schema de prioritate programată. Canalul prioritar este stabilit în S4, în funcție de semnalele DRQ prezente.

Dacă HLDA devine inactiv, în prezența DRQ, se eliberează magistrala, inhibându-se HOLD şi se revine în S1, starea după RESET. Dacă READY devine inactiv, esantionat în S3, se inserează stări SW. La activarea READY, din SW se intră în S4. În acest interval semnalele de selectie DACK şi de eşantionare $\overline{IOR/W}$, $\overline{MEMR/W}$ sunt active, prelungind timpul de seleție al portului/memoriei, în vederea acceptării unui timp de acces mai mare.

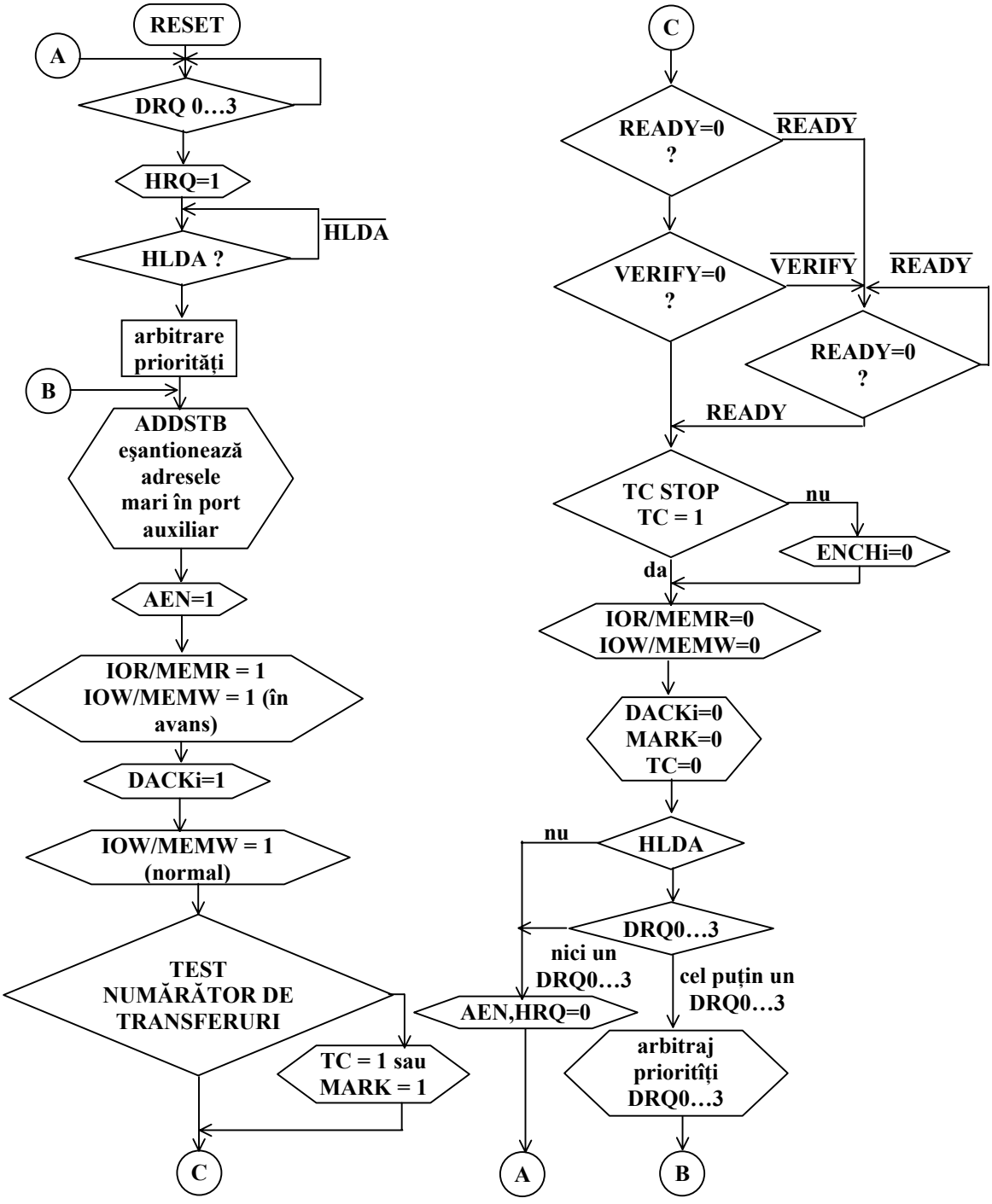
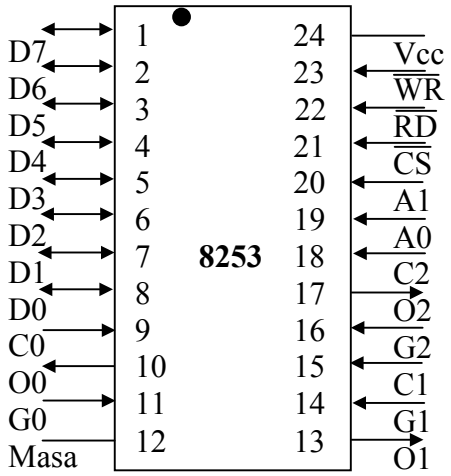


Figura 7.9. Schema logică de funcționare a circuitului 8257.

7.5. Circuitul contor/periodizator programabil 8253



D_7-D_0	- conexiuni la magistrala de date a microsistemului (bidirecțional);
C0	- ceas contor 0 (intrare);
O0	- ieşire contor 0 (ieşire);
G0	- poarta contor 0 (intrare);
C1	- ceas contor 1 (intrare);
O1	- ieşire contor 1 (ieşire);
G1	- poarta contor 1 (intrare);
C2	- ceas contor 2 (intrare);
O2	- ieşire contor 2 (ieşire);
G2	- poarta contor 2 (intrare);
A0, A1	- intrări de selecție contor (intrare);
$\overline{CS}$	- selecție circuit (intrare);
$\overline{RD}$	- comandă citire (intrare);
$\overline{WR}$	- comandă scriere (intrare);
Vcc	- alimentare +5V.
MASA	

Figura 7.10. Semnificația terminalelor circuitului 8253.

Circuitul 8253, realizat în tehnologia NMOS pe o pastilă cu 24 de terminale, ale căror semnificații sunt prezentate în figura 7.10, constitute un contor/periodizator programabil. El este organizat sub forma a 3 contoare independente, de câte 16 biti, având asociată logica corespunzatoare pentru comunicația cu unitatea centrala de prelucrare şi cu mediul exterior. Circuitul este văzut, de unitatea centrală de prelucrare, sub forma unui tablou de porturi de I/E şi poate fi folosit ca: generator programabil de semnale dreptunghiulare, contor de evenimente, ceas de timp real, monostabil numeric, element pentru generarea comenzilor unor motoare pas cu pas. Folosirea lui în sistemele cu microprocesor, în aplicațiile privind prelucrarea unor semnale sub formă de trenuri de impulsuri, simplifică în mod considerabil software-ul necesar acestor prelucrări.

După cum se poate observa în figura 7.11, schema bloc a circuitului constă diu mai multe componente: tamponul magistralei de date, logica scrie/citeşte, registrul cuvâtului de comnandă şi cele trei contoare 0, 1 si 2.

Tamponul magistralei de date este bidirecțional, organizat pe 8 biți, cu elemente cu trei stari, realizează interfața circuitului 8253 cu magistrala de date a unității centrale de prelucrare. Datele sunt transmise sau recepționate de tampon, la execuția instrucțiunilor IN şi OUT, de către unitatea centrală de prelucrare. Prin intermediul acestui tampon se realizează: programarea modurilor de lucru pentru 8253, încărcarea contorilor, citirea valorilor datelor din contori.

Logica scrie/citeşte pe baza semnalelor $\overline{CS}$, $\overline{RD}$, $\overline{WR}$, A_0 şi A_l, primite de la magistrala de legătura cu unitatea centrală de prelucrare, generează semnalele de comandă pentru buna funcționare a întregului circuit. Sermnalul CS, activ pe nivel coborât, activează/dezactivează circuitul astfel încât, dacă circuitul nu este selectat ($\overline{CS}$ = 0), funcționarea lui nu este influențată. Semnalul $\overline{RD}$ activ comandă citirea datelor de la contorii circuitului, în timp ce semnalul $\overline{WR}$ activ asigura Sncarcarea rogistrului de coraanda sail a datelor In contori. Liniile A_0 şi A_l sunt conectate la magistrala de adrese şi servesc la selectarea unuia dintre contori sau a registrului de

comandă. În tabelul 7.3 se prezintă efectele semnalelor amintite mai sus asupra funcționării circuitului.

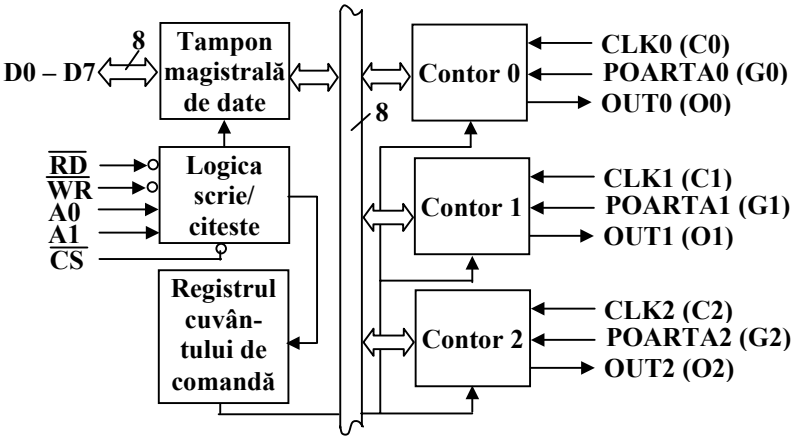


Figura 7.11. Structura internă a circuitului 8253.

TABELUL 7.3.

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A1	A0	Funcția
0	1	0	0	1	Încarcă contorul 0
0	1	0	0	0	Încarcă contorul 1
0	1	0	1	1	Încarcă contorul 2
0	1	0	1	0	Încarcă cuvântul de comandă
0	0	1	0	1	Citeşte contorul 0
0	0	1	0	0	Citeşte contorul 1
0	0	1	1	1	Citeşte contorul 2
0	0	1	1	0	Neoperațional, starea de mare impedanță
1	x	x	x	x	Circuit neselectat, starea de mare impedanță

Registrul cuvântului de comandă primeşte de la magistrală, informația prin care se comandă modul de lucru al fiecărui contor. Acest registru se selecteaza prin adresa $A_1A_0 = 11$. Conținutul său nu poate fi citit.

Contorii 0, 1 şi 2 sunt identici, fiind implementați prin numărătoare de 16 biți, al căror conținut poate fi prestabilit, numărarea efectuându-se în sens descrescator. Contorii sunt independenți şi pot fi programați să opereze în modul binar sau BCD, în diverse configurații, privind intrarea de ceas, poarta de comandă şi citirea. Citirea conținutului umui contor oarecare se realizează direct, în cazul contorizării unor evenimente sau printr-o tehnică specială, în celelalte cazuri, fără a bloca intrarea de ceas.

Interfațarea circuitului 8253 cu sistemul se realizează în maniera obişnuită, a circuitelor de interfață din familia 8080. Intrarile A_0 şi A_l se conectează la liniile de adrese A_0, A_l, ale sistemului, iar semnalul de selecție $\overline{CS}$ poate fi generat din semnalele magistralei de adrese a sistemului, prin selecție liniară sau decodificare. Semrialele $\overline{RD}$, $\overline{WR}$ sunt derivate din semnalele $\overline{IORC}$, $\overline{IOWC}$ ale magistralei sistemului.

Modul de funcționare poate fi stabilit complet prin software, încărcând registrul de comandă cu un cuvânt corespunzator.

Fiecare contor este decrementat cu o unitate pe fiecare front căzător al semnalului aplicat la intrarea de ceas. Semnalul de la intrarea de ceas poate fi asincron sau sincron. In primul caz contorul va fi folosit pentru numărarea unor evenimente. În al doilea caz contorul este folosit pentru generarea unor intervale de timp. Frecvența semnalului aplicat la intrarea de ceas poate varia între 0 si 3MHz.

Intrarea pe poarta G poate activa sau bloca funcționarea contorului/periodizatorului respectiv.

Ieşirea 0 a fiecărui contor, în funcție de modul de programare, poate fi folosită ca impuls singular, cerere de întrerupere sau ca semnal simplu de comandă.

Cuvântul de comandă.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

bitul 0

0 – contorul selectat este tratat ca valoare binară (max. 65535_z);
1 – contorul selectat este tratat ca valoare BCD (max. 9999_{BCD});

biții 3,2,1

000 – Mod 0;
001 – Mod 1;
x10 – Mod 2;
x11 – Mod 3;
100 – Mod 4;
101 – Mod 5;

biții 5,4

00 – operația de citire a contorului, a se vedea procedura READ/WRITE;
01 – citeşte/încarcă octetul cel mai semnificativ (la incărcare anulează octetul cel mai puțin semnificativ);
10 – citeşte/încarcă octetul cel mai puțin semnificativ (la incărcare anulează octetul cel mai semnificativ);
11 – citeşte/încarcă cel mai semnificativ octet, apoi cel mai puțin semnificativ octet;

biții 7,6

00 – selecție contor 0;
01 – selecție contor 1;
10 – selecție contor 2;
11 – comandă ilegală;

Figura 7.12. Cuvântul de comandă pentru circuitul 8253.

Conținutul cuvântului de comandă este prezentat în figura 7.12. Se constată că: biții 7, 6 sunt folosiți pentru selectarea contorului la care se referă comanda respectivă, biții 5, 4 specifică modalitățile de manipulare ale conținuturilor octeților inferior şi superior, care formează contorul selectat, bitii 3, 2, 1 indică modul de operare, în timp ce bitul 0 caracterizează funcționarea binara (valoare maximă: 65535_{10}) sau zecimală codificată binar (valoare maxima: 9999_{BCD}), ale contorului dat.

Contorul selectat se consideră încărcat atunci când în el s-au înscris unul sau doi octeți, în funcție de specificațiile biților 5, 4 din cuvântul de comandă, operația fiind urmată de un front pozitiv şi unul negativ ale impulsului de ceas. O eventuala citire înainte de frontul căzător va conduce la o valoare incorectă.

În continuare vor fi prezentate modurile de funcționare ale circuitului 8253.

Modul 0 este definit ca întrerupere, la terminarea numărării. După încărcarea cuvântului de comandă, corespunzator acestui mod, ieşirea contorului selectat va fi forțată la nivelul coborât (fig. 7.13). În continuare se încarcă contorul cu o anumită valore numerică, care va fi decrementată, prin aplicarea semnalului de ceas, pe intrarea corespunzatoare. Ieşirea se va menține pe nivel coborât pâna în momentul când valoarea numărului din contor devine zero, moment în care ieşirea trece pe nivel ridicat, menținându-se astfel, pâna la o noua încărcare a contorului. Decrementarea continuă şi după atingerea valorii finale. Reânscrierea contorului, pe durata decrementării, are ca efect blocarea operației curente, dacă se încarcă primul octet, sau amorsarea unei noi operații, dacă se încarcă al doilea octet.

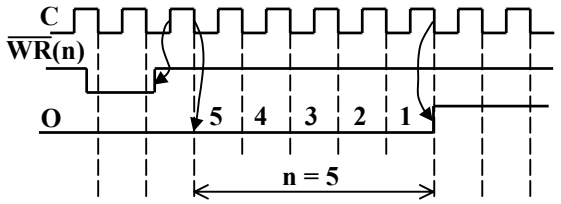


Figura 7.13. Modul zero

În figura 7.13 contorul a fost încărcat cu valoarea 5, ca urmare a execuției unei operații de înscriere ($\overline{WR}$ = 0). Decrementarea se va declanşa după un front pozitiv, urmat de unul negativ ale semnalului de ceas. Decrementarea este blocată pe durata aplicării, la intrarea poartă, a unui semnal de nivel coborit.

Citirea conținutului contorului, pe durata decrermentării, necesită o tehnică specială, care va fi prezentată la sfârşitul acestui paragraf.

Cuvintul de comanda are urmatoarea structura:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	0	0	1	0	0	0	0

Modul 1 este definit ca monostabil programabil. Ieşirea contorului va trece de la un nivel ridicat, la unul coborât mumai după ce cuvântul de comandă şi contorul au fost încărcate şi semnalul aplicat la poartă are o tranziție pozitivă. Această tranziție declnşează decrementarea care, la atingerea valorii finale (zero), va aduce la nivel ridicat ieşirea contorului selectat (fig. 7.14).

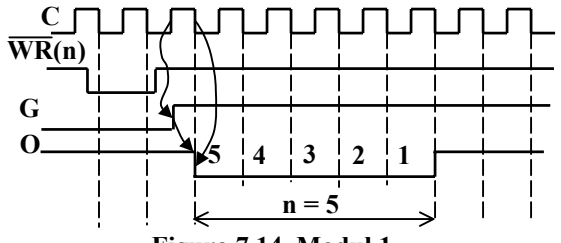


Figura 7.14. Modul 1

Dacă în timp ce ieşirea este la nivel coborât, o nouă valoare este înscrisă în contor, aceasta nu va afecta durata monoimpulsului, până la următoarea declanşare. Valoarea curentă a contorului poate fi citită în orice moment, fără a afecta monoimpulsul. Dacă intrarea la poarta G are o tranziție pozitivă, indiferent de faptul că decrementarea s-a terminat sau nu, contorul este relansat cu valoarea încărcată inițial. În cazul în care, pe parcursul decrementării, se încarca o nouă

valoare în contor, aceasta va fi luată în considerație la prima tranziție pozitiva a semnalului la poarta G.
Cuvâtul de comandă:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	1	1	0	0	0	1	1

Modul 2 este definit ca generator de impulsuri divizate cu N. Ieşirea va fi forțată la nivel coborât, după încărcarea cuvântului de comandă şi a numărătorului selectat cu constanta N, pe o perioadă egală cu cea a semnalului de ceas. Perioada impulsurilor astfel generate va fi egală cu N (fig. 7.15).

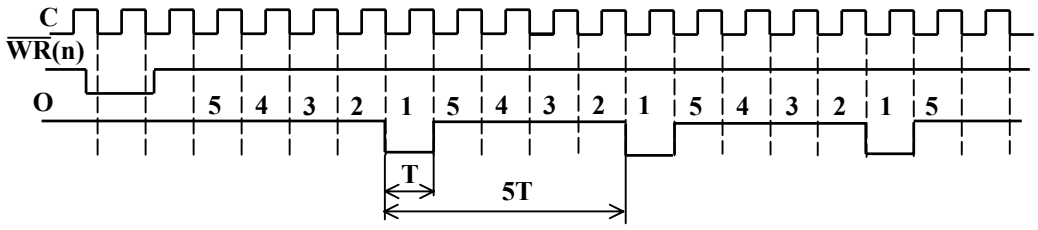


Figura 7.15. Modul 2

Dacă pe parcursul operării, intrarea G (poarta) va fi forțată la nivel coborât, ieşirea 0 va fi adusă la nivel ridicat, iar la revenirea lui G la un nivel ridicat, contorul se va decrementa din nou, de la valoarea initială. Astfel, intrarea G poate fi folosită pentru sincronizarea contorului.

Cuvântul de comandă:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	0	1	1	0	1	0	0

Modul 3 este definit ca generator de impulsuri dreptunghiulare. El este similar cu modul 2, cu excepția faptului că iesirea va ramâne la un nivel ridicat pâna la realizarea decrementarii, conform cu N/2 sau (N-1)/2 intervale de ceas şi la nivel coborât, pentru celelalte N/2 sau (N-1)/2 intervale de ceas, după cum N este par sau impar. Aceasta se realizează (fig. 7.16) prin decrementarea cu 2 pe frontul căzător al fiecarui semnal de ceas. Când contorul ajunge în zero, ieşirea se modifică, contorul este încărcat cu valoarea inițială şi procesul se continuă, având un caracter repetitiv.

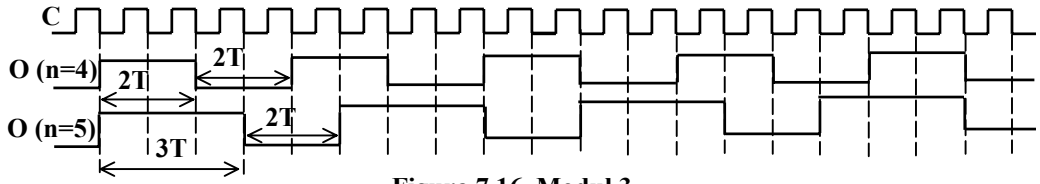


Figura 7.16. Modul 3

Modul 4, definit ca strob comandat prin software, asigură o ieşire la nivel ridicat, după stabilirea modului. Decrementarea contorului selectat are loc după încărcarea lui cu numărul dat. Ieşirea va fi forțată la nivel coborât pe durata unei perioade a semnalului de ceas, când contorul a fost decrementat la zero. Decrementarea va fi inhibată, dacă intrarea G va fi forțată la nivel coborât. Reîncărcarea contorului va reporni decrementarea, plecând de la noua valoare plasată în contor (fig. 7.17).

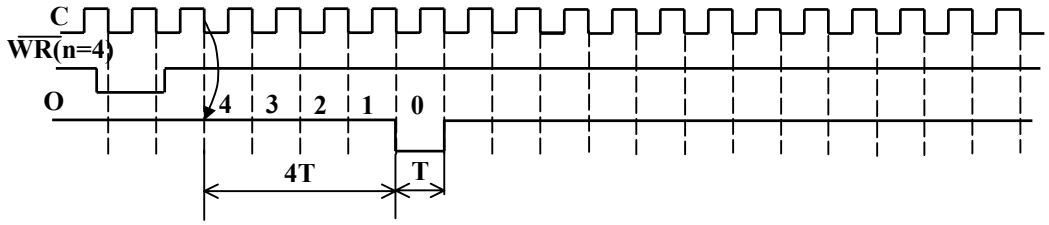


Figura 7.17. Modul 4.

Cuvântul de comandă:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	0	0	1	1	0	1	0

Modul 5, definit ca strob comandat prin hardware, asigură decrementarea contorului selectat, după încărcarea lui, începând cu frontul crescător al semnalului aplicat la poarta G. Ieşirea va fi forțată la nivel coborât, pe durata unei perioade de ceas, în momentul în care conținutul contorului a devenit zero (fig. 7.18).

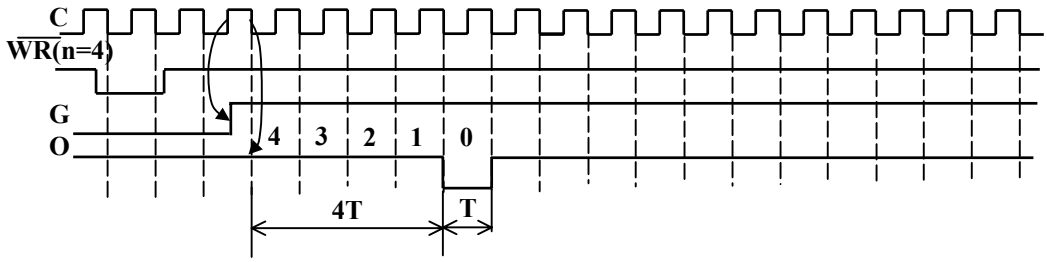


Figura 7.18. Modul 5.

Operația de citire a conținutului unui contor este importantă pentru a cunoaşte valoarea acestuia la un moment dat. Citirea se poate face în două moduri diferite.

În primul mod, citirea se face în timpul decrementarii. Pentru aceasta, pe baza unei comenzi, conținutul contorului, care urmează să fie citit, se forțează într-un regisru suplimentar, din circuitul 8253, contorul continuând operaița de decrementare. Cuvâtul de comandă necesar are urmatoarea structură:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
SC1	SC0	0	0	x	x	x	x

SC1, SC0 - biții 7 şi 6 din cuvântul de comandă specifică contorul al cărui conținut va fi salvat în vederea citirii (00, 01, 10). Biții 5, 6 sunt egali cu 0 şi specifică operația de memorare a conținutului contorului. Ceilalți biți sunt indiferenți.

În al doilea mod se blochează în prealabil ceasul, printr-o logică externă sau controlând semnalul de comandă la poarta G. După aceasta se citeşte conținutul contorului şi se transferă în memorie.

7.6. Circuitele timer ale microcontrolerelor

În cazul microcontrolerelor, circuitele contor/periodizator sunt larg răspândite şi reprezintă unul din circuitele de bază ale acestora, îndeplinind funcții complexe.

Se va prezenta în continuare modulul circuitelor de contorizare/periodizare ale microcontrolerului TI 320F240 (GPTimer) datorită numeroaselor funcții îndeplinite de acesta, reprezentând astfel un modul complet şi complex.

7.6.1. Timerele de uz general GPTimer

În modulul EV sunt trei timere de uz general (GP Timers). Aceste timere pot fi folosite ca baze de timp independente în aplicații cum ar fi:

- generarea perioadei de eşantionare în sistemele de control;
- furnizarea unei baze de timp pentru operarea circuitului QEP sau unitățile de captură;
- furnizarea unei baze de timp pentru operarea unitățile de comparare simple sau complete (full) şi a circuitelor PWM asociate pentru a genera ieşiri comparare/PWM.

Blocurile funcționale ale timerelor GP

Figura 7.19. prezintă schema bloc a timeului GP. Fiecare timer conține:

- un contor up/down scriere/citire (R/W) de 16 biți, TxCNT (x=1,2,3);
- un registru R/W de 16 biți pentru comparare (cu imagine în memorie) TxCMPR (x=1,2,3);
- un registru R/W de 16 biți pentru perioadă (cu imagine în memorie) TxPR (x=1,2,3);
- un registru de control R/W de 16 biți, TxCON (x=1,2,3);
- prescalare (divizare) programabilă a ceasului de intrare intern sau extern;
- logică de control a întreruperilor;
- un pin de ieşire al comparatorului GP timer, TxPWM/TxCMP (x=1,2,3);
- ieşire logică.

Un alt registru de control, GPTCON specifică acțiunile care se vor iniția pentru diferite evenimente ale timerelor şi indică direcția de numărare pentru toate cele trei timere. GPTCON poate fi citit sau scris, scrierea celor rei biți de stare neavând nici un efect.

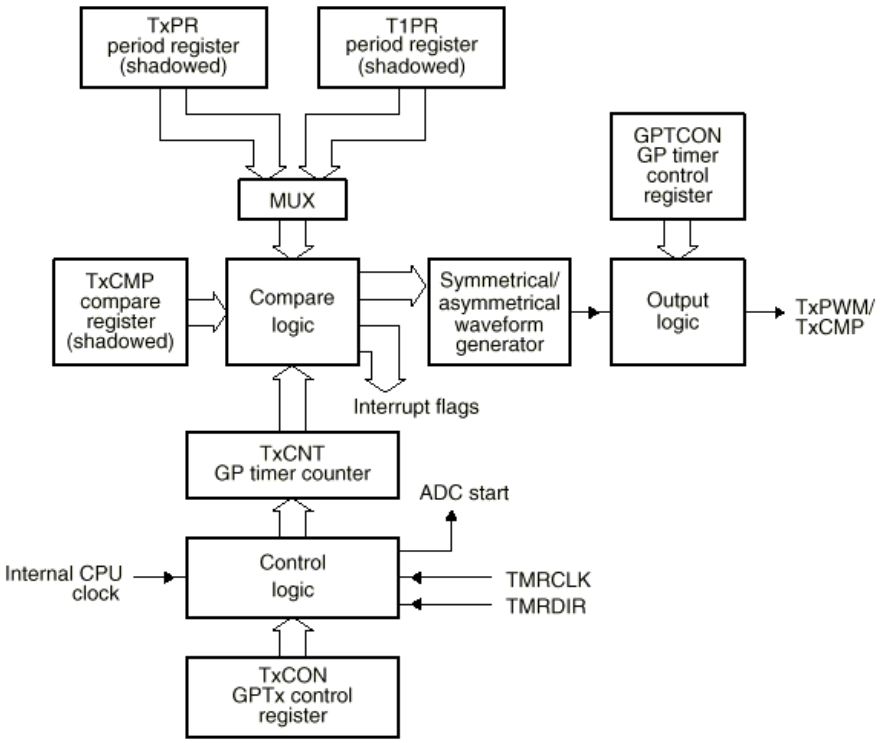


Fig. 8.4. Schema bloc a timerului GP
x = 1, 2 sau 3

Intrările timerelor GP

Intrările timerelor GP sunt:

- ceasul intern CPU care vine direct din miez şi deci are aceeaşi frecvență ca şi ceasul CPU;
- ceas extern, TMRCLK care are frecvența maximă un sfert din frecvența ceasului CPU;
- intrare de direcție TMRDIR pentru utilizarea timerelor GP în modul direcțional up/down;
- semnal de reset RESET.

În plus timerul GP 3 poate folosi depăşirea timerului GP 2 ca intrare de ceas când timerele GP 2 şi 3 sunt cascadate într-un timer de 32 de biți. Când un timer GP este utilizat cu circuitul QEP, circuitul QEP generează atât ceasul pentru contor cât şi sensul de numărare.

Ieşirile timerelor GP

Timerele GP au următoarele ieşiri:

- ieşirea comparare/PWM a timerului GP TxPWM/TxCMP, (x=1,2,3);
- semnal de start pentru modulul ADC;
- semnale de depăşire inferioară, superioară, egalitate la comparare şi egalitate la perioadă pentru logica proprie de comparare sau pentru unitățile de comparare simplă sau completă;
- bit de indicare a sensului de numărare.

Controlul operării timerului GP

Modul de operare al timerului GP este controlat de registrul de control TxCON. Biții din registrul TxCON determină:

- unul din cele şase moduri de numărare este folosit pentru timerul GP;
- sursa de ceas internă sau externă pentru timerul GP;
- factorul de divizare a ceasului de intrare (în domeniul 1 la 1/128);
- condiția de reîncărcare a registrului comparatorului timerului;
- activarea sau dezactivarea timerului;
- registrul de perioadă este registrul propriu sau registrul timerului GP 1 (numai pentru T2CON şi T3CON);
- utilizarea semnalului de depăşire a timerului GP 2 ca sursă de ceas pentru timer GP 3 (numai la T3CON);

Registrul de control al timerului GP (GPTCON)

Registrul de control GPTCON specifică acțiunile care se vor face la diferitele evenimente furnizate de timerele GP.

Registrele de comparare ale timerelor GP

Registrele de comparare asociate cu timerele GP stochează valoarea care va fi în mod continuu comparată cu contorul timerului GP. Când se întâlneşte egalitatea se produc mai multe evenimente. Aceste evenimente includ tranziția ieşirii comparare/PWM asociate şi start ADC în concordanță cu starea bitului din GPTCON. În plus fanionul corespunzător întreruperii de comparare este setat. Această operație poate fi activată sau dezactivată de către bitul 1 din TxCON.

Registrul de perioadă a timerului GP

Valoarea registrului perioadă a timerului GP determină perioada timerului. Operarea timerului se opreşte reținând valoarea curentă, resetează la zero, sau porneşte numărătoarea invers atunci când se întâlneşte o egalitate între registrul perioadei şi conținutul timerului modul de reacție depinzând de modul de numărare setat.

Dubla bufferare ale registrelor perioadei şi comparatorului ale timerelor GP

Regiştrii de comparare şi ai perioadei TxCMPR şi TxPR ai timerului GP au corespondent în memorie. O valoare nouă poate fi scrisă în oricare dintre aceşti regiştrii în orice moment. Valoarea este scrisă în corespondentul din memorie. Pentru registrul de comparare conținutul registrului din memorie este încărcat în registrul de lucru (activ) numai când un anumit eveniment specificat de TxCON se produce. Pentru registrul de perioadă, registrul de lucru este reîncărcat cu valoarea registrului din memorie numai când valoarea în registrul contorului TxCNT este zero. Condiția la care registrul de comparare este reîncărcat poate fi una din următoarele:

- imediat după ce registrul din memorie a fost scris;
- la o depăşire inferioară, aceasta însemnând atunci când valoarea contorului timerului GP este zero;
- la depăşire superioară sau la egalitatea perioadei, atunci când valoarea contorului este zero sau când are o valoare egală cu cea a registrului perioadei.

Facilitatea de dublă bufferare permite codului de aplicație să actualizeze registrul de comparare sau al perioadei în orice moment. Acesta permite schimbări ale perioadei timerului şi lățimii pulsului PWM pentru ciclul următor. Schimbarea "din zbor" a valorii perioadei timerului în cazul generării PWM înseamnă schimbarea "din zbor" a frecvenței purtătoarei PWM.

Inițializarea registrului perioadei

Registrul perioadei timerului GP trebuie inițializat înainte de inițializarea contorul corespunzător la o valoare diferită de zero. În caz contrar valoarea registrului perioadei va rămâne neschimbată până la următoarea depăşire inferioară.

Notă: Registrul de comparare este transparent (noua valoare încărcată se duce direct în registrul activ) atunci când operația de comparare asociată este dezactivată. Acest lucru se aplică tuturor registrelor de comparare din EV.

Ieşirea comparare/PWM a timerului GP

Ieşirea GP Timer comparare/PWM poate fi specificată a fi activă high, activă low forțată high sau forțată low în funcție de cum sunt configurați biții GPTCON. Ieşirea va comuta din low în high (high în low) la prima egalitate la comparație atunci când este activă high (low). Ea va comuta din high în low (low în high) la a doua egalitate la comparare dacă GP timer este în modul de numărare sus/jos sau la egalitatea cu perioada dacă GP timer este în modul numărare sus. Ieşirea de comparare a timerului va deveni high (low) în acelaşi mod atunci când este setată să fie forțată în starea high (low).

Sensul de numărare a timerului GP

Sensul de numărare pentru toate cele trei timere GP este reflectat în biţii respectivi din GPTCON pe timpul tuturor operaţiilor timerelor cu:

- 1 reprezentând numărare sus;
- 0 reprezentând numărare jos.

Pinul de intrare TMRDIR determină sensul de numărare când timerul GP este în modul de numărare direcţional up/down. Când TMRDIR este setat high este specificată numărarea înainte iar când este pus în starea low este specificat modul de numărare înapoi.

Ceasul timerelor GP

Sursa de ceas a timerelor GP poate fi ceasul intern al CPU sau ceasul aplicat extern la pinul TMRCLK. Frecvenţa ceasului extern trebuie să fie mai mică sau egală cu un sfert din ceasul CPU. Timerele GP 2,3 sau 2 şi 3 împreună formând un numărător pe 32 de biţi pot fi utilizate cu circuitul QEP în modul de numărare direcţional up/down. În acest caz circuitul QEP furnizează atât semnalul de ceas cât şi cel de sens.

Un domeniu larg de factori de prescalare este prevăzut pentru fiecare timer GP.

Timer de 32 biţi

Semnalul de depăşire al timerului GP 2 este folosit ca intrare de ceas pentru timerul GP 3 atunci când timerele 2 şi 3 sunt cascadate într-un timer de 32 de biţi. Când acest lucru se întâmplă, timerul GP 2 furnizează cei mai puţini semnificativi 16 biţi ai contorului de 32 de biţi. Contorul de 32 de biţi astfel obţinut poate opera numai în modul direcţional up/down de numărare cu ceas intern sau extern şi cu intrare de sens externă. Circuitul QEP poate fi de asemenea ales pentru a furniza impulsuri de ceas şi de sens pentru numărătorul de 32 de biţi. Registrul perioadei pentru timerele GP 2 şi 3 sunt cascadate în acest caz pentru a furniza un registru de perioadă de 32 de biţi pe când operaţia de comparare este bazată pe compararea registrelor individuale şi se generează semnale de egalitate individuale pentru fiecare potrivire pe 16 biţi. Evenimentele de depăşire inferioară şi superioară sunt generate pentru numărător de 32 biţi. Fanioanele de depăşire inferioară şi superioară ale timerului GP 2 semnalează în acest caz depăşirile registrului de 32 de biţi, cele ale timerului GP 3 neavând nici o relavanţă în acest caz. Fanioanele de comparare ale timerelor GP 2 şi 3 sunt setate individual la egalitatea comparării.

Ceasul de intrare la operare QEP

Circuitul pulsurilor codate în cuadratură (QEP), atunci când este selectat, poate genera semnal de ceas şi de direcţie pentru timerele GP 2 sau 3 sau 2 şi 3 împreună pentru a forma un numărător de 32 de biţi, în modul de numărare direcţional up/down. Această intrare de ceas nu poate fi scalată de circuitele de prescalare ale timerelor GP

(factorul de prescalare va fi totdeauna egal cu unu). De asemenea frecvența de ceas generată de circuitul QEP va fi de patru ori mai mare decât frecvența oricărui canal de intrare QEP din cauză că timerul selectat numără fronturile crescătoare şi căzătoare de pe cele două intrări ale QEP. Frecvența de intrare la intrările QEP trebuie să fie mai mică sau egală cu un sfert din frecvența ceasului CPU.

Sincronizarea timerelor GP

Timerele GP 2 şi 3 pot fi sincronizate individual cu timerul GP 1 prin configurarea adecvată a registrelor T2CON şi T3CON în modurile următoare:

- pornirea operării timerului GP 2 sau 3 utilizând acelaşi bit de control în T1CON care porneşte şi operarea timerului GP 1;
- inițializarea contoarelor timerelor GP 2 sau 3 cu valori diferite înainte de pornirea operației sincronizate;
- specificarea unuia din timerele GP 2 sau 3 care să utilizeze registrul perioadei timerului GP 1 ca registru propriu de perioadă (ignorând propriul registru de perioadă).

Aceste metode permit sincronizarea dorită între evenimentele timerelor GP. Dacă fiecare timer porneşte operația de numărare de la valoarea lui curentă (memorată în registrul contorului) un timer GP poate fi programat să pornească cu o întârziere cunoscută după un alt timer GP. Trebuie notat că sunt necesare două scrieri în T1CON pentru a sincroniza timerul GP1 cu timerul GP2 sau timerele GP 2 şi 3.

Pornirea ADC la eveniment timer GP

Biții din GPTCON pot specifica evenimentul GP Timer pe care porneşte conversia analog-digitală (ADC), eveniment care poate fi: depăşirea inferioară, egalitatea la comparare, sau egalitatea la perioadă. Această facilitate permite sincronizarea evenimentele GP timer şi pornirea ADC fără nici o intervenție din partea CPU.

Întreruperile timerelor GP

Sunt 12 fanioane de întrerupere în EVIFRA şi EVIFRB pentru cele trei timere GP. Fiecare timer poate genera patru întreruperi la următoarele evenimente:

- depăşire superioară TxOFINT (x=1,2 sau 3);
- depăşire inferioară TxUFINT (x=1,2 sau 3);
- egalitate la comparare TxCINT (x=1,2 sau 3);
- egalitate la perioadă TxPINT (x=1,2 sau 3).

Evenimentul de egalitate la comparare se produce atunci când conținutul contorului timerului GP este egal cu cel al registrului de comparare. Fanionul

corespunzător intreruperii de comparare este setat două perioade de ceas CPU după ce se produce egalitatea dacă operația de comparare a fost activată.

Un eveniment de depăşire superioră se produce atunci când numărătorul timerului este egal cu FFFFh. O depăşire inferioară se produce atunci când contorul are valoarea 0000h. În mod similar un eveniment la perioadă se produce când valoarea din contorul timerului este egală cu conținutul registrului de perioadă. Fanioanele de depăşire superioară, inferioară sau la perioadă sunt setate doi ciclii de perioadă de ceas CPU după ce s-a întâlnit evenimentul respectiv. Este de notat că definirea depăşirii superioare şi inferioare este diferită de definiția convențională.

Operația de numărare a timerului GP

Fiecare timer GP are şase moduri selectabile de operare:

- stop cu reținere;
- numărare sus o singură dată;
- numărare sus continuă;
- numărare direcțională sus/jos;
- numărare o singură dată sus/jos;
- numărare continuă sus/jos.

Configurația biților corespunzători în registrul de control TxCON determină modul de numărare a timerului GP. Bitul de validare a timerului TxCON[6] validează sau invalidează operația de numărare a timerului. Când timerul este dezactivat operația de numărare se opreşte şi prescalatorul timerului este resetat la x/1. Când timerul este activat acesta porneşte operația de numărare în concordanță cu modul specificat de biții corespunzători din TxCON.

Modul stop/reținere

În acest mod operarea timerului se opreşte şi reține valoarea curentă. Contorul timerului, ieşirea de comparare, contorul de prescalare rămân neschimbate.

Modul de numărare sus o singură dată

Timerul GP numără în acest mod ceasul divizat de contorul de prescalare până când valoarea din contorul timerului este egală cu cea din registrul perioadei. Pe următorul front crescător a ceasului de intrare după această potrivire timerul GP resetează la zero şi îşi dezactivează operația de numărare prin resetarea bitului de activare TxCON[6].

Fanionul întreruperii timerului este setat două perioade de ceas CPU după ce s-a produs egalitatea între registrul perioadei şi contorul timerului. O comandă de start este trimisă către modulul ADC în acelaşi timp cât fanionul întreruperii este setat dacă întreruperea de perioadă a fost selectată cu ajutorul biților corespunzători din GPTCON pentru a porni ADC.

Două perioade de ceas după ce timerul GP a devenit zero este setat fanionul de depăşire inferioară. Un semnal de pornire este trimis către modulul ADC în acelaşi timp cât fanionul întreruperii este setat dacă întreruperea de perioadă a fost selectată cu ajutorul biţilor corespunzători din GPTCON pentru a porni ADC.

Fanionul de întrerupere la depăşire superioară este setat două perioade de ceas CPU după ce valoarea TxCNT atnge valoarea FFFFh.

Durata de numărare a perioadei este (TxPER)+1 ciclii de ceas prescalat de la intrarea timerului dacă contorul timerului are valoarea zero la începutul perioadei.

Valoarea iniţială a timerului GP poate fi orice valoare între 0h şi FFFFh. Când valoarea iniţială este mai mare decât valoarea registrului perioadei timerul numără sus până la FFFFh, resetează la zero si numără sus în continuare până când atinge valoarea perioadei. Când valoarea iniţială a registrului contorului timerului este egală cu valoarea înscrisă în registrul perioadei, timerul setează fanionul întreruperii la perioadă, resetează la zero setează fanionul de întrerupere la depăşire inferioară şi imediat după aceasta termină de numărat până la perioadă. Dacă valoarea iniţială a contorului este între zero şi valoarea perioadei, timerul numără sus până la terminarea perioadei într-un mod similar cu cel în care numărătorul timerului este egal cu cel al registrului perioadei.

O dată ce perioada s-a terminat timerul GP poate fi pornit din nou numai de către software prin scrierea bitului de validare a timerului TxCON[6].

Bitul care indică sensul de numărare din GPTCON este 1 în acest caz. Se poate utiliza atât ceas intern cât şi cel extern. Pinul de intrare TMRDIR este ignorat în acest mod de operare.

În figura 7.20. este prezentat modul de numărare sus o singură dată considerând factorul de prescalare este 1.

De notat că timerul GP începe numărătoarea imediat ce TxCON[6] este setat. Acest lucru este valoabil pentru toate modurile de numărare.

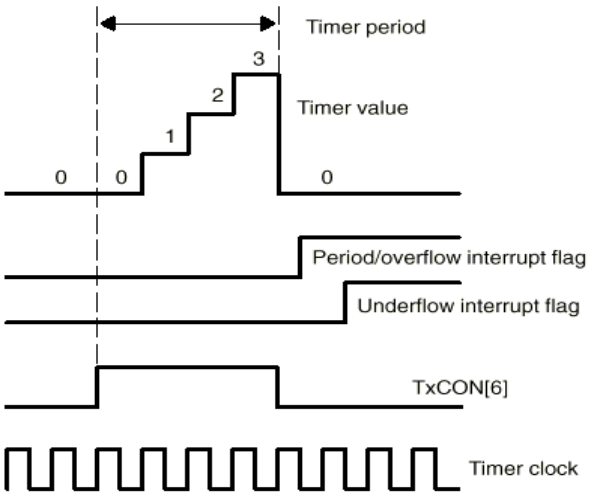


Fig. 7.20. Modul numărare sus o singură dată

Modul de numărare sus continuu

Modul de numărare sus continuu este similar cu numărare sus o singură dată repetat de fiecare dată când timerul este resetat la zero. Numărarea se face în sens direct după ceasul primit la intrare, prescalat corespunzător, până când registrul contorului este egal cu cel al perioadei. În acest moment contorul resetează la zero şi porneşte o nouă perioadă. Durata perioadei este TxPR+1 ciclii de ceas scalat cu excepţia primei perioade. Durata primei perioade este aceeaşi numai în cazul în care contorul porneşte de la zero.

Valoarea iniţială a contorului timerului GP poate fi oricare în intervalul 0h şi FFFFh. Când valoarea iniţială este mai mare decât valoarea registrului de perioadă timerul numără până la FFFFh, resetează la zero şi continuă operarea cu valoarea iniţială zero. Când valoarea iniţială este egală cu valoarea registrului de perioadă, timerul setează fanionul de întrerupere de perioadă, resetează la zero, setează fanionul întreruperii de depăşire inferioară şi continuă operarea cu valoarea iniţială zero. Când valoarea iniţială este cuprinsă între zero şi valoarea conţinutului registrului de perioadă, timerul numără sus până la valoarea registrului de perioadă şi continuă operarea la fel ca în cazul când valoarea iniţială este egală cu valoarea registrului perioadei.

Întreruperile asociate evenimentelor (depăşire superioară, inferioară, perioadă) sunt generate în situaţiile respective la fel ca la numărarea o singură dată.

Bitul de sens din GPTCON este 1 pentru acest mod. Poate fi selectată şi sursa de ceas extern. Intrarea TMRDIR este ignorată în acest mod de operare.

Numărarea continuă sus făcută de timerul GP este în mod particular utilă pentru generarea PWM cu fronturi triggerate sau pentru perioade de eşantionare în diferite sisteme de control ale motoarelor şi mişcării.

Figura 7.21 prezintă modul de numărare sus continuu cu factor de prescalare 1.

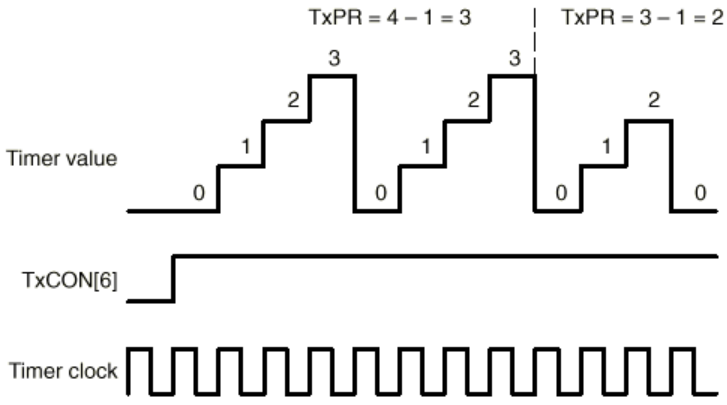


Fig. 7.21. Modul de numărare sus continuu

Din figura 7.21 se vede că că nu se pierde nici un impuls de ceas în momentul când contorul atinge valoarea registrului perioadei şi porneşte un nou ciclu de numărare.

Modul de numărare direcţional up/down al timerelor GP 1 şi 3

Timerele în modul de numărare direcţional up/down numără sus (up) sau jos (down) în concordanţă cu ceasul scalat şi intrarea TMRDIR. Când pinul TMRDIR este în starea high contorul numără până atinge perioada sau valoarea FFFFh. Când numărătorul atinge una din aceste valori se opreşte. Numărătorul numără jos până la valoarea zero când intrarea TMRDIR este în starea low. La această valoare (zero) numărătorul se opreşte.

Valoarea iniţială a numărătorului poate fi oricare între zero şi FFFFh. Când valoarea iniţială este mai mare decât perioada timerul numără sus până atinge FFFFh şi se opreşte aici dacă TMRDIR este high. Dacă TMRDIR este low atunci numără jos până la valoarea perioadei după care continuă ca în situaţia când valoarea iniţială este egală cu valoare perioadei. Dacă valoarea iniţială este egală cu valoarea registrului perioadei atunci contorul se opreşte la acestă valoare dacă TMRDIR este high sau numără jos de la acestă valoare dacă TMRDIR este low.

Fanioanele de întrerupere pe evenimentele respective sunt generate similar ca la modul de numărare o singură dată sus.

Intervalul de timp între modificarea TMRDIR şi modificarea sensului de numărare este de 2 ciclii de ceas CPU după sfârşitul numărătorii în curs – comutării curente - (altfel spus la sfârşitul perioadei impulsului de numărare prescalat).

Figura 7.22 ilustrează modul de numărare direcţional up/down.

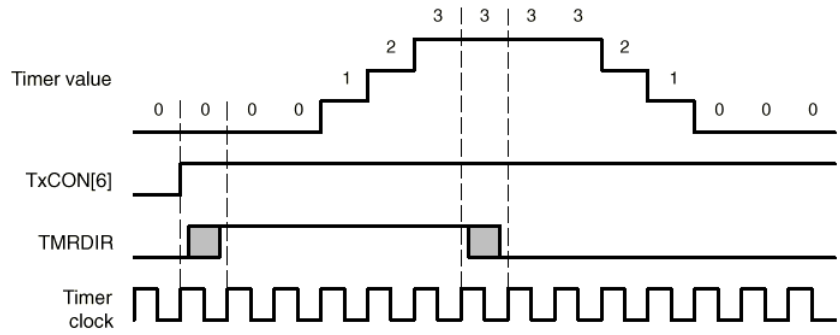


Fig. 7.22. Modul de numărare direcţional sus/jos

Modul de numărare direcţional up/down al timerului GP 2

Modul de numărare direcţional up/down la timerului GP 2 diferă de cele ale timerelor 1 şi 3. Timerul GP 2 numără până la perioadă şi furnizează depăşire superioară şi inferioară.

Acest mod de operare poate fi utilizat pentru a măsura durate sau numărarea apariţiei evenimentelor externe în controlul motorelor/mişcării (deplasării) şi aplicaţii ale electronicii de putere. Nu apar tranziţii pe ieşirile de comparare asociate cu modulul de comparare (care include comparare timer GP, comparare completă sau simplă) care utilizează timerul GP ca bază de timp.

Când QEP este selectat ca sursă de ceas pentru timer GP acesta trebuie pus în modul de unmărare direcțional up/down.

Modul de numărare o singură dată up/down

Timerul GP în acest mod numără în concordanță cu ceasul prescalat până la valoarea din registrul perioadei. Atunci el schimbă direcția de numărare şi numără jos până la valoarea zero. Când s-a atins valoarea zero este resetat TxCON[6] şi contorul de prescalare se opreşte şi reține starea sa curentă.

Perioada timerului GP este 2x(TxPR) ciclii de ceas scalat dacă valoarea sa inițială este 0.

Valoarea inițială a numărătorului poate fi oricare între zero şi FFFFh. Când valoarea inițială este mai mare decât perioada timerul numără sus până atinge FFFFh resetează la zero şi continuă numărătoarea ca în cazul când valoarea inițială este zero. Dacă valoarea inițială este egală cu valoarea registrului perioadei atunci numărătorul numără jos până la zero şi termină perioada aici. Dacă valoarea inițială este între zero şi valoarea perioadei atunci contorul numără sus până la atingerea perioadei şi apoi continuă ca în situația când contorul este egal cu perioada.

Fanioanele de întrerupere pe evenimentele respective sunt generate similar ca la modul de numărare o singură dată sus. Este de notat că evenimentul egalitatea perioadei se produce la jumătatea perioadei de numărare la egalitatea contorului cu registrul de perioadă.

O dată ce s-a stabilit acest mod de numărare operația de numărare nu poate fi pornită decât scriind 1 în TxCON[6]. Bitul de sens este 1 la numărare sus şi 0 la numărare jos. Poate fi selectat şi ceasul intern şi cel extern. Intrarea TMRDIR este ignorată în acest mod.

Figura 7.23 ilustrază modul de numărare o singură dată sus/jos.

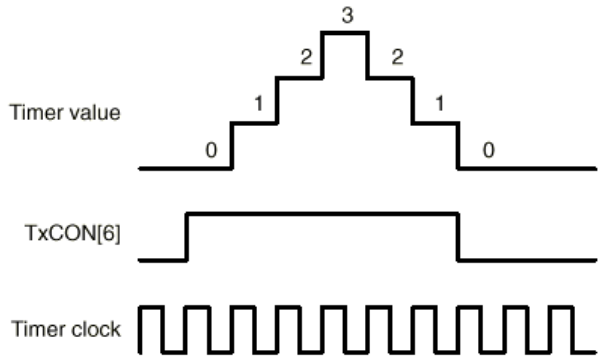


Fig. 7.23. Modul de numărare o singură dată sus/jos *(TxPR=3)*

Modul de numărare up/down continuu

În acest mod operarea este similară modului de numărare o singură dată sus/jos cu precizarea că timerul reia operațiunea de fiecare dată când este resetat la zero. O dată

ce acest mod de operare este pornit nu este necesară nici o intervenție software sau hardvare pentru contorizarea repetată a perioadei.

Perioada timerului GP este 2x(TxPR) ciclii de ceas scalat cu excepția primei perioade. Durata primei perioade de numărare are aceeaşi valoare dacă valoarea inițială a contorului timerului este 0.

Valoarea inițială a numărătorului poate fi oricare între zero şi FFFFh. Când valoarea inițială este mai mare decât perioada timerul numără sus până atinge FFFFh resetează la zero şi continuă numărătoarea ca în cazul când valoarea inițială este zero. Dacă valoarea inițială este egală cu valoarea registrului perioadei atunci numărătorul numără jos până la zero şi continuă la fel ca în cazul în care valorea inițială este zero. Dacă valoarea inițială este între zero şi valoarea perioadei atunci contorul numără sus până la atingerea perioadei şi apoi continuă ca în situația când contorul este egal cu perioada.

Fanioanele de întrerupere pe evenimentele respective sunt generate similar ca la modul de numărare o singură dată sus.

Sensul de numărare este indicat de bitul corespunzător din GPTCON care este 1 atunci când timerul numără sus şi 0 când numără jos. Poate fi selectat atât ceasul intern cât şi cel extern. Intrarea TMRDIR este ignorată în acest mod.

Figura 7.24 prezintă modul de numărare continuu sus/jos.

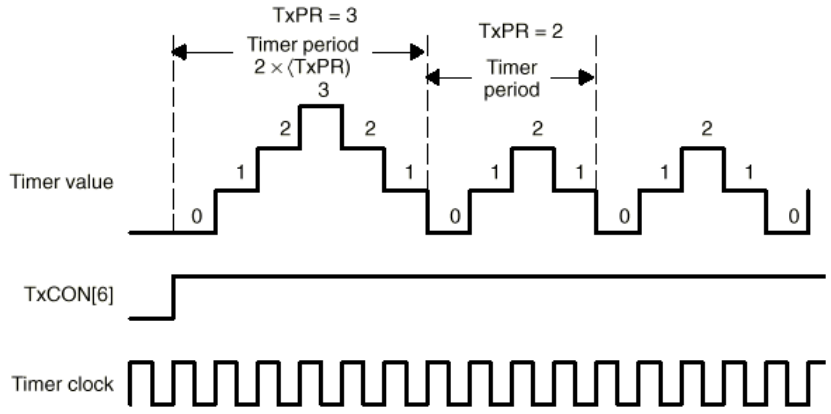


Fig. 7.24. Modul de numărare continuu sus/jos

Operația de comparare a timerului GP

Fiecare timer GP are asociat un registru de comparare TxCMPR şi un pin de ieşire de comparare/PWM. Valoarea timerului GP este în mod constant comparată cu registrulde comparare asociat. Evenimentul la comparare se produce atunci când registrul de comparare asociat timerului este egal cu conținutul timerului. Operația de comparare este validată atunci când bitul TxCON[1] este setat la 1. Dacă acest bit este setat la 1 atunci la apariția unui eveniment de comparare se produc următoarele operații:

- fanionul de întrerupere la comparare este setat pe două perioade de ceas CPU după producerea egalității între registrul timer şi registrul comparator;
- dacă timerul nu este într-un mod de directional de numărare sus/jos se produce o tranziție pe ieşirea asociată comparare/PWM în concordanță cu bitul de configurare din GPTCON, un ciclu de ceas după producerea egalității;
- dacă fanionul de întrerupere la comparare a fost selectat să pornească ADC atunci se trimite un semnal de start la acesta în acelaşi moment în care fanionul este setat.

Dacă operația de comparare a timerului GP este dezactivată atunci ieşirea de comparare/PWM este în înaltă impedanță şi nici unul din evenimentele descrise anterior nu se produce.

Tranzițiile comparare/PWM

Tranzițiile ieşirilor de comparare/PWM sunt controlate de un generator de forme de undă simetrice şi asimetrice asociat logicii de ieşire şi depinde de următoarele:

- definirea biților în GPTCON;
- modul de numărare în care se găseşte timerul;
- sensul de numărare când modul de numărare este un singur ciclu sus jos sau numărare continuă sus/jos.

Generatorul de forme de undă asimetrice/simetrice

Generatorul de forme de undă asimetrice/simetrice generează o formă de undă asimetrică sau simetrică pe ieşirea comparare/PWM în funcție de modul de numărare în care se găseşte timerul GP.

Generarea formei de undă asimetrice

O formă de undă asimetrică cum este cea prezentată în figura 7.25 este generată când timerul este în modul de numărare un ciclu sau continuu sus. Când timerul este într-unul din aceste două moduri, ieşirea generatorului de forme de undă se schimbă în felul următor:

- zero înainte ca operația de numărare să pornească;
- rămâne neschimbată până la producerea unei egalități de comparare;
- comută la apariția egalității de comparare;
- rămâne neschimbată până la sfârşitul perioadei;
- resetează la zero la sfârşitul perioadei dacă o nouă valoare de comparare pentru perioada următoare este diferită de zero.

Ieşirea este 1 pe întreaga perioadă dacă valoarea de comparare este zero la începutul perioadei. Ieşirea nu resetează la zero dacă o nouă valoare de comparare pentru perioada următoare este zero. Acest lucru este important pentru că permite generarea punsurilor PWM cu factor de umplere între 0% şi 100% fără apariţia impulsurilor parazite. Ieşirea este zero pe întreaga perioadă dacă valoarea de comparare este mai mare decât valoarea din registrul perioadei. Ieşirea este 1 pentru o perioadă a ceasului de intrare scalat dacă valoarea de comparare este egală cu valoarea perioadei.

O caracteristică a formei de undă asimetrice de comparare/PWM este aceea că o modificare a valorii în registrul de comparare afectează numai un front la pulsului comparare/PWM.

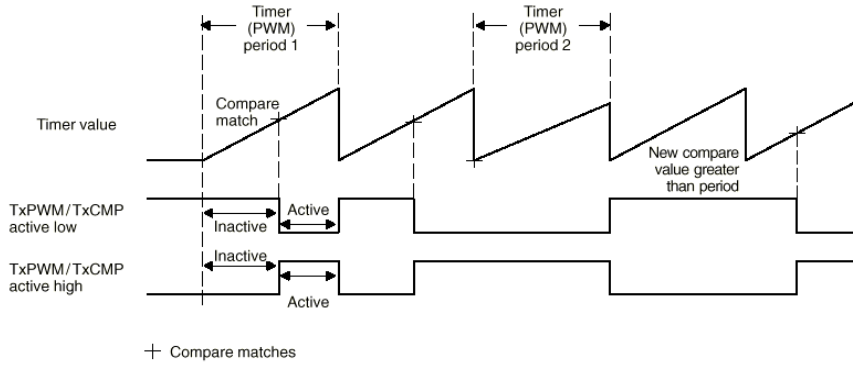


Fig. 7.25. Ieşirea comparare/PVM a timerului GP pentru modul de numărare sus continuu

Generarea formei de undă simetrice

O formă de undă simetrică aşa cum este cea prezentată în figura 7.26 este generată atunci când timerul este în modul de numărare un singur ciclu sau continuu sus/jos. Când timerul este într-unul din aceste două moduri, ieşirea generatorului de forme de undă se schimbă în felul următor:

- zero înainte ca operaţia de numărare să pornească;
- rămâne neschimbat până la producerea unei egalităţi de comparare;
- comută la apariţia primei egalităţii de comparare;
- rămâne neschimbată până se întâlneşte a doua egalitate de comparare;
- comută la apariţia celei de-a doua egalităţii de comparare
- rămâne neschimbat până la sfârşitul perioadei;
- resetează la zero la sfârşitul perioadei dacă nu s-a întâlnit a doua egalitate la comparare şi o nouă valoare de comparare pentru perioada următoare este diferită de zero.

Ieşirea este setată la 1 la începutul perioadei şi rămâne 1 până la a doua egalitate de comparare dacă valoarea de comparare este 0. După prima tranziţie de la 0 la 1,

ieşirea rămâne 1 până la sfârşitul perioadei dacă valoarea comparatorului este 0 pentru restul perioadei. Când acest lucru se întâmplă ieşirea nu trebuie să reseteze la zero dacă valoarea de comparare pentru următoarea perioadă este în continuare zero. Acest lucru este făcut pentru a asigura generarea punsurilor PWM cu factor de umplere între 0% şi 100% fără nici un impuls parazit. Prima tranziție nu trebuie să se producă dacă valoarea comparatorului este mai mare sau egală cu valoarea perioadei pentru prima jumătate a perioadei. Ieşirea va comuta totuşi când o egalitate de comparare se produce în ce-a de-a doua jumătate a perioadei. Această eroare în comutarea ieşirii adesea ca rezultat a erorilor de calcul din programele de aplicații este corectată la sfârşitul perioadei din cauză că ieşirea resetează la zero numai dacă noua valoare de comparare pentru perioada următoare este zero. Dacă se întâmplă mai târziu, ieşirea rămâne 1, ceea ce o pune din nou în starea corectă.
Notă: Logica de ieşire stabileşte polaritatea pentru toți pinii de ieşire.

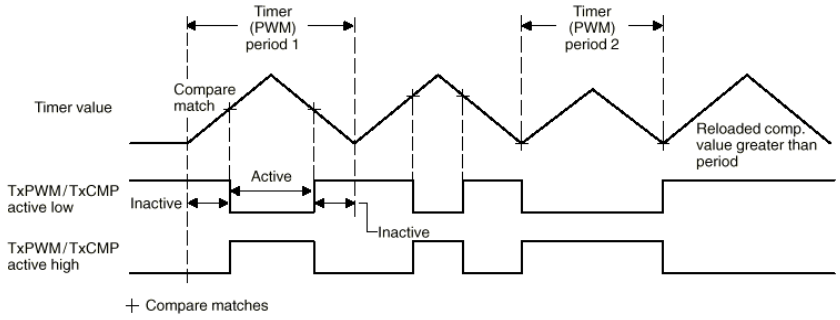


Fig. 7.26. Ieşirea comparare/PWM a timerului GP în modul de numărare continuu sus/jos

Logica de ieşire

Logica de ieşire permite stabilirea condițiilor pentru ieşirea comparare/PWM în scopul comenzii diferitelor tipuri de dispozitive electronice de putere. Se pot specifica pentru ieşirea comparare/PWM următoarele condiții: activă sus, activă jos, forțată sus sau forțată jos prin configurarea adecvată a biților din GPTCON.

Pentru starea activă, cele două stări, activă sus şi activă jos sunt complementare. Stare activă în 1 înseamnă că ieşirea furnizează 1 când este activă şi zero în stare inactivă. Ieşirea comparare/PWM este setată în 1 (sau 0) imediat ce biții corespunzători din GPTCON sunt setați astfel încât starea ieşirii să fie forțată sus (sau jos).

Tabelele următoare descriu starea ieşirii pentru diverse moduri de numărare.

TABELUL 7.4. Ieşirea de comparare a timerului GP în mod numărare sus un ciclu sau continuu

Desfăşurarea evenimentelor	**Starea ieşirii de comparare**
Înainte de egalitate comparare	Inactivă
La egalitate la comparare	Setată activă
La egalitate la perioadă	Setată inactivă

TABELUL 7.5.
Ieşirea de comparare a timerului GP în mod numărare sus/jos un ciclu sau continuu

Desfăşurarea evenimentelor	**Starea ieşirii**
Înainte de prima egalitate de comparare	Inactivă
La prima egalitate la comparare	Setată activă
La a doua egalitate la comparare	Setată inactivă
După a doua egalitate la comparare	Inactivă

Toate ieşirile comparare/PWM sunt puse în înaltă impedanță când se produce unul din următoarele evenimente:

- GPTCON[6] este setat la 1 de către software;
- PDPINT este pus low şi este nemascat;
- se comandă RESET;
- când operația este dezactivată pentru timerele GP.

Ieşirea de comparare în modul de numărare direcțional sus/jos

Când timerul este în modul de numărare direcțional sus/jos nu se produce nici o tranziție la ieşirile de comparare. În mod similar, nu se produce nici o tranziție la ieşirile de comparare asociate cu unitățile de comparare completă când timerul GP 1 este în modul de numărare direcțional sus/jos. Nu se produce nici o tranziție la ieşirile asociate unităților de comparare simplă atunci când timerul GP selectat ca bază de timp pentru acestea este programat în modul de numărare direcțional sus/jos. Setarea fanioanelor ce întrerupere la comparare şi generarea cererilor de întrerupere nu depinde de modul de numărare în care este setat timerul GP.

Calcularea timpului activ/inactiv

Pentru modurile de numărare sus, valoarea registrului de comparare reprezintă timpul scurs între începutul perioadei şi întâlnirea preimei egalități la comparare; rezultă că aceasta este lungimea fazei inactive. Timpul scurs este egal cu perioada ceasului prescalat înmulțită cu TxCMPR. Rezultă că lungimea fazei active, lățimea impulsului, este dată de relația: TxPR – TxCMPR+1 ciclii ai ceasului scalat.

Pentru modurile de numărare sus/jos, registrul de comparare poate avea valori diferite pentru numărarea jos şi numărarea sus. Lungimea fazei active, adică lățimea impulsului, este dată de relația: TxPR – $TxCMPR_{UP}$ + TxPR – $TxCPMR_{DN}$ ciclii de ceas prescalat.

Când valoarea TxCMPR este zero atunci ieşirea de comparare a timerului GP este activă pe întreaga perioadă dacă timerul este în modul de numărare sus. Pentru modurile de numărare sus/jos, ieşirea de comparare este activă de la începutul perioadei dacă $TxCMPR_{UP}$ este zero. Ieşirea rămâne activă după sfârşitul perioadei dacă $TxCMPR_{DN}$ este de asemenea zero.

Lungimea fazei active (lățimea impulsului) este zero atunci când valoarea TxCMPR este mai mare decât TxPR pentru modurile de numărare sus. Pentru modurile de numărare sus/jos prima tranziție este pierdută când $TxCMPR_{UP}$ este mai mare sau

egală cu TxPR. Similar, a doua tranziție este pierdută când $TxCMPR_{DN}$ este mai mare sau egală cu TxPR. Ieşirea de comparare a timerului GP este inactivă pe întreaga perioadă dacă ambele valori de comparare $TxCMPR_{UP}$ şi $TxCMPR_{DN}$ sunt mai mari sau egale cu TxPR pentru modurile de numărare sus/jos.

Registrele de control ai timerelor GP (GPTCON şi TxCON)

Registrul de control timer GP (GPTCON) – adresa 7400h

15	14	13	12-11	10-9	8-7
T3STAT	T2STAT	T1STAT	T3TOADC	T2TOADC	T1TOADC
R-1	R-1	R-1	RW-0	RW-0	RW-0

7	5-4	3-2	1-0
TCOMPOE	T3PIN	T2PIN	T1PIN
RW-0	RW-0	RW-0	RW-0

Notă: R=citire, W=scriere, -0=valoare după reset

Bit 15 T3STAT. Stare timer GP 3
0 = Numără jos
1 = Numără sus

Bit 14 T2STAT. Stare timer GP 2
0 = Numără jos
1 = Numără sus

Bit 13 T1STAT. Stare timer GP 1
0 = Numără jos
1 = Numără sus

Biții 12-11 T3TOADC. Porneşte conversie ADC de către eveniment timer GP 3
00 = Nici un eveniment nu porneşte ADC
01 = Întreruperea la depăşirea inferioară porneşte ADC
10 = Întreruperea la perioadă porneşte ADC
11 = !ntreruperea la comparare porneşte ADC

Biții 10-9 T2TOADC. Porneşte conversie ADC de către eveniment timer GP 2
00 = Nici un eveniment nu porneşte ADC
01 = Întreruperea la depăşirea inferioară porneşte ADC
10 = Întreruperea la perioadă porneşte ADC
11 = Întreruperea la comparare porneşte ADC

Biții 8-7 T1TOADC. Porneşte conversie ADC de către eveniment timer GP 1
00 = Nici un eveniment nu porneşte ADC
01 = Întreruperea la depăşirea inferioară porneşte ADC
10 = Întreruperea la perioadă porneşte ADC
11 = Întreruperea la comparare porneşte ADC

Bit6 TCOMPOE. Activează ieşirea de comparare. PDPINT activ scrie zero la acest bit.
0 = Dezactivează toate cele trei ieşiri de comparare ale timerelor GP (le pune în starea de înaltă impedanță)
1 = Activează toate cele trei ieşiri de comparare ale timerelor GP

Biții 5-4 T3PIN. Polaritatea ieşirii de comparare a timerului GP 3

00 = forțată în starea jos
01 = activă în zero
10 = activă în unu
11 = forțată în unu

Biții 3-2 T2PIN. Polaritatea ieşirii de comparare a timerului GP 2

00 = forțată în starea jos
01 = activă în zero
10 = activă în unu
11 = forțată în unu

Biții 1-0 T1PIN. Polaritatea ieşirii de comparare a timerului GP 1

00 = forțată în starea jos
01 = activă în zero
10 = activă în unu
11 = forțată în unu

Registrul de control al timerului GP (TxCON; x=1,2 şi 3) – regiştrii sunt la adresele 7404h, 7408h şi 740Ch.

15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Free	Soft	TMODE 2	TMODE 1	TMODE 0	TPS2	TPS1	TPS0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TSWT1	TENABLE	TCLKS1	TCLKS 0	TCLD1	TCLD0	TECMPR	SELT1PR
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Notă: R=citire, W=scriere, -0=valoare după reset

Biții 15-14 Free, Soft. Biți de control a emulării

00 = Stop imediat ce emularea a fost oprită
01 = Stop după ce perioada curentă a timerului este completă la oprirea emulării
10 = Operarea nu este afectată de oprirea emulării
11 = Operarea nu este afectată de oprirea emulării

Biții 13-11 TMODE2-TMODE0. Selectarea modului de numărare

000 = Stop/hold
001 = Modul numărare directă (sus) o singură dată (un ciclu)
010 = Modul numărare continuă directă (sus)
011 = Mod numărare bidirectional (up/down)
100 = Modul numărare bidirectională o singură dată (un ciclu)
101 = Mod numărare continuu bidirecțional
110 = Rezervat. Rezultat imprevizibil

111 = Rezervat. Rezultat imprevizibil

Biții 10-8 TPS2-TPS0. Factor de divizare a ceasului de intrare

000 = x/1
001 = x/2
010 = x/4
011 = x/8
100 = x/16
101 = x/32
110 = x/64
111 = x/128

x – frecvența de ceas a unității centrale (CPU)

Bit 7 TSWT1. (Pornirea timer GP cu timer GP 1). Porneşte timerul cu bitul de validare al timerului GP 1. Acest bit este rezervat la T1CON

0 = Utilizează propriul bit TENABLE

1 = Utilizează bitul TENABLE din T1CON pentru validarea sau invalidarea operației ignorând propriul bit TENABLE

Bit 6 TENABLE. Validare timer. În modul un singur ciclu numărare directă sau bidirecțională acest bit este şters la zero de către timer după ce se completează o perioadă a operației

0 = Dezactivează operarea timerului: pune timerul în starea de oprire (hold) şi resetează contorul de divizare (prescalare)

1 = Activează operarea timerului

Bit 5-4 TCLKS1, TCLKS0. Selectarea sursei de ceas.

00 = Intern

01 = Extern

10 = Cascadare cu timerul GP 2. (este aplicabil numai la T3CON când timerele GP 2 şi 3 sunt legate pentru a forma un timer de 32 biți; rezervat în T1CON şi T2CON; ilegal dacă SELT1PR=1 – adică rezultatul este imprevizibil)

11 = Circuit pentru impulsuri codate în cuadratură (Aplicabil numai la T2CON şi T3CON; rezervat în T1CON; ilegal dacă SELT1PR este 1 – adică rezultatul este imprevizibil)

Biții 3-2 TCLD1, TCLD0. Condiția de reîncărcare a registrului (activ) pentru comparator

00 = Când contorul este 0

01 = Când valoarea contorului este zero sau egală cu valoarea registrului perioadei

10 = Imediat

11 = Rezervat

Bit 1 TECMPR. Validarea comparării timerului

0 = Dezactivează operația de comparare a timerului

1 = Activează operația de comparare a timerului

Bit 0 SELT1PR. Selecția registrului de perioadă. Acest bit este rezervat în T1CON

0 = Utilizează registru propriu de perioadă

1 = Utilizează T1PR ca registru de perioadă ignorând registrul propriu

Notă: Sincronizarea timerelor GP

Două scrieri consecutive în T1CON sunt necesare pentru a asigura sincronizarea timerelor GP când T1CON[6] este folosit pentru validarea timerelor GP 2 sau 3:

1. Configurați toți ceilalți biți cu T1CON[6] setat la zero
2. Validați timerul GP 1 şi astfel timerul GP 2 sau timerele GP 2 şi 3 prin setarea T1CON[6] la unu.

Generarea ieşirilor de comparare şi PWM utilizând timerele GP

Fiecare timer poate fi folosit independent pentru a furniza un canal de ieşire de comparare sau PWM. Rezultă că se pot genera până la trei ieşiri de comparare sau PWM.

Operația de comparare

Pentru a genera o ieşire de comparare modul de operare corespunzător trebuie întâi selectat pentru timerul GP. Pentru aceasta următoarele activități trebuie realizate:

- setarea TxCMPR în concordanță cu valoarea de comparat;
- setarea GPTCON cu tranziția dorită a ieşirii care să se producă la egalitatea de comparare;
- încărcarea TxPR cu valoarea dorită a perioadei, dacă este necesar;
- încărcarea TxCNT cu valoarea inițială a contorului, dacă este necesar;
- setarea TxCON cu modul de numărare specific, sursa de ceas şi pornirea operării.

Operarea PWM

Pentru a genera o ieşire PWM cu un timer GP se selectează modul de numărare sus continuu sau modul de numărare sus/jos continuu. Forme de undă PWM cu fronturi trigerate sau asimetrice sunt generate când este selectat modul de numărare sus continuu. Forme de undă PWM centrate sau simetrice sunt generate când modul de numărare sus/jos este selectat. Pentru a seta timerul GP pentru această operație se efectuează următoarele activități:

- setarea TxPR în concordanță cu perioada dorită a PWM (purtătoarea);
- setarea TxCON în modul de numărare specific, sursa de ceas şi pornirea operației;
- încărcarea TxCMPR cu valorile corespunzătoare cu valorile calculate on-line ale lățimii pulsurilor PWM.

Valoarea perioadei este obținută prin împărțirea perioadei dorite a PWM la perioada ceasului de intrare a timerului GP şi scăzând 1 din valoarea obținută când este selectat modul de numărare sus continuu pentru a genera forme de undă PWM asimetrice. Această valoare este obținută prin împărțirea perioadei dorite pentru PWM la de 2 ori perioada de ceas de intrare a timerului GP când este selectat modul de numărare continuu sus/jos pentru a genera forme de undă PWM simetrice.

Timerul GP poate fi inițializat în acelaşi fel cum s-a arătat în exemplele anterioare. Pe durata timpului de numărare, registrul de comparare al timerului GP este în mod continuu încărcat cu noile valori determinate pentru comparare pentru noile cicluri active determinate.

Resetarea timerului GP

Când apare semnalul RESET se produc următoarele evenimente:

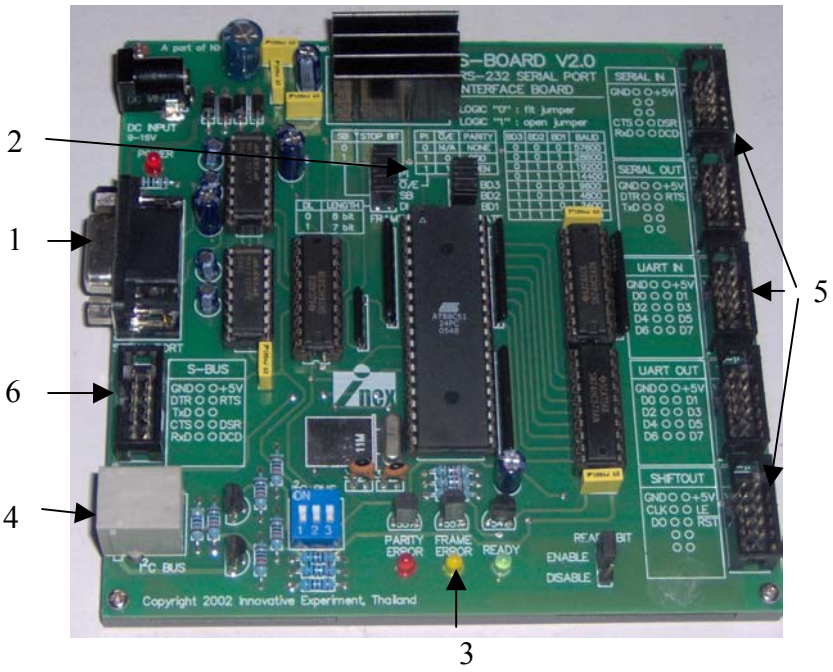
- toți biții registrelor timerelor GP cu excepția bitului de direcție din GPTCON sunt resetați la zero; deci toate operațiile timerelor GP sunt dezactivate. Biții indicatori ai direcției sunt toți setați la 1;
- toate fanioanele de întrerupere ale timerelor sunt resetate la 0;
- toți biții de mascare ai întreruperilor sunt resetați la zero; deci toate întreruperile timerelor GP sunt mascate;
- toate ieşirile de comparare ale timerelor GP sunt puse în starea de înaltă impedanță.

Laboratorul nr. 1

Utilizarea interfeței seriale

În cadrul acestui laborator se va folosi placa S-board de interfață cu portul serial al calculatorului în scopul realizării mai multor experimente privind transmisia serială.

Imaginea plăcii S-board este următoarea:



Placa S-Board permite şi conține:

- interfața cu portul serial RS-232 (1);
- funcția UART controlat prin circuitul UTX8100;
- selectarea ratei de transfer a informației şi a formatului ei prin jamperi (2);
- leduri pentru afişarea stării circuitului şi a erorilor (3);
- circuit de conversie pentru realizarea transmisiei utilizând modul I2C (4);
- Conectori de extensie pentru DATA BUS, UART IN, UART AUT, SHIFT OUT (5) şi S-BUS (6) necesari pentru realizarea conexiunilor cu plăcile din seria EX

Toate semnalele portului paralel ajung la portul S-BUS prin intermediul circuitului MAX 232 şi 74HC541, circuit de tip buffer, care realizează şi protecția la erorile ce pot apărea. Conectorul SERIAL OUTPUT permite controlul unor componente conectate la acest port prin intermediu semnalelor TxD, DTR şi RTS. Pentru recepția unor semnale de la plăcile de extensie se foloseşte portul SERIAL INPUT ce include pinii DCD, CTS, RxD şi DSR.

Pentru realizarea unei comunicații complete prin intermediul portului serial, placa S-Board conține circuitul UART de tip UTX8100 ce realizează conversia semnalului serial preluat de la portul serial în semnal paralel pe 8 biți transmis prin

intermediul pinilor D0-D7 şi invers. Circuitul UTX8100 poate fi controlat din punct de vedere al ratei de transfer a informaţie (8 posibilităţi) cu ajutorul jamperilor (2). De asemenea tot prin intermediul lor se poate seta numărul de biţi transmişi, controlul parităţii transmisiei şi setarea bitului de stop.

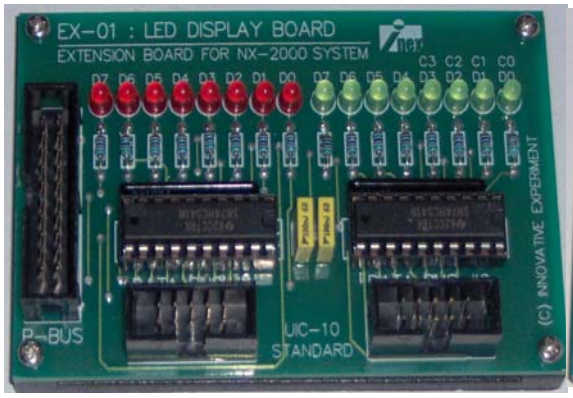
În cazul apariţiei unei erori de transmisie circuitul UTX8100 transmite aceste informaţii (Eroare de paritate PE şi eroare de cadru FE) prin intermediul unor pini de ieşire.

Placa S-Board permite creşterea numărului porturilor de ieşire utilizând placa de extensie EX-09, conectată la portul SHIFTOUT.

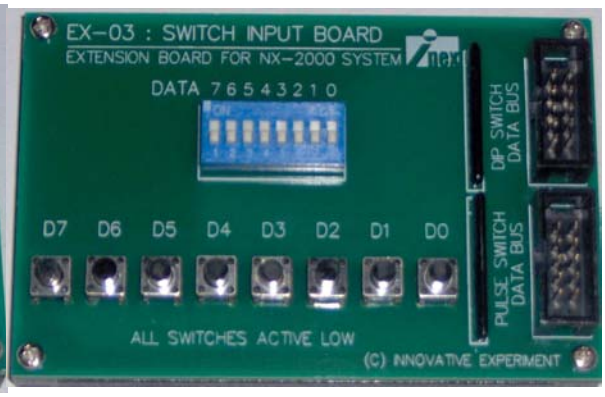
Experimentul nr. 1

Realizarea unei transmisii de semnal serial preluat de portul calculatorului convertit în semnal paralel

Pentru realizarea experimentului se va folosi placa S-Board, placa EX-01 care conţine un afişaj cu leduri pe 2x8 biţi, un calculator care să aibă instalat programul Visual BASIC V5.0 sau mai nouă şi cablul de conexiune IDC-10.



Placa EX-01

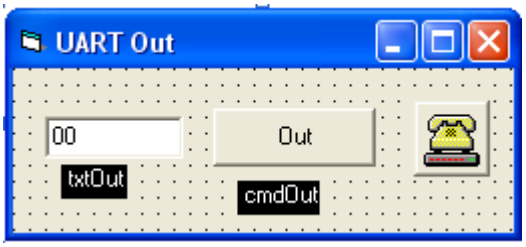


Placa EX-03

Procedura de execuţie a experimentului

Se va defini protocolul de transmisie a semnalului serial dintre calculator şi circuitul UART al plăcii setându-se următorii parametrii: rata de transfer 57600, 8 biţii de date, fără control de paritate şi un bit de stop.

1. Cu ajutorul cablului IDC-10 se va realiza conexiunea între portul UART OUT a plăcii S-Board şi DATA BUS1 a plăcii EX-01;
2. Se setează rata de transfer la 57600 cu ajutorul jamperilor BD1, BD2 şi BD3 puşi pe 0 logic;
3. Se selectează fără control de paritate şi un bit de stop cu ajutorul jamperilor PI şi SB puşi pe 0 logic;
4. Se selectează 8 biţii de date cu ajutorul jampărului DL pus pe 0 logic;
5. Se porneşte calculatorul şi se lansează aplicaţia Visual BASIC.
6. Realizaţi o formă şi editaţi controalele din figura următoare;



7. Scrieți următorul program pentru evenimentul **Form_Load** pentru definirea formatului comunicației:

```
Private Sub Form_Load()
        MSComm1.CommPort = 1
        MSComm1.Settings = „57600,n,8,1”
        MSComm1.PortOpen = True
End Sub
```

8. Scrieți următorul program pentru evenimentul **cmdOut_Click**

```
Private Sub cmdOut_Click ()
        MSComm1.Output = Chr(Val(“&H” & txtOut.Text) Mod 256)
End Sub
```

Formatul datei txtOut este în hexazecimal dar formatul datei transmise prin MSComm1.Output este de tip caracter. Prin urmare se transformă datele prima dată în zecimal, în domeniul 0-256 şi utilizând funcția Chr() se convertesc în date de tip caracter.

Experimentul nr. 2

Conversia unui semnal paralel în semnal serial

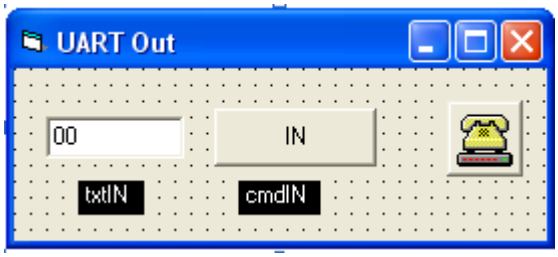
Pentru realizarea experimentului se va folosi placa S-Board, placa EX-03 care conține 16 comutatoare logice de semnal, un calculator care să aibă instalat programul Visual BASIC V5.0 sau mai nouă şi cablul de conexiune IDC-10.

În cadrul acestui experiment se va prelua un semnal paralel emis de placa EX-03 prin intermediul plăcii S-Board care realizează conversia lui în semnal serial.

Procedura de execuție a experimentului

Cu ajutorul cablului IDC-10 se va realiza conexiunea între portul UART IN a plăcii S-Board şi DATA BUS a plăcii EX-03;

1. Se setează rata de transfer la 57600 cu ajutorul jamperilor BD1, BD2 şi BD3 puşi pe 0 logic;
2. Se selectează fără control de paritate şi un bit de stop cu ajutorul jamperilor PI şi SB puşi pe 0 logic;
3. Se selectează 8 biții de date cu ajutorul jampărului DL pus pe 0 logic;
4. Se porneşte calculatorul şi se lansează aplicația Visual BASIC.
5. Realizați o formă şi editați controalele din figura următoare;



6. Scrieți următorul program pentru evenimentul **Form_Load** pentru definirea formatului comunicației:

```
Private Sub Form_Load()
        MSComm1.CommPort = 1
        MSComm1.Settings = „57600,n,8,1”
        MSComm1.PortOpen = True
        MSComm1.DTREnable = False
End Sub
```

7. Scrieți rutina de întârziere cum urmează:

```
Private Sub Delay()
Dim a As Single
        a = Timer + 0.01
        Do While a > Timer
                DoEvents
        Loop
End Sub
```

9. Scrieți următorul program pentru evenimentul **cmdIn_Click**

```
Private Sub cmdIn_Click ()
Dim tmp As String
        MSComm1.DTREnable = True
        Delay
        MSComm1.DTREnable = False
        Delay
        If  MSComm1.InBufferCount > 0 Then
                tmp = MSComm1.Input
                txtIn.Text = Hex(Asc(tmp))
        End If
End Sub
```

Deoarece calculatorul utilizat este mai rapid decât circuitul UART , există posibilitatea ca acesta să nu poată citi datele la timp. De aceea este foarte importantă utilizarea rutinei de întârziere.

Laboratorul nr. 2

Comunicația între sistemele de calcul

1. Introducere

Comunicația între calculatoare se realizează în scopul schimbului de date între sistemele de calcul. Indiferent de ceea ce reprezintă datele comunicate (imagini, texte, baze de date, comenzi, etc.), modul de transmitere a datelor între calculatoare face obiectul unor reglementări internaționale în scopul universalizării schimbului de date între sistemele de calcul.

Comunicarea de date presupune stabilirea legăturii între două sau mai multe calculatoare şi gestionarea şi controlul transferului de informație. O interfață de comunicație reprezintă un set de reguli şi convenții standardizate care specifică modul de realizare a unei legături.

Comunicația între calculatoare poate fi realizată paralel, pe distanțe scurte, sau serial, pe distanțe lungi, evident cu diferențe legate de viteza de transmisie şi de resurse materiale.

Comunicația serială se realizează prin conectare directă între sistemele de calcul, prin interfața serială (comunicație digitală - comunicație prin modem nul), pe distanțe scurte (sub 15 m), sau cu ajutorul modem-urilor (comunicație analogică), de obicei prin linii telefonice, pe distanțe lungi.

Viteza de transmisie pe o linie serială se măsoară în bps (biți pe secundă), iar valorile standardizate pentru aceste viteze sunt: 150 bps, 300 bps, 600 bps, 1200 bps, 2400 bps, 4800 bps, 9600 bps, 19200 bps, 38400 bps, 57600 bps, 115200 bps.

Atunci când se realizează o legătură între două sau mai multe calculatoare, în special în cazul rețelelor de calculatoare, pentru a face distincție între rolul fiecărui calculator, se folosesc termenii de: file server şi client.

File server (numit în continuare server), reprezintă calculatorul care coordonează întreaga activitate iar stațiile de lucru, clienții, sunt reprezentați de către calculatoarele pe care lucrează utilizatorii.

2. Realizarea comunicației între două calculatoare prin intermediul sistemului de operare MS-DOS

Începând cu versiunea 6 a sistemului de operare MS-DOS două calculatoare pot fi legate în scopul schimbului de date, cu ajutorul comenzilor: INTERLNK.EXE şi INTERSVR.EXE care se găsesc în directorul DOS de pe discul C.

Pentru a obține informații cu privire la modul de utilizare al acestor comenzi ele pot fi lansate sub forma: **interlnk /?** şi respectiv **intersvr /?** ,iar pe ecran apar informații sumare cu privire la aceste comenzi.

Pentru informații mai complete se dă comanda **help** şi în ecranul afişat se caută cele două comenzi cu ajutorul tastelor cu săgeți sau cu ajutorul tastelor Page Up sau Page Down. Dacă cursorul a fost poziționat pe una din aceste comenzi, prin apăsarea tastei ENTER se afişează informații despre comanda respectivă.

Aceste programe de comunicație realizează indirectarea discurilor şi al imprimantelor calculatorului server spre calculatorul client.

De exemplu, dacă calculatorul server are discurile A:, B:, C:, iar calculatorul client are discurile A: şi C:, după instalarea driver-ului de comunicație vor apare pe calculatorul client, suplimentar față de situația anterioară, discurile D:, E:, F:, care corespund discurilor A:, B:, respectiv C: de pe calculatorul server.

Ordinea indirectării poate fi schimbată dacă se lansează programul interlnk.exe de pe calculatorul client, aşa cum se va arăta mai jos.

2.1. Realizarea legăturii fizice (hardware)

Realizarea comunicației între două calculatoare presupune realizarea legăturii fizice între calculatoare si lansarea programelor de comunicație. Legătura fizică între calculatoare se realizează prin conectarea interfețelor seriale sau paralele libere, ale celor două calculatoare.

Realizarea conexiunii se face astfel:

- pentru interfața serială conectorii disponibili pot fi conectori cu 9 pini sau cu 25 de pini. Conexiunea se realizează printr-un cablu cu conectori mamă la ambele capete.

Cablurile trebuie să aibă 3 fire pentru transmisiile de date, legăturile necesare fiind: Ground-Ground, Transmit-Receive, şi Receive-Transmit, sau 7 fire pentru a copia fişiere la distanță, conexiunile necesare fiind:

9 pin	25 pin		25 pin	9 pin	
pin 5	pin 7	< >	pin 7	pin 5	(Ground-Ground)
pin 3	pin 2	< >	pin 3	pin 2	(Transmit-Receive)
pin 7	pin 4	< >	pin 5	pin 8	(RTS - CTS)
pin 6	pin 6	< >	pin 20	pin 4	(DSR - DTR)
pin 2	pin 3	< >	pin 2	pin 3	(Receive-Transmit)
pin 8	pin 5	< >	pin 4	pin 7	(CTS - RTS)
pin 4	pin 20	< >	pin 6	pin 6	(DTR - DSR)

Pentru legătura pe interfața paralelă sunt necesari 2 conectori-tată cu 25 de pini la ambele capete. Sunt necesare 11 legături conform tabelului de mai jos:

25 pin			25 pin	
pin 2	<	>	pin 15	
pin 3	<	>	pin 13	
pin 4	<	>	pin 12	
pin 5	<	>	pin 10	
pin 6	<	>	pin 11	
pin 15	<	>	pin 2	
pin 13	<	>	pin 3	
pin 12	<	>	pin 4	
pin 10	<	>	pin 5	
pin 11	<	>	pin 6	
pin 25	<	>	pin 25	(Ground-Ground)

Pentru realizarea lucrării de laborator se vor conecta două calculatoare cu ajutorul cablului cu şapte fire pe interfețele seriale COM 2 şi cu ajutorul cablului cu unsprezece fire pe LPT1.

2.2. Realizarea legăturii prin program (software)

Realizarea legăturii software între cele două calculatoare presupune instalarea driver-ului (programul de comunicație) pe calculatorul client şi lansarea programului intersvr.exe pe calculatorul file server.

Instalarea driver-ului de comunicație se face prin adăugarea unei linii în fişierul config.sys care se găseşte în directorul rădăcină al discului C: şi care este fişierul de configurare al sistemului, citit la lansarea sistemului de operare. În acelaşi timp trebuie să ne asigurăm că în acest fişier comanda: LASTDRIVE=n are valoarea n suficient de mare ca discurile calculatorului server să poată fi indirectate. De exemplu, pentru situația prezentată mai sus, in fişierul config.sys al calculatorului client comanda LASTDRIVE trebuie să fie: **LASTDRIVE=F**

Comanda LASTDRIVE informează sistemul de operare care este numărul maxim de unități disc care pot fi instalate.

Instalarea driver-ului de comunicație presupune, aşa cum s-a arătat, adăugarea la sfârşitul fişierului config.sys a liniei urmatoare:

DEVICE=[drive:][path]INTERLNK.EXE [/DRIVES:n] [/NOPRINTER] [/COM[:][n|address]] [/LPT[:][n|address]] [/AUTO] [/NOSCAN] [/LOW] [/BAUD:rate] [/V]

În care parametri sunt:

[drive:][path]
Specifică localizarea fisşierului INTERLNK.EXE (de obicei C:\DOS\)

Comutatoare (switches):

/DRIVES:n

Specifică numărul de discuri indirectate. Valoarea implicită este 3. Dacă se specifică valoarea zero atunci vor fi indirectate numai imprimantele.

/NOPRINTER
Specifică faptul că imprimantele nu vor fi indirectate la instalarea INTERLNK.EXE. Dacă acest comutator nu este specificat atunci vor fi indirectate toate imprimantele găsite pe server.

/COM[:][n|address]
Specifică portul serial care va fi utilizat pentru transferul de date.
Parametrul n specifică numărul portului serial iar address specifică adresa portului serial.
Dacă omiteți numărul portului sau adresa, programul interlnk de pe calculatorul client citeşte toate porturile seriale şi alege primul port serial pe care-l găseşte conectat la server.
Dacă puneți numai opțiunea /COM şi omiteți opțiunea /LPT atunci calculatorul client caută numai porturile seriale.
Implicit programul interlnk va citi toate porturile seriale şi paralele.

/LPT[:][|address]
Specifică portul paralel care va fi utilizat pentru transferul de date.
Parametrul n specifică numărul portului paralel iar address specifică adresa portului paralel.
Dacă omiteți numărul portului sau adresa programul, interlnk de pe calculatorul client citeşte toate porturile paralele şi alege primul port paralel pe care-l găseşte conectat la server.
Dacă puneți numai opțiunea /LPT şi omiteți opțiunea /COM atunci calculatorul client caută numai porturile paralele.
Implicit programul interlnk va citi toate porturile seriale şi paralele.

/AUTO
Instalează driver-ul în memorie numai când calculatorul client poate stabili o legătură cu server-ul, atunci când calculatorul client este pornit.
Implicit driver-ul este instalat în memorie chiar dacă calculatorul client nu poate stabili o legătură cu calculatorul server.

/NOSCAN
Instalează interlnk.exe în memorie dar împiedică stabilirea unei legături între client şi server în timpul configurării sistemului la încărcarea sistemului de operare.
Implicit calculatorul client încearcă stabilirea unei legături cu calculatorul server imediat după instalarea driver-ului interlnk.exe.

/LOW
Încarcă driver-ul de comunicație în memoria convențională chiar dacă memoria înaltă este liberă.
Implicit interlnk.exe este încărcat în memoria înaltă dacă aceasta este liberă.

/BAUD:rate
Setează valoarea maximă a vitezei de transfer pentru interfața serială.
Valorile permise pentru viteza de transmisie sunt: 9600, 19200, 38400, 57600, şi 115200.
Valoarea implicită a vitezei de transfer este 115200.

/V
Previne conflictele cu ceasul calculatorului.
Această opțiune se va pune dacă aveți o conexiune serială între două calculatoare şi unul dintre ele se opreşte în timp ce se accesează prin interlnk o unitate de disc sau o imprimantă.

Pentru realizarea acestei lucrări de laborator se va scrie în fişierul cvonfig.sys al calculatorului client, la sfârşitul fişierului, următoarea linie:

device=c:/dos/interlnk /noprinter /com:2 /baud:9600 /v

După repornirea calculatorului, prin apăsarea simultană a tastelor CTRL+ALT+DEL se va constata că față de situația anterioară, au mai apărut suplimentar trei unități de disc care nu conțin nici un fişier.

În acest moment se poate stabili o legătură prin lansarea pe calculatorul server a programului intersvr.exe. Sintaxa instrucțiunii intersvr.exe este:

INTERSVR [drive:]...]] [/X=drive:[...]] [/LPT:[n|address]] [/COM:[n|address]] [/BAUD:rate] [/B] [/V]

Pentru copierea fişierelor de pe un calculator pe altul se va folosi comanda

INTERSVR /RCOPY

În care parametri sunt:

[drive:][path]
Specifică localizarea fisşierului INTERSVR.EXE (de obicei C:\DOS\)

Comutatoare (switches):

/X=drive:
Specifică litera sau literele unităților de disc care nu vor fi redirectate.
În acest fel se poate limita accesul calculatorului client la unele discuri de pe server.
Implicit toate unitățile de disc ale serverului vor fi redirectate.

/LPT:[n|address]
Specifică portul paralel care va fi utilizat pentru transferul de date.
Parametrul n specifică numărul portului paralel iar address specifică adresa portului paralel.

Dacă omiteți numărul portului sau adresa, programul intersvr de pe calculatorul server citeşte toate porturile paralele şi alege primul port paralel pe care-l găseşte conectat la client.
Dacă puneți numai opțiunea /LPT şi omiteți opțiunea /COM atunci calculatorul server caută numai porturile paralele.
Implicit programul intersvr va citi toate porturile seriale şi paralele.

/COM:[n|address]
Specifică portul serial care va fi utilizat pentru transferul de date.
Parametrul n specifică numărul portului serial iar address specifică adresa portului serial.
Dacă omiteți numărul portului sau adresa, programul intersvr de pe calculatorul server citeşte toate porturile seriale şi alege primul port serial pe care-l găseşte conectat la client.
Dacă puneți numai opțiunea /COM şi omiteți opțiunea /LPT atunci calculatorul server caută numai porturile seriale.
Implicit programul interlnk va citi toate porturile seriale şi paralele.

/BAUD:rate
Setează valoarea maximă a vitezei de transfer pentru interfața serială.
Valorile permise pentru viteza de transmisie sunt: 9600, 19200, 38400, 57600, şi 115200.
Valoarea implicită a vitezei de transfer este 115200.

/B
Afişează ecranul intersvr în alb-negru. Implicit afişarea se face în culori.

/V
Previne conflictele cu ceasul calculatorului.
Această opțiune se va pune dacă aveți o conexiune serială între două calculatoare şi unul dintre ele se opreşte în timp ce se accesează prin interlnk o unitate de disc sau o imprimantă.

/RCOPY
Programul intersvr realizează copierea fişiere de pe un calculator pe altul, dacă cele două calculatoare sunt legate printr-un modem nul, pe interfața serială cu un cablu cu 7 fire şi comanda MODE este disponibilă pe calculatorul pe care ați instalat interlnk.exe.

În lucrarea de laborator, pe calculatorul server, se va da comanda:

intersvr /com:2 /baud:9600 /v

După lansarea programului intersvr cele două calculatoare sunt conectate împreună.

Pentru redirectarea unei unități de disc a serverului la o unitate client, altfel decât s-a făcut prin indirectarea implicită se va proceda în felul următor: să presupunem că

dorim indirectarea unității de disc c: de pe server la unitatea de disc e: a clientului; comanda care se dă pe calculatorul client, este următoarea:

interlnk e:=c:

Pentru încetarea indirectării se dă comanda pe calculatorul client:

interlnk e:=

3. Desfăşurarea lucrării de laborator

Pentru realizarea lucrării de laborator se vor conecta împreună două calculatoare în modul prezentat mai sus.

Cu ajutorul utilitarului NORTON se vor copia fişiere de pe calculatorul server pe calculatorul client şi invers.

Referatele de laborator vor conține informațiile suplimentare despre aceste comenzi, obținute cu comanda help şi modul în care răspunde utilitarul NORTON.

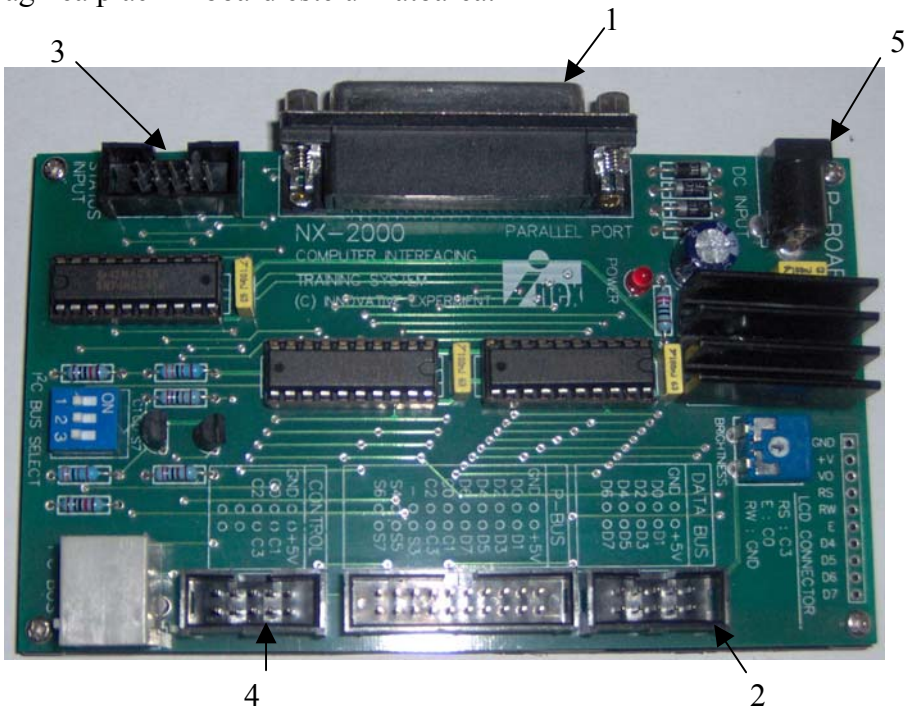
Se va crea un program scurt in C++ pe calculatorul server şi apoi acest program va fi lansat de pe calculatorul client. Referatul va conține observațiile făcute.

Laboratorul nr. 3

Utilizarea interfeței paralele

În cadrul acestui laborator se va folosi placa P-board de interfață cu portul paralel al calculatorului în scopul realizării mai multor experimente privind transmisia paralelă.

Imaginea plăcii P-board este următoarea:



Placa U-Board se va conecta direct la calculator prin intermediul portului paralel (1) şi în plus are un circuit de buffer pentru prevenirea apariției erorilor de transmisie. De asemenea placa conține circuit de conversie pentru magistrală I2C. Ea asigură 3 porturi pentru transferul semnalului de la portul paralel.

1. Data Port (2) – are 8 pini de semnal denumiți D0 – D7. Este un port numai de ieşire a datelor. Toate semnalele de la portul paralel sunt transmise la acest conector. Aceste semnale sunt asigurate şi la portul P-BUS (3).

2. Control port (4) – conține 4 pini de semnal denumiți C0 – C3. C1 şi C3 sunt pini cu logică inversă. Este un port de ieşire identic ca Data Port. Pinii sunt prescrişi în standardul UIC-10 dar sunt utilizați numai 4 pini şi combinații de semnale în P-BUS.

În plus pinii portului de control sunt utilizați ca pini de semnal ai magistralei I2C. C1 pentru controlul SCL(Serial clock), C0 controlul SDA (Serial data output).

3. Status port (3) – are 5 pini denumiți S3 – S7. Pinii sunt prescrişi în standardul UIC-10 dar sunt utilizați numai 5 pini şi combinații de semnale în P-BUS. În plus pinul S7 este utilizat ca pin de semnal ai magistralei I2C.

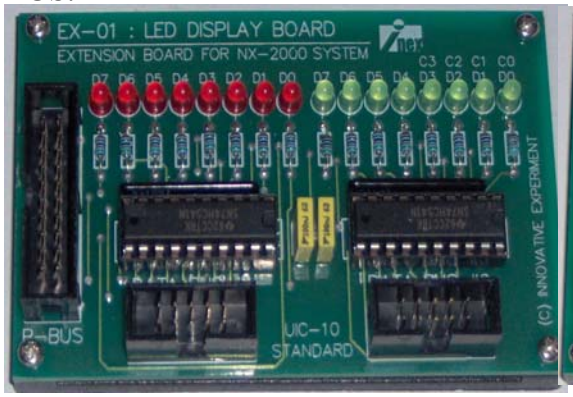
P-Board necesită o tensiune de alimentare de 9-12V/ 500 mA de la o sursă externă aplicată prin intermediul conectorului 5. Placa conține un circuit stabilizator de tensiune de 5 V, necesară alimentării circuitelor de pe placă sau a altor plăci conectate prin intermediul porturilor P-BUS, DATA BUS şi I2C.

Experimentul nr. 1

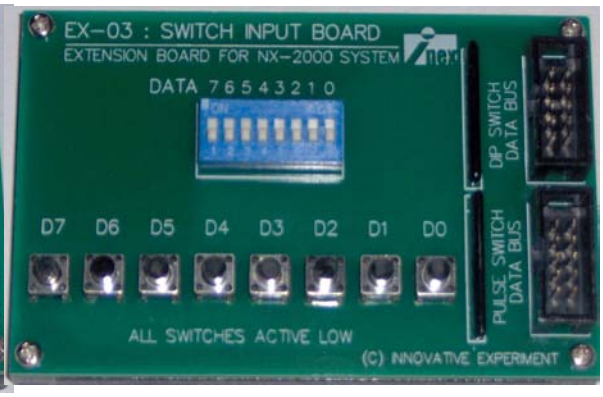
Trimiterea de date de ieşire utilizând programul Visual BASIC prin portul de date al plăcii P-Board

Pentru realizarea experimentului se va folosi placa P-Board, placa EX-01 care conține un afişaj cu leduri pe 2x8 biți, un calculator care să aibă instalat programul Visual BASIC V5.0 sau mai nouă şi cablul de conexiune IDC-10.

Experimentul presupune realizarea aprinderii şi stingerii ledurilor de pe placa EX-01 prin comenzi date prin intermediul softului Visual BASIC, transmise prin intermediul portului paralel al calculatorului şi a plăcii P-Board utilizând portul DATA BUS.



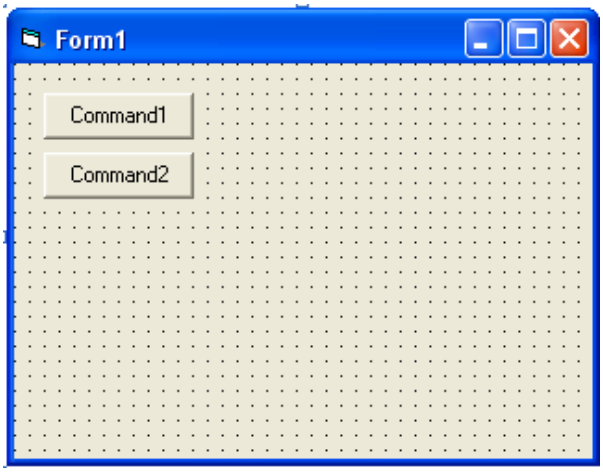
Placa EX-01



Placa EX-03

Procedura de execuție a experimentului:

1. Se conectează placa P-Board la portul paralel al calculatorului şi cu placa EX-01 utilizând cablul IDC-10 prin intermediul portului DATA BUS.
2. Se porneşte calculatorul şi se lansează aplicația Visual BASIC.
3. Din meniul **Project al** softului selectați **Add File** pentru includerea fişierului INPOUT32.BAS în proiect.
4. Plasați două butoane de comandă în fereastra formei Form1, ca în figură



5. Daţi dublu click pe butonul Command 1 pentru introducerea următorului cod sursă în **Viev Code menu :**

```
Private Sub Command1_Click()
Out &H378, &HFF
End Sub
```

6. Aceiaşi operaţie trebuie efectuată şi pentru butonul Command2:

```
Private Sub Command2_Click()
Out &H378, 0
End Sub
```

7. Conectaţi placa P.Board la sursa de tensiune.
8. Executaţi programul. Daţi click pe butonul Command1. Observaţi efectul acestei comenzi asupra ledurilor de pe placa EX-01. Toate ledurile se vor aprinde.
9. Daţi click pe butonul Command2. Observaţi efectul acestei comenzi asupra ledurilor de pe placa EX-01. Toate ledurile se vor stinge.
   Explicaţi aceste comenzi plecând de la faptul că &H378 este adresa portului DATA exprimată în bază 16(378) iar &HFF semnalul trimis de aprindere a ledurilor (FF – 1111 1111, 1 LED aprins, 0 LED stins).
10. Dacă experimentul s-a desfăşurat cu succes, schimbaţi valoarea trimisă spre placa EX-01, de exemplu OUT &H378, &H55. Ledurile vor fi aprinse şi stinse alternativ deoarece 55 în bază 16 este 01010101 în bază 2.

Experimentul nr. 2

Trimiterea de date de ieşire utilizând programul Visual BASIC prin portul de control al plăcii P-Board

Pentru realizarea experimentului se va folosi placa P-Board, placa EX-01 care conţine un afişaj cu leduri pe 2x8 biţi, un calculator care să aibă instalat programul Visual BASIC V5.0 sau mai nouă şi cablul de conexiune IDC-10.

Experimentul presupune realizarea aprinderii şi stingerii ledurilor de pe placa EX-01 prin comenzi date prin intermediul softului Visual BASIC, transmise prin intermediul portului paralel al calculatorului şi a plăcii P-Board utilizând portul CONTROL BUS.

Procedura de execuţie a experimentului:

1. Se conectează placa P-Board la portul paralel al calculatorului şi cu placa EX-01 utilizând cablul IDC-10 prin intermediul portului CONTROL BUS.
2. Se porneşte calculatorul şi se lansează aplicaţia Visual BASIC.
3. Se utilizează programul de la experimentul 1 care se modică în felul următor.
4. Daţi dublu click pe butonul Command 1 pentru introducerea următorului cod sursă în **Viev Code menu :**

```
Private Sub Command1_Click()
Out &H37A, &HF4
End Sub
```

5. Aceiaşi operaţie trebuie efectuată şi pentru butonul Command2:

```
Private Sub Command2_Click()
Out &H37A, &HB
End Sub
```

6. Conectaţi placa P.Board la sursa de tensiune.
7. Executaţi programul. Daţi click pe butonul Command1. Observaţi efectul acestei comenzi asupra ledurilor de pe placa EX-01. Numai 4 leduri se vor aprinde deoarece portul de control are numai 4 biţi corespunzători pinilor C0 – C3.
8. Daţi click pe butonul Command2. Observaţi efectul acestei comenzi asupra ledurilor de pe placa EX-01. Toate ledurile se vor stinge deoarece C0, C1 şi C3 sunt pini cu logică inversă.

Trimiterea datelor prin Control port este similară cu trimiterea prin Data port cu diferenţa că cele două porturi au adrese diferite (&H37A şi &H378). Altă diferenţă este dată de numărul diferit de pini de date: 4 corespunzător celor 4 biţi inferiori la Control port şi 8 la Data port. Pinii C0, C1 şi C3 ai portului de control sunt cu logică inversă.

Experimentul nr. 3

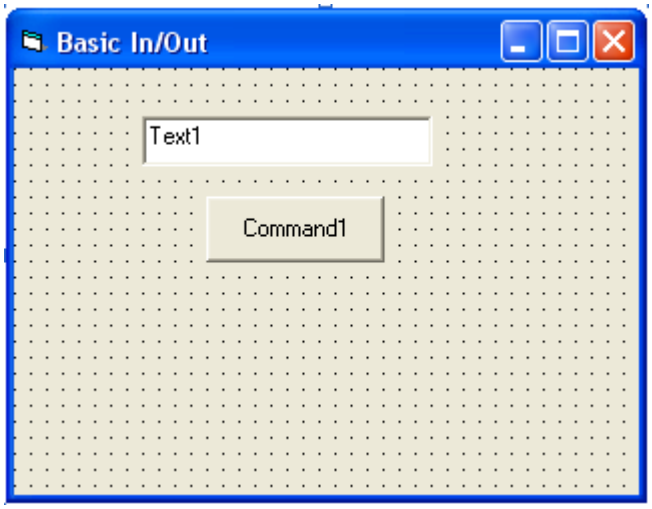
Citirea de date utilizând portul STATUS al plăcii P-Board şi programul Visual BASIC

Pentru realizarea experimentului se va folosi placa P-Board, placa EX-03 care conţine 16 comutatoare logice de semnal, un calculator care să aibă instalat programul Visual BASIC V5.0 sau mai nouă şi cablul de conexiune IDC-10.

Experimentul presupune citirea unei informaţii trimisă de pe placa EX-03 prin intermediul softului Visual BASIC, transmisă prin intermediul portului paralel al calculatorului şi a plăcii P-Board utilizând portul STATUS .

Procedura de execuţie a experimentului:

1. Se conectează placa P-Board la portul paralel al calculatorului şi cu placa EX-03 utilizând cablul IDC-10 prin intermediul portului STATUS INPUT.
2. Se porneşte calculatorul şi se lansează aplicaţia Visual BASIC.
3. Realizaţi o formă, plasaţi un buton de comandă şi o căsuţă de text necesară afişării datelor citite.



4. Adăugaţi fişierul Inpout32.dll în program.
5. Daţi dublu click pe butonul Command 1 pentru introducerea următorului cod sursă în **Viev Code menu :**

```
Private Sub Command1_Click()
Text1.Text = Inp(&H379)
End Sub
```

6. Conectaţi placa P-Board la sursa de tensiune.
7. Executaţi programul. Schimbaţi valoarea logică generată de placa EX-03 prin intermediul comutatoarelor logice şi observaţi efectul în fereastra de text.

Dacă la placa EX-03 se efectuează modificări la comutatoarele D3-D7 valoarea se va schimba în căsuţa de text dar modificările făcute la comutatoarele D0-D2 nu vor avea nici un efect deoarece portul STATUS va transmite numai primii 5 biţi superiori din cei 8 prin pinii S3-S7.

8. Pentru a se realiza o citire mai corectă a biţilor transmişi de la placa EX-03 se poate modifica comanda în felul următor

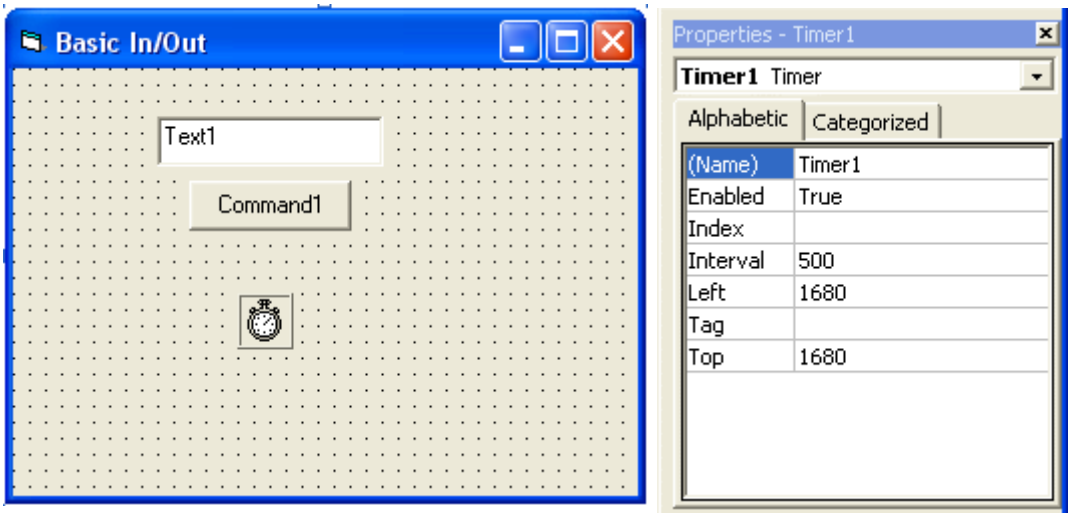
```
Text1.Text = Hex$ (Inp(&H379) And &HF8 Xor &H80)
```

Biţii S0, S1 şi S2 sunt 0 iar valoarea bitului S7 trebuie inversată.

9. Pentru citirea valorilor utilizatorul trebuie să de click tot timpul pe butonul Command1. Pentru a se realiza o citire automată se poate folosi linia de comandă următoare:

```
Private Sub Timer1_Timer ()
Text1.Text = Hex$ (Inp(&H379) And &HF8 Xor &H80)
End Sub.
```

După ce programul rulează 0,5 s el va sări în subrutina Timer1_Timer () pentru a citi valorile semnalelor de la intrare. Dacă au apărut modificări ele vor fi afişate în fereastra de text imediat.



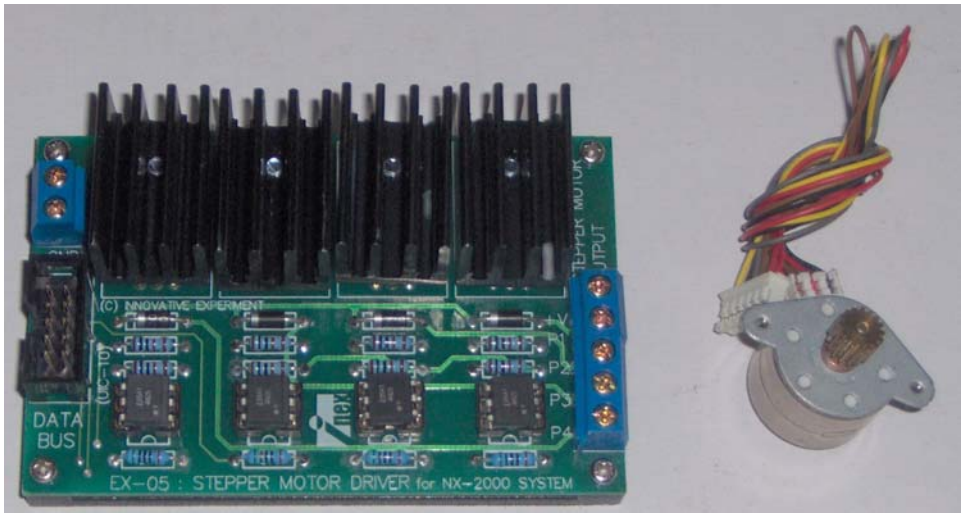
Pentru a realiza o viteză mai mare de citire se va seta valoarea intervalului de timp mai mică în fereastra **Properties – Timer1** prezentată mai sus.

Laboratorul nr. 4

Comanda unui motor pas cu pas prin interfața paralelă

În cadrul acestui laborator se va experimenta comanda unui motor pas cu pas cu ajutorul calculatorului utilizând placa P-board de interfață cu portul paralel al calculatorului şi placa EX-05 destinată special comenzii motorului.

Placa EX-05 şi motorul pas cu pas comandat sunt prezentate în figura următoare:



Placa EX-05 permite comanda unui motor pas cu pas unipolar utilizând semnale de comandă primite la conectorul DATA BUS de la portul paralel al calculatorului prin intermediul plăcii P-Board. Semnalele primite sunt transmise prin intermediul unor optocuploare, tranzistorelor finale care comandă mişcarea motorului.

Pentru punerea în mişcare a motorului pot fi folosite trei metode: rotirea cu un pas complet, jumătate de pas sau un micropas. Pentru aceasta se pot folosi 4 biți ca date de intrare pentru realizarea rotirii motorului.

Placa EX-05 necesită alimentare de la o sursă de tensiune continuă externă.

Experimentul nr. 1

Comanda unui motor pas cu pas pe o singură fază

Pentru realizarea experimentului se va folosi placa P-Board, placa EX-05 , un calculator care să aibă instalat programul Visual BASIC V5.0 sau mai nouă, o sursă de tensiune continuă +12 V 2A necesară alimentării plăcii EX-05, un motor pas cu pas unipolar 12 V 100 Ω 7.5 grade/pas şi cablul de conexiune IDC-10.

Procedura de execuție a experimentului

Datele de intrare pentru comanda pe o singură fază a motorului pas cu pas sunt împărțite în două grupuri. Primul format din comenzile 1, 2, 4, şi 8 necesare rotirii motorului în sensul antiorar iar celălalt grup în sensul orar cu comenzile 8, 4, 2, şi 1. Aceste comenzi sunt prezentate în tabelele de mai jos.

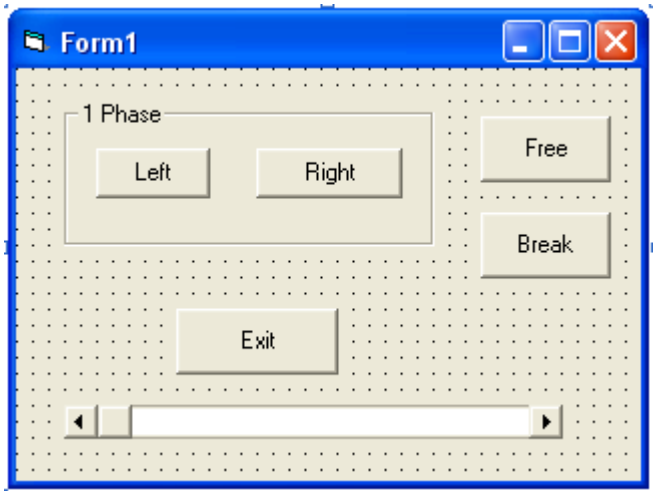
Pas	Faza 4	Faza 3	Faza 2	Faza 1
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	1	0	0	0

Rotire la stânga

Pas	Faza 4	Faza 3	Faza 2	Faza 1
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Rotire la dreapta

1. Se porneşte calculatorul şi se lansează aplicația Visual BASIC.
2. Realizați o formă şi editați controalele din figura următoare;



3. Scrieți codul sursă pentru butonul de comandă **Command1 (Left):**

```
Private Sub Command1_Click():
        Left = False
        Rights = True
        Do
                DoEvents
                Out &H378, 1
                        Call delay
                Out &H378, 2
                        Call delay
                Out &H378, 4
                        Call delay
                Out &H378, 8
                        Call delay
```

```
        Loop Until Lefts = True
    End Sub
```

4. Conectați placa P-Board cu placa EX-05 cu ajutorul cablului IDC-10 prin intermediul portului DATA BUS. Conectați motorul pas cu pas la placa EX-05 şi verificați că fazele sunt conectate corect.
5. Alimentați plăcile EX-05 şi P-Board .
6. Rulați programul creat.
7. Dând click pe butonul Left, programul va trimite valorile 1, 2, 4, şi 8 în ordine. Fiecare comandă este separată printr-o rutină de întârziere. În acelaşi timp, comanda **Do … Loop Until** va verifica starea comenzii **Rights** . Observați rotirea motorului.
8. În realizarea rutinei de întrerupere pot fi utilizate mai multe metode. Una din ele presupune utilizarea bazei de timp a calculatorului. Avantajul metodei este că timpul de întârziere este identic la toate calculatoarele dar dezavantajul este dat de limitarea de viteză. Viteza de execuție a programului ar putea să nu permită execuția unor condiții corect. Valoarea minimă este de 0,01 secunde. Un program simplu pentru realizarea subrutinei este prezentat mai jos:

```
Sub delay()
        Times = Timer
        Do
                DoEvents
        Loop Until Timer >= Times + 0.01
End Sub
```

Această rutină verifică valoarea variabilei **Timer**. Aceasta este o valoare internă a calculatorului **Times** + 0,01 în sec.

9. Altă metodă de realizare a rutinei de întârziere este prezentată mai jos. Această metodă creşte viteza de lucru dar depinde de calculatorul utilizat.

```
Sub delay()
        For i = 1 To HScroll1.Value
                DoEvents
        Next i
End Sub
```

Acestă rutină utilizează comanda **For … Next** şi cursorul **Hscroll** pentru ajustarea vitezei. În cazul în care perioada de timp a buclei este mare şi viteza programului scade, inserați comanda **Doevents** în buclă.

10. Codul sursă pentru butonul de comandă **Command2 (Rights):**

```
Private Sub Command2_Click():
        Left = True
        Rights = False
```

```
Do
        DoEvents
        Out &H378, 8
                Call delay
        Out &H378, 4
                Call delay
        Out &H378, 2
                Call delay
        Out &H378, 1
                Call delay
    Loop Until Rights = True
End Sub
```

Operațiile şi explicațiile sunt similare ca rotirea la stânga.

Experimentul nr. 2

Comanda unui motor pas cu pas pe două faze

Pentru realizarea experimentului se va folosi placa P-Board, placa EX-05 , un calculator care să aibă instalat programul Visual BASIC V5.0 sau mai nouă, o sursă de tensiune continuă +12 V 2A necesară alimentării plăcii EX-05, un motor pas cu pas unipolar 12 V 100 Ω 7.5 grade/pas şi cablul de conexiune IDC-10.

Procedura de execuție a experimentului

Datele de intrare pentru comanda pe două faze a motorului pas cu pas sunt împărțite în două grupuri. Primul format din comenzile 9, 3, 6, şi 12 necesare rotirii motorului în sensul antiorar iar celălalt grup în sensul orar cu comenzile 12, 6, 3, şi 9. Aceste comenzi sunt prezentate în tabelele de mai jos.

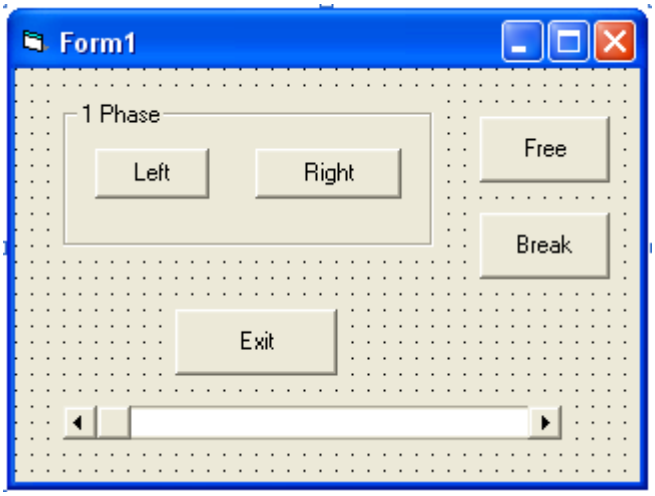
Pas	Faza 4	Faza 3	Faza 2	Faza 1
1	1	0	0	1
2	0	0	1	1
3	0	1	1	0
4	1	1	0	0

Rotire la stânga

Pas	Faza 4	Faza 3	Faza 2	Faza 1
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

Rotire la dreapta

1. Se porneşte calculatorul şi se lansează aplicația Visual BASIC.
2. Realizați o formă şi editați controalele din figura următoare;



3. Scrieți codul sursă pentru butonul de comandă **Command1 (Left):**

```
Dim i As Integer
Dim Lefts, Rights as Boolean
Private Sub Command1_Click():
        Left = False
        Rights = True
        Do
                DoEvents
                Out &H378, 9
                        Call delay
                Out &H378, 3
                        Call delay
                Out &H378, 6
                        Call delay
                Out &H378, 12
                        Call delay
        Loop Until Lefts = True
End Sub
```

4. Codul sursă pentru butonul de comandă **Command2 (Rights):**

```
Private Sub Command2_Click():
        Left = True
        Rights = False
        Do
                DoEvents
                Out &H378, 12
                        Call delay
                Out &H378, 6
                        Call delay
                Out &H378, 3
```

```
                Call delay
            Out &H378, 9
                Call delay
        Loop Until Rights = True
    End Sub
```

5. Conectați placa P-Board cu placa EX-05 cu ajutorul cablului IDC-10 prin intermediul portului DATA BUS. Conectați motorul pas cu pas la placa EX-05 şi verificați că fazele sunt conectate corect.
6. Alimentați plăcile EX-05 şi P-Board .
7. Rulați programul creat.
8. Observați rotirea motorului şi faceți comparația cu experimentul 1.

Comanda motorului pas cu pas pe două faze determină obținerea unui cuplu mai mare la motor dar consumul de energie este mai mare.

Experimentul nr. 3

Comanda unui motor pas cu pas pentru rotirea cu jumătate de pas

Pentru realizarea experimentului se va folosi placa P-Board, placa EX-05 , un calculator care să aibă instalat programul Visual BASIC V5.0 sau mai nouă, o sursă de tensiune continuă +12 V 2A necesară alimentării plăcii EX-05, un motor pas cu pas unipolar 12 V 100 Ω 7.5 grade/pas şi cablul de conexiune IDC-10.

Procedura de execuție a experimentului

Datele de intrare pentru această comandă a motorului pas cu pas sunt împărțite în două grupuri de câte 8 valori. Primul format din comenzile 9, 1, 3, 2, 6, 4, 12 şi 8 necesare rotirii motorului în sensul antiorar iar celălalt grup în sensul orar cu comenzile 8, 12, 4, 6, 2, 3, 1 şi 9. Aceste comenzi sunt prezentate în tabelele de mai jos.

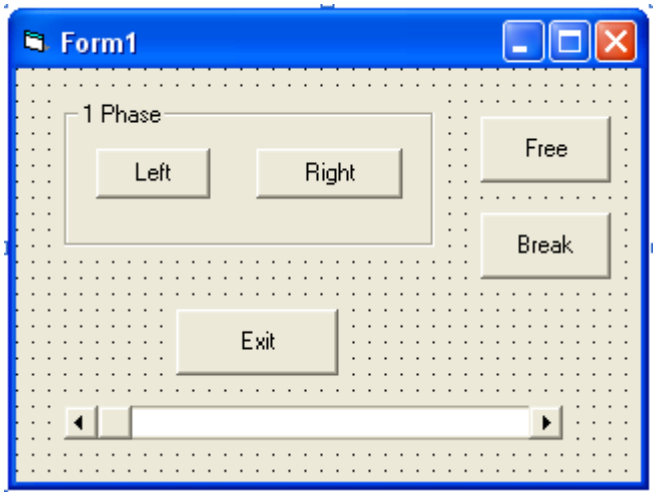
Pas	Faza 4	Faza 3	Faza 2	Faza 1
1	1	0	0	1
2	0	0	0	1
3	0	0	1	1
4	0	0	1	0
5	0	1	1	0
6	0	1	0	0
7	1	1	0	0
8	1	0	0	0

Rotire la stânga

Pas	Faza 4	Faza 3	Faza 2	Faza 1
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

Rotire la dreapta

1. Se porneşte calculatorul şi se lansează aplicația Visual BASIC.
2. Realizați o formă şi editați controalele din figura următoare;



3. Scrieți codul sursă pentru butonul de comandă **Command1 (Left):**

```
Private Sub Command1_Click():
        Left = True
        Rights = False
        Do
                DoEvents
                Out &H378, 9
                        Call delay
                Out &H378, 1
                        Call delay
                Out &H378, 3
                        Call delay
                Out &H378, 2
                        Call delay
                Out &H378, 6
                        Call delay
                Out &H378, 4
                        Call delay
                Out &H378, 12
                        Call delay
                Out &H378, 8
                        Call delay
                Out &H378, 0
                        Call delay

        Loop Until Rights = True
End Sub
```

4. Codul sursă pentru butonul de comandă **Command2 (Rights):**

```
Private Sub Command2_Click():
        Left = False
        Rights = True
        Do
                DoEvents
                Out &H378, 8
                        Call delay
                Out &H378, 12
                        Call delay
                Out &H378, 4
                        Call delay
                Out &H378, 6
                        Call delay
                Out &H378, 2
                        Call delay
                Out &H378, 3
                        Call delay
                Out &H378, 1
                        Call delay
                Out &H378, 9
                        Call delay
                Out &H378, 0
                        Call delay

        Loop Until Lefts = True
End Sub
```

5. Conectați placa P-Board cu placa EX-05 cu ajutorul cablului IDC-10 prin intermediul portului DATA BUS. Conectați motorul pas cu pas la placa EX-05 şi verificați că fazele sunt conectate corect.
6. Alimentați plăcile EX-05 şi P-Board .
7. Rulați programul creat.
8. Observați rotirea motorului şi faceți comparația cu experimentul 1 şi 2.

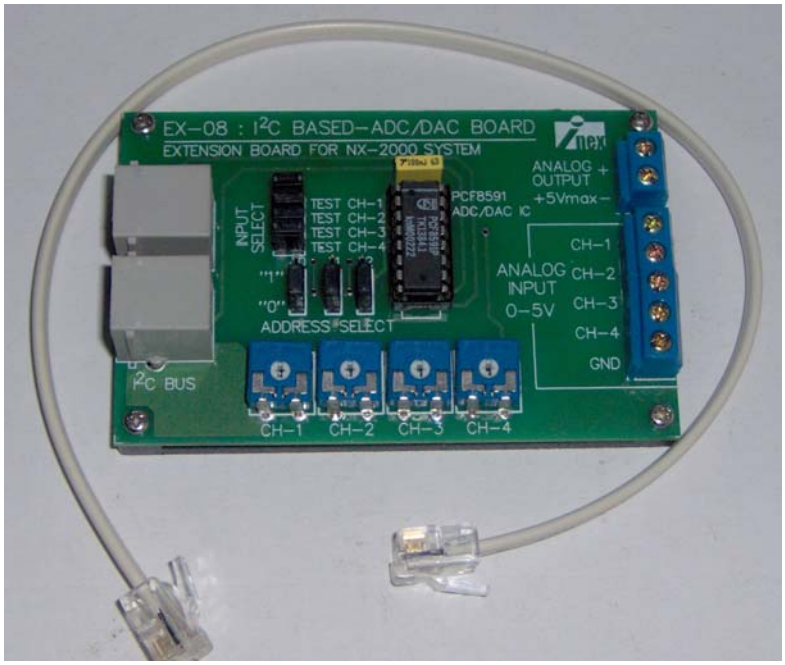
Motorul se va roti mai încet decât la experimentele 1 şi 2 dar rotația permite o rezoluție mai bună, un pas mai mic al motorului.

Laboratorul nr. 5

Studiul convertorului analog numeric

În cadrul acestui laborator se va experimenta conversia unui semnal analogic]n semnal digital şi invers cu ajutorul calculatorului utilizând placa P-board de interfață cu portul paralel al calculatorului şi placa EX-08 ce conține circuitul PCF8591 convertor AD/DA.

Placa EX-08 este prezentată în figura următoare:



Circuitul principal al plăcii PCF8591 permite achiziție de date pe 4 canale analogice, are o ieşire analogică şi comunică cu alte componente prin interfața I2C. Cei trei pini de adresă A0 – A2 sunt folosiți pentru programarea hardware a circuitului privind comunicarea cu maximum 8 dispozitive. Adresele, semnalul de control şi semnalul de date spre dispozitive sau de la ele sunt transferate serial prin două linii de date bidirecționale I2C.

Principalele funcții ale plăcii sunt următoarele: multiplexarea semnalului analog de la intrare, conversie pe 8 biți analog – digital şi digital – analog. Rata maximă de conversie este strâns legată de viteza maximă a magistralei I2C.

Semnalul de intrare poate fi selectat din 2 surse. Una din exterior cuplată la conectorul de intrare iar a doua sursă internă reprezentată prin rezistoarele de pe placă. Selecția surselor se efectuează prin jamperii JP704-JP707 de pe placă.

Interfață analogică cu portul paralel al calculatorului prin intermediul magistralei I2C

Pentru realizarea experimentului se va folosi placa P-Board, placa EX-08 , un calculator care să aibă instalat programul Visual BASIC V5.0 sau mai nouă, o sursă de tensiune continuă 0-5 V cu patru ieşiri , un multimetru digital şi cablul de conexiune I2C.

Procedura de execuție a experimentului 1

Experimentul presupune citirea unui semnal analogic continuu cu ajutorul plăcii EX-08. Paşii urmați de control al circuitului PCF8591 sunt următorii:

1. Trimiterea condiției de START
2. Trimiterea adresei - 000 (A0 – A2 se conectează la masă) şi definirea modului de scriere (bitul LSB clear 0 R/W = 0)
3. Aşteptarea condiției ACK de la PCF8591
4. Trimiterea datelor de control spre PCF8591, 45H – adică activează ieşirea analogă. Se setează intrarea analogă pe modul single, citire continuă şi începere de citire a semnalului de la ADC pe canalul 1
5. Aşteptarea condiției ACK de la PCF8591
6. Trimiterea condiției de STOP
7. Trimiterea condiției de START
8. Trimiterea adresei - 000 (A0 – A2 se conectează la masă) şi definirea modului de scriere (bitul LSB clear 1 R/W = 1) pentru începerea citirii datelor de la intrarea analogă
9. Aşteptarea condiției ACK de la PCF8591
10. Citeşte intrarea ACD canalul 1
11. Trimite condiția MAck (Master Ack) la PCF8591
12. Citeşte intrarea ACD canalul 2
13. Trimite condiția MAck (Master Ack) la PCF8591
14. Citeşte intrarea ACD canalul 3
15. Trimite condiția MAck (Master Ack) la PCF8591
16. Citeşte intrarea ACD canalul 4
17. Trimite condiția MAck (Master Ack) la PCF8591
18. Trimite condiția de STOP

Toți aceşti paşi pot fi convertiți în cod VISUALBasic după cum urmează:

```
'Read 4 analog input in continuous

Private Sub Timer1_Timer()
```

```
        Call I2CStart
        Call Send8BIT(&H90)
        Call Ack
        Call Send8BIT(&H45)
        Call Ack
        Call I2CStop
        Call I2CStart
        Call Send8BIT(&H91)
        Call Ack
        Text1.Text = (DAT * 5)/ 255
        Call MAck
        Text2.Text = (DAT * 5)/ 255
        Call MAck
        Text3.Text = (DAT * 5)/ 255
        Call MAck
        Text4.Text = (DAT * 5)/ 255
        Call Ack
        Call I2CStop
End Sub
```

Subrutina MAck este utilizată pentru trimiterea condiției de confirmare de la calculator la circuitul PCF8591. Codul ei este prezentat mai jos:

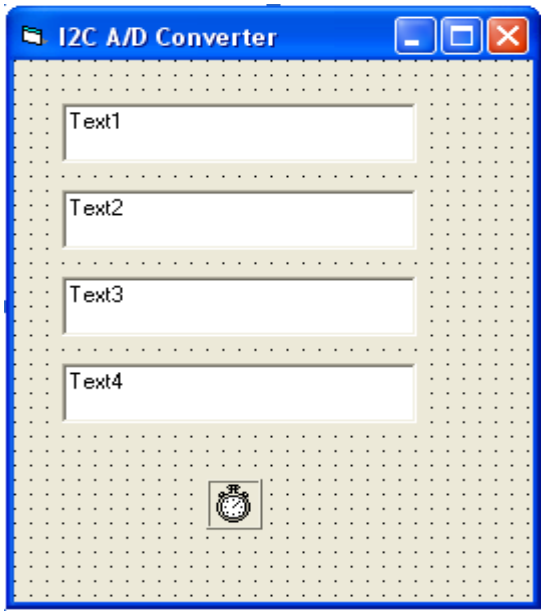
```
        'Mack subroutine
Private Sub Mack()
        Out &H37A, Inp(&H37A) And &HFE 'SDA=0
        Out &H37A, Inp(&H37A) Or 2 'SDL=1
        Out &H37A, Inp(&H37A) And &HFD 'SDL=0
        Out &H37A, Inp(&H37A) Or 1 'SDA=1
End Sub
```

Subrutinele I2CStart, I2CStop Ack şi Send8BIT sunt prezentate mai jos:

```
Private Sub I2CStart()
        Out &H37A, Inp(&H37A) Or 1 'SDA=1
        Out &H37A, Inp(&H37A) Or 2 'SCL=1
        Out &H37A, Inp(&H37A) And &HFE 'SDA=0
        Out &H37A, Inp(&H37A) And &HFD 'SCL=0
End Sub
Private Sub I2CStop()
        Out &H37A, Inp(&H37A) And &HFE 'SDA=0
        Out &H37A, Inp(&H37A) Or 2 'SCL=1
        Out &H37A, Inp(&H37A) Or 1 'SDA=1
End Sub
Private Sub Ack()
        Out &H37A, Inp(&H37A) Or 1 'SDA=1
        Out &H37A, Inp(&H37A) Or 2 'SCL=1
```

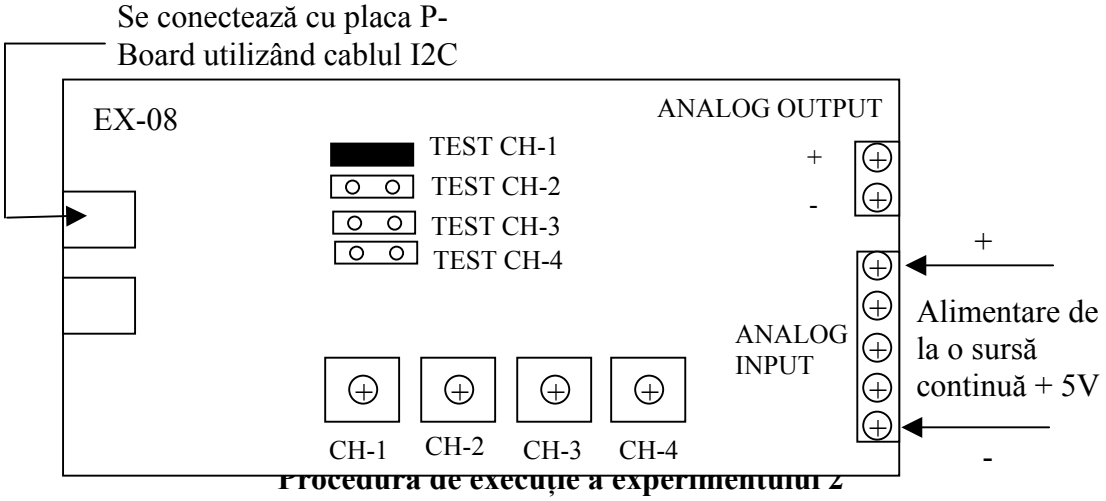
```
    Out &H37A, Inp(&H37A) And &HFD 'SCL=0
End Sub
```

În cod se setează 4 căsuțe de text care vor afișa datele de pe cele 4 canale analoge ale ADC. Valorile reprezintă tensiunile citite de la intrările circuitului PCF8591. Imaginea formei create în VISUALBasic este următoarea:



Modul de desfăşurare practică a experimentului

1. Se setează adresa 000 la circuitul PCF8591 cu ajutorul jamperilor de pe placă
2. Se conectează placa P-Board cu placa EX-08 prin intermediul cablului I2C
3. Se conectează placa P-Board la portul paralel al calculatorului şi se alimentează
4. Se porneşte calculatorul se lansează aplicația Visual BASIC se creează forma precedentă şi se scrie programul.
5. Se deschide fişierul Lab12A.vbp
6. Se setează jamperii pentru selectarea modului TEST CH-1 intrare analogică (se selectează sursa analogă de pe placă, vezi figura de mai jos)
7. Se ajustează rezistorul variabil CH-1
8. Se observă valoarea tensiunii ce se modifică în căsuța Text1
9. Testați şi celelalte intrări CH-2 – CH-4. Se urmăresc paşii 7 şi 8 precedenți observând modificările valorilor din căsuțele de text: Text2 pentru CH-2, Text3 pentru CH-3, Text4 pentru CH-4
10. Se setează jamperii pentru test la toate canalele
11. Se ajustează rezistoarele pentru fiecare canal în parte şi se observă valorile din cele 4 căsuțe de text.



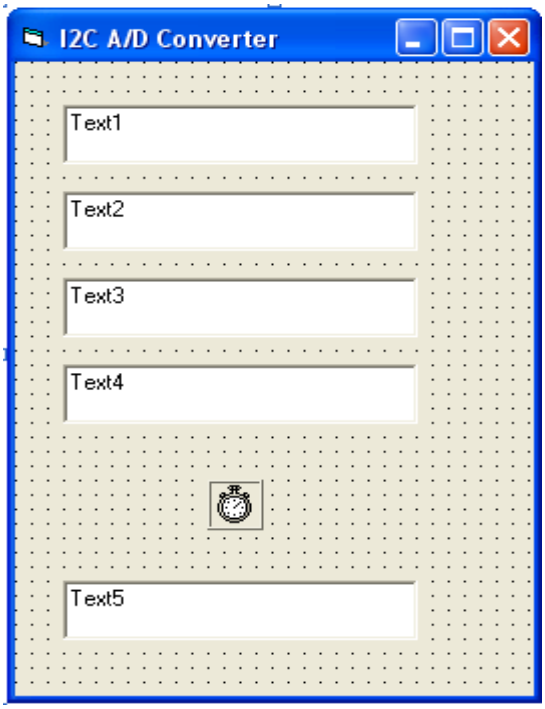
Procedura de execuție a experimentului 2

Scrierea de date pe 8 biți în modulul DAC a circuitului PCF8591. Paşi ce trebuie urmați pentru desfăşurarea experimentului sunt următorii:

1. Trimiterea condiției de START
2. Trimiterea adresei - 000 (A0 – A2 se conectează la masă) şi definirea modului de scriere (bitul LSB clear 0 R/W = 0)
3. Aşteptarea condiției ACK de la PCF8591
4. Trimiterea datelor de control spre PCF8591, 44H – adică activează ieşirea analogă.
5. Aşteptarea condiției ACK de la PCF8591
6. Trimitere de date în intervalul 0-255 la ieşirea analogă
7. Aşteptarea condiției ACK de la PCF8591
8. Trimite condiția de STOP

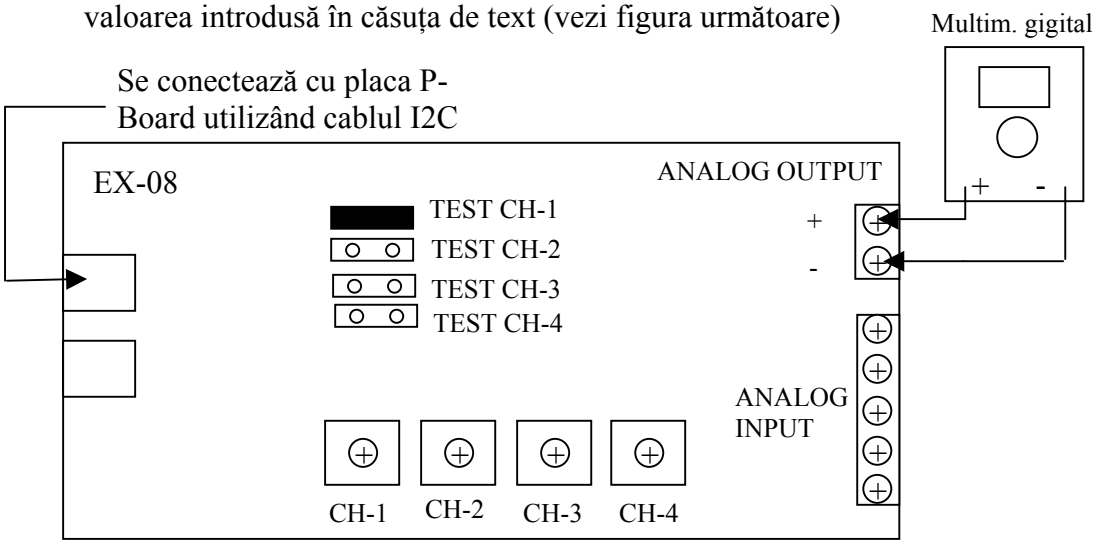
Toți aceşti paşi pot fi convertiți în cod VISUALBasic după cum urmează:

```
Private Sub Text5_Change()
        If Val(Text5.Text) > 5 Then Text5.Text = 5
        Call I2CStart
        Call Send8BIT(&H90)
        Call Ack
        Call Send8BIT(&H44)
        Call Ack
        Call Send8BIT(Val(Text5.Text) * 51.2)
        Call Ack
        Call I2CStop
End Sub
```

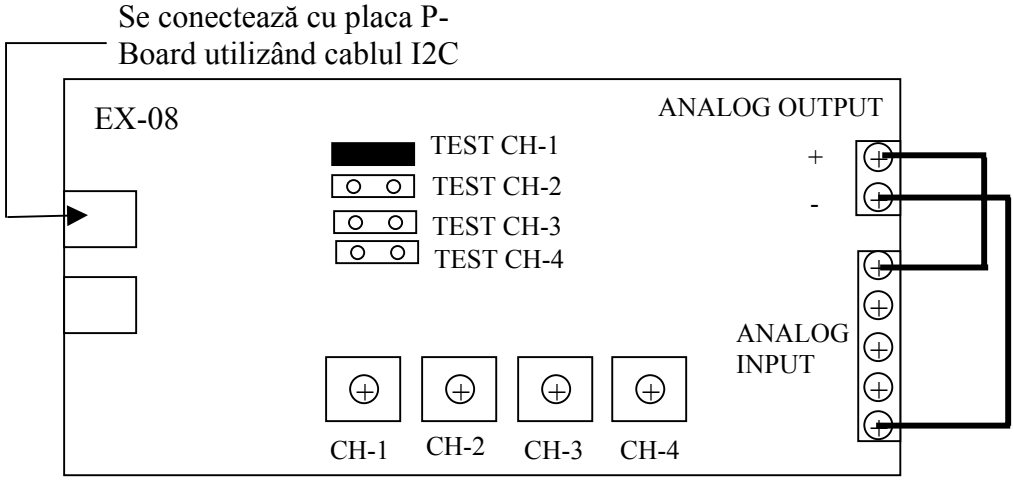


Această rutină utilizează căsuța de text Text5 pentru definirea datei ce va trimisă circuitului PC8591.

9. Se setează adresa 000 la circuitul PCF8591 cu ajutorul jamperilor de pe placă
10. Se conectează placa P-Board cu placa EX-08 prin intermediul cablului I2C
11. Se conectează placa P-Board la portul paralel al calculatorului şi se alimentează
12. Se porneşte calculatorul se lansează aplicația Visual BASIC
13. Se deschide fişierul Lab12A.vbp
14. Introduceți o valoarea a tensiunii în căsuța Text5 (valoarea maximă 5)
15. Utilizați multimetrul digital, setat pentru măsurarea unei tensiuni continue, pentru măsurarea tensiunii la ieşirea analogă a plăcii EX-08 şi comparație cu valoarea introdusă în căsuța de text (vezi figura următoare)



16. Apoi conectați ieşirea analogă la intrarea CH-1 a plăcii EX-08 ca în figura următoare
17. Introduceți o valoarea a tensiunii în căsuța Text5 (valoarea maximă 5)
18. Observați modificarea valorii din interiorul căsuței de text Text1. Este în concordanță cu valoarea din Text5?



La paşii 14 şi 15 se testează performanța circuitului PCF8591 privind conversia semnalului. Valoarea introdusă în căsuța de text Text5 este convertită în semnal digital şi apoi trimisă circuitului ce realizează conversia semnalului digital în semnal analogic. Semnalul analogic este trimis la ieşirea analogică a plăcii EX-08.

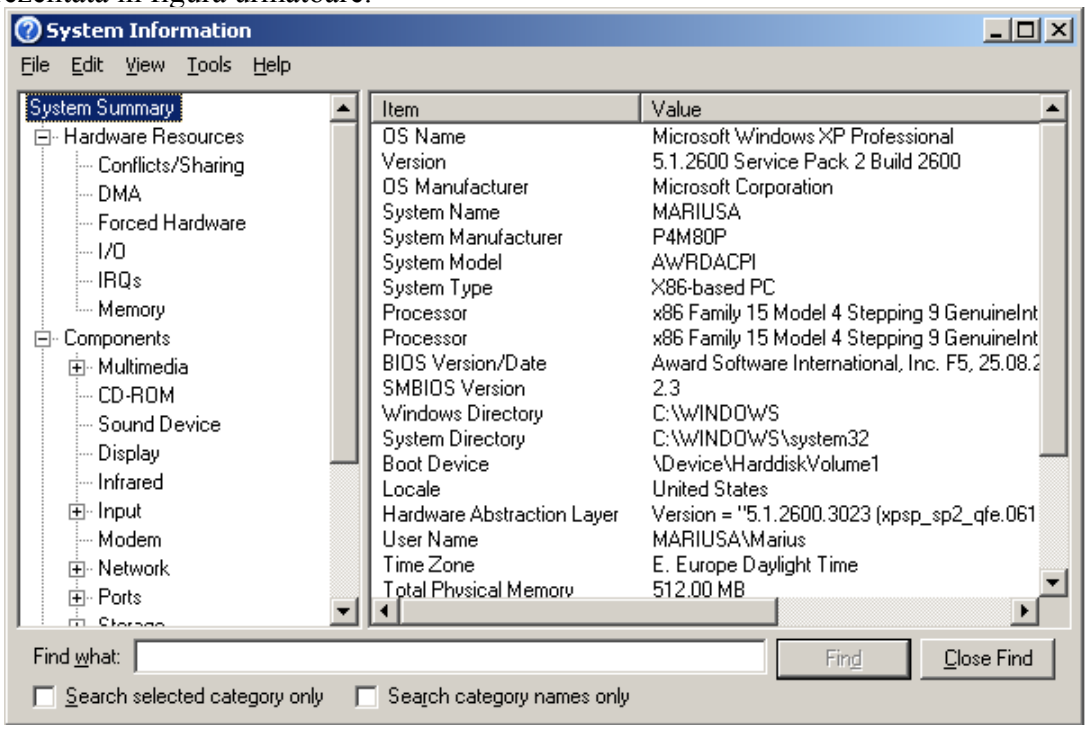
La paşii 17 şi 18 se testează funcția ADC şi apoi DAC a circuitului PCF8591. Programul va converti valoarea introdusă în căsuța de text Text5 în semnal digital şi o trimite modului DAC a circuitului ce realizează conversia în semnal analogic. După aceea această tensiune este trimisă înapoi pe intrarea analogă modulului ADC ce realizează conversia în semnal digital vizualizat apoi în căsuța de text Text1.

Laboratorul nr. 6

Programe pentru determinarea structurii şi a performanţelor sistemului de calcul

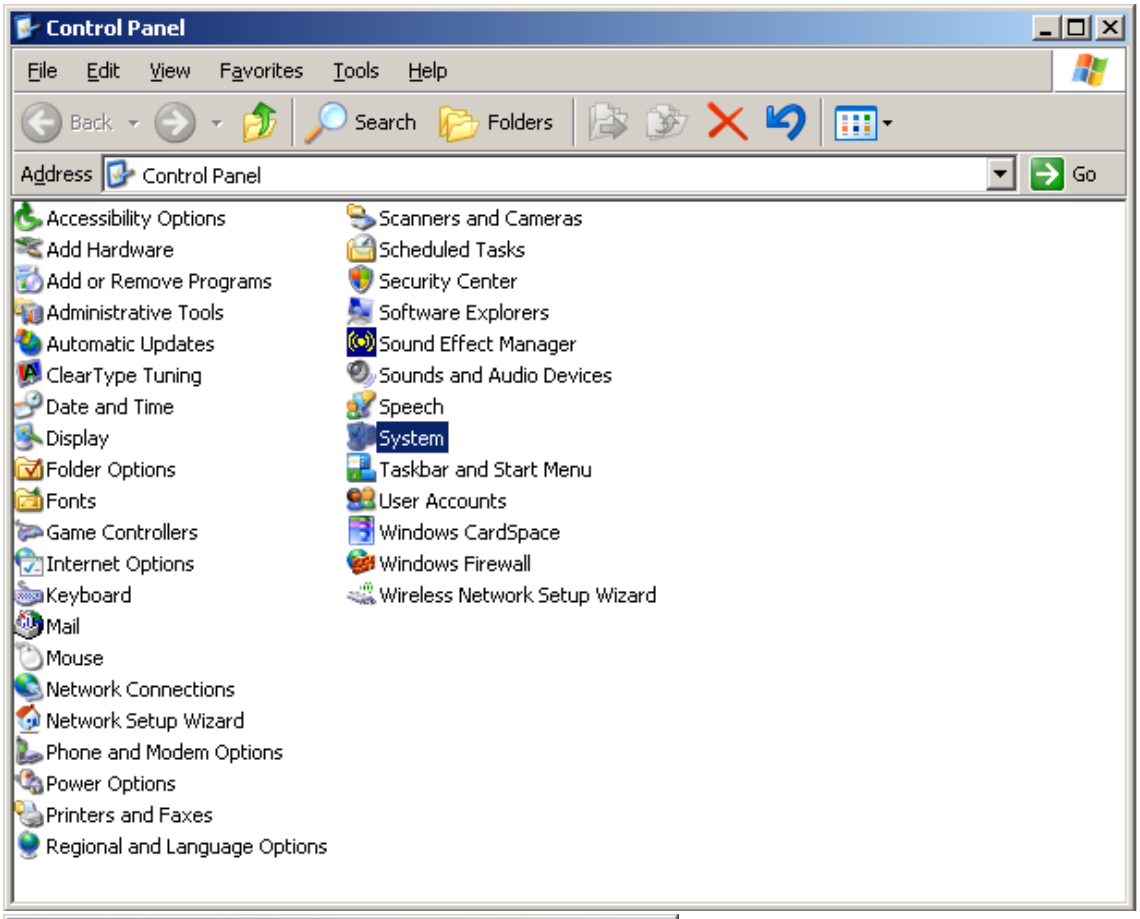
În cadrul acestui laborator vor fi prezentate mai multe aplicaţii utile, necesare pentru determinarea structurii hardware a unui sistem de calcul precum şi a performanţelor componentelor în parte sau a sistemului pe ansamblu.

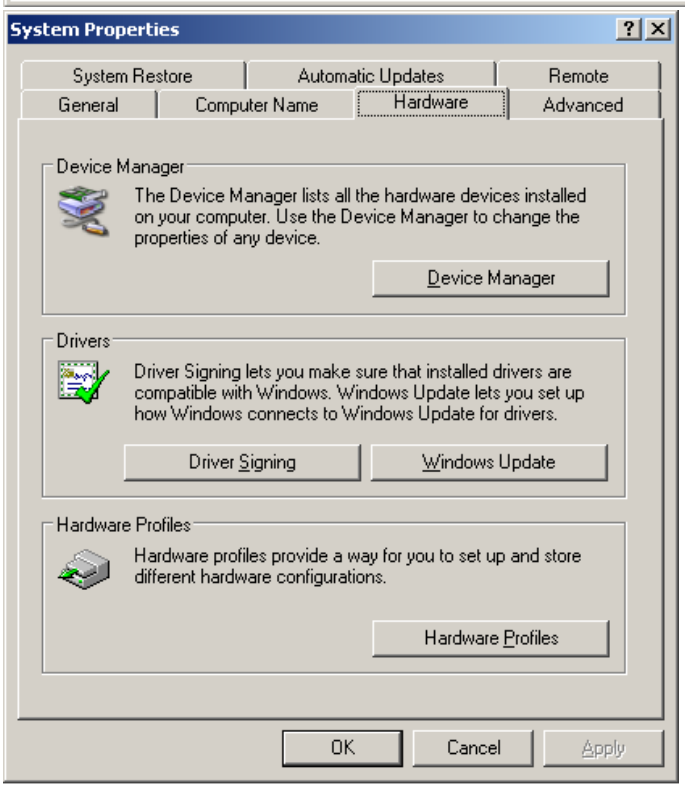
Prima aplicaţie care poate fi folosită pentru determinarea structurii hardware a unui sistem de calcul, este o componentă a sistemului de operare folosit, în cazul de faţă Microsoft Windows XP, denumită System Information. Lansarea în execuţie se face normal ca şi orice aplicaţie a sistemului urmând calea Start – Programs – Accessories – System Tools - System Information şi dând click .Interfaţa acestei aplicaţii este prezentată în figura următoare.



Aplicaţia la deschidere prezintă informaţii generale despre sistemul folosit prezentate în panoul din dreapta al ferestrei, dar pentru informaţii mai detaliate privind componentele sistemului se poate interoga aplicaţia, utilizând structura arborescentă din panoul din stânga al ferestrei, dând click pe aceea componetă.

Din punct de vedere numai al perifericelor care intră în componenţa sistemului şi cel mai important al controlului lor (instalare, dezinstalare, verificare funcţionare) se poate folosi aplicaţia System a sistemului de operare care poate fi lansată în mai multe moduri, de exemplu din fereastra Control Panel.

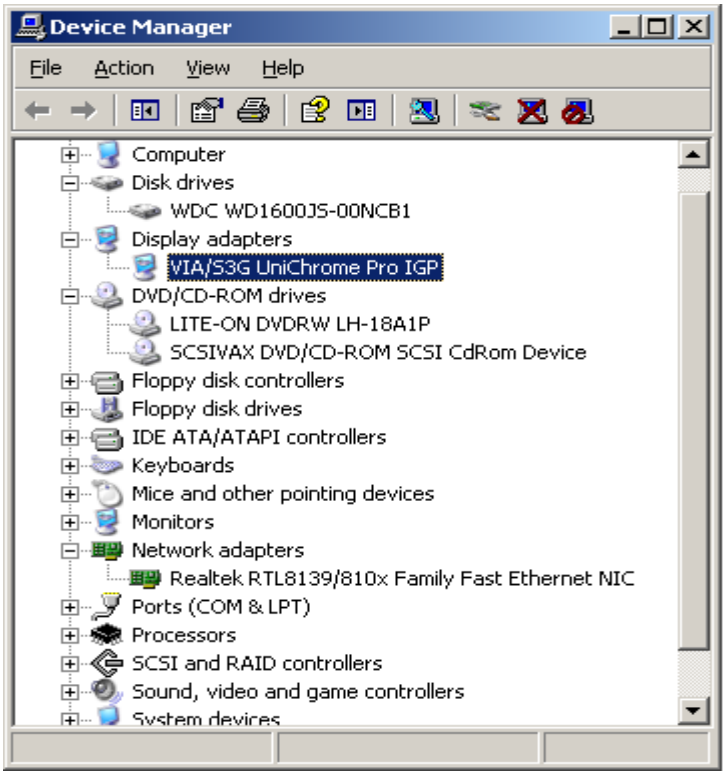




Se va da click pe butonul Device Manager din panoul Hardware a aplicației şi se va deschide o nouă fereastră.

Printr-o structură arborescentă, ne sunt prezentate perifericele din componența sistemului. Starea fiecărui periferic poate fi aflată interogând aplicația prin comanda Properties.

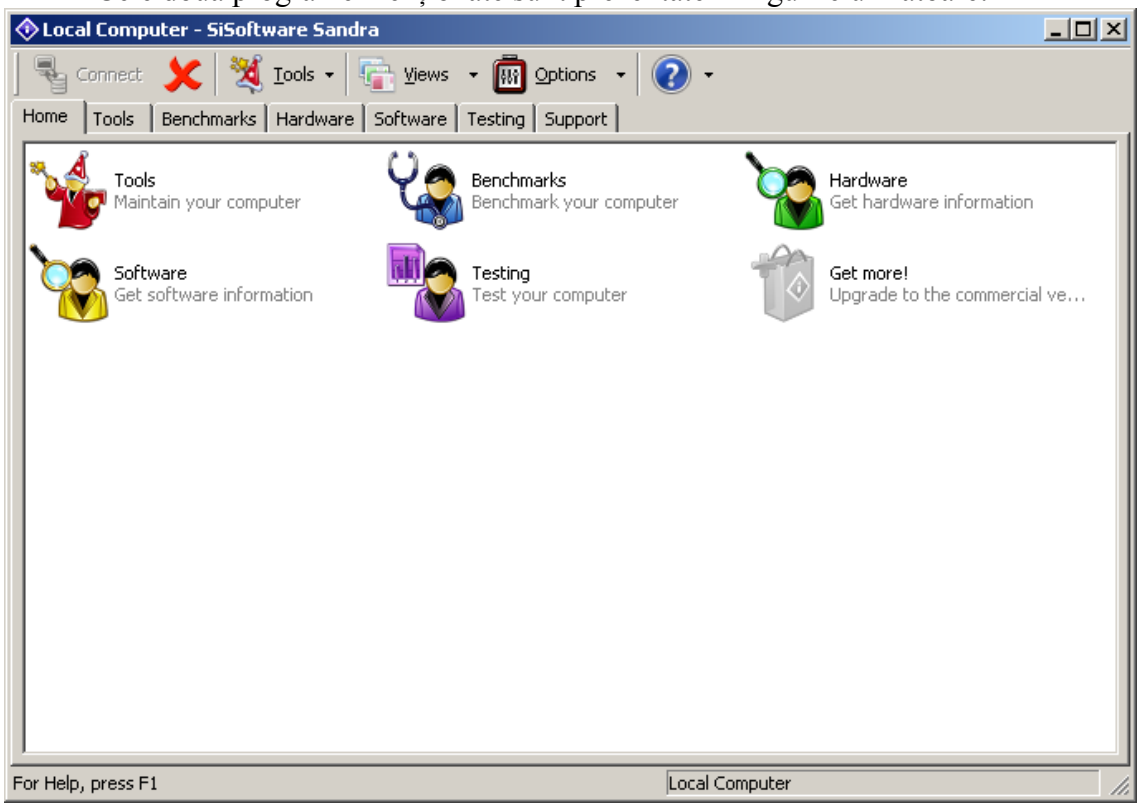
Dacă o componentă nu funcționează bine sau sistemul nu a recunoscut-o şi din punct de vedere al driverelor necesare, aplicația semnalizează grafic acest lucru afişând semnul exclamării într-un cerc galben în dreptul componentei.

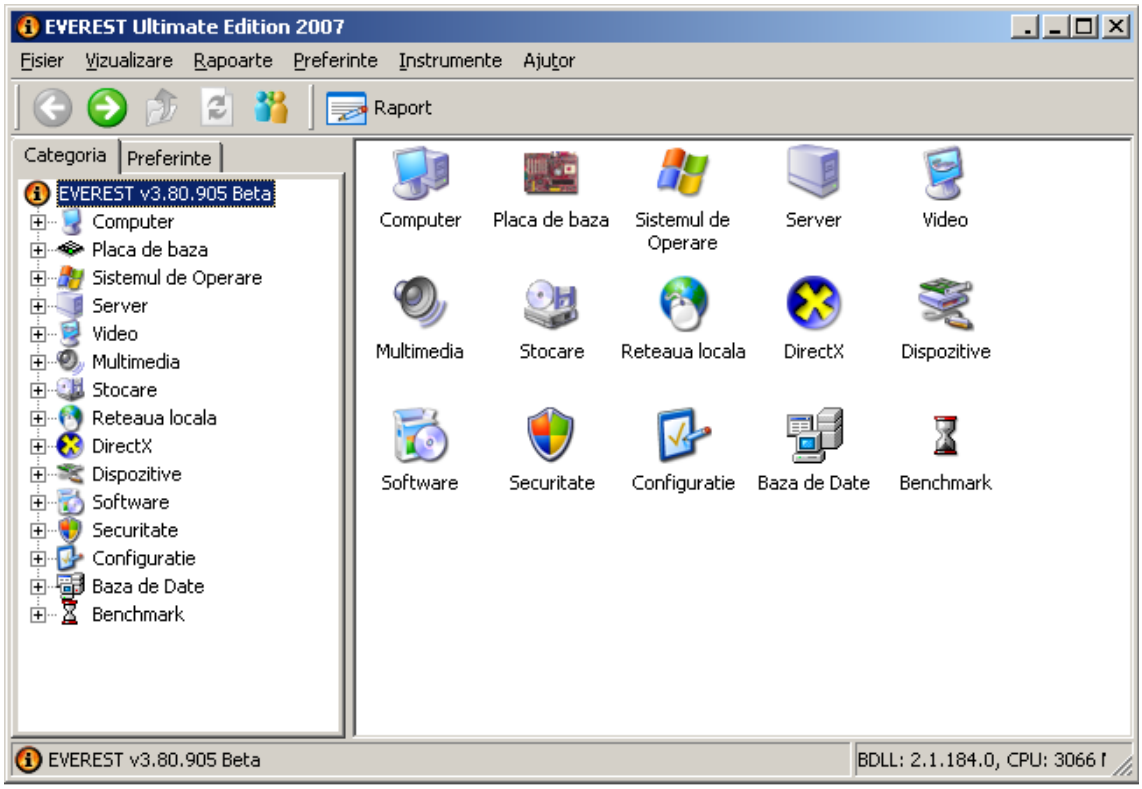


Sistemul de operare Microsoft Windows XP, are în componență și alte aplicații utile privind afișarea componenței hardware a sistemului și mai ales a testării performanțelor componentelor de exemplu: Computer Management, Performance, Services etc.

Tot în scopul determinării structurii și a performanțelor unui sistem de calcul pot fi utilizate următoarele programe: EVEREST și SiSoftware Sandra. Aceste două programe au un avantaj față de aplicațiile sistemului de operare deoarece ele integrează atât partea de afișare a structurii sistemului de calcul precum și de testare a performanțelor componentelor sistemului.

Cele două programe menționate sunt prezentate în figurile următoare:





În cadrul acestui laborator studenții trebuie să utilizeze aplicațiile prezentate şi să verifice structura sistemului utilizat, informațiile obținute fiind trecute în referatul de laborator. În plus, se va lansa aplicația Benchmark de verificare a performanțelor sistemului de calcul a fiecărui soft în parte şi se va face o comparație între rezultatele obținute.

Laboratorul nr. 7

Metode de testare a memoriei

În cadrul acestui laborator vor fi prezentate şi folosite mai multe programe care permit testarea integrității memoriei interne a sistemului de calcul precum şi performanțele acesteia.

Pentru testarea integrității unităților de memorie internă pot fi folosite mai multe softuri de exemplu Memtest sau Mem86 ambele fiind gratuite. Pentru ca rezultatul testării să fie cât mai corect ambele softuri necesită ca ele să fie date în execuție fără ca sistemul de operare al calculatorului să intre în execuție (să nu booteze). Pentru aceasta programele trebuie să fie memorate pe o unitate de memorie externă, CD sau dischetă bootabilă. Pentru a realiza o dischetă bootabilă se dă comanda de formatare a dischetei, în MSDOS Prompt , cu opțiunea de copiere a fişierelor sistem minimale IO.SYS, MSDOS.SYS şi COMMAND.COM

Ex: **Format a: /S**

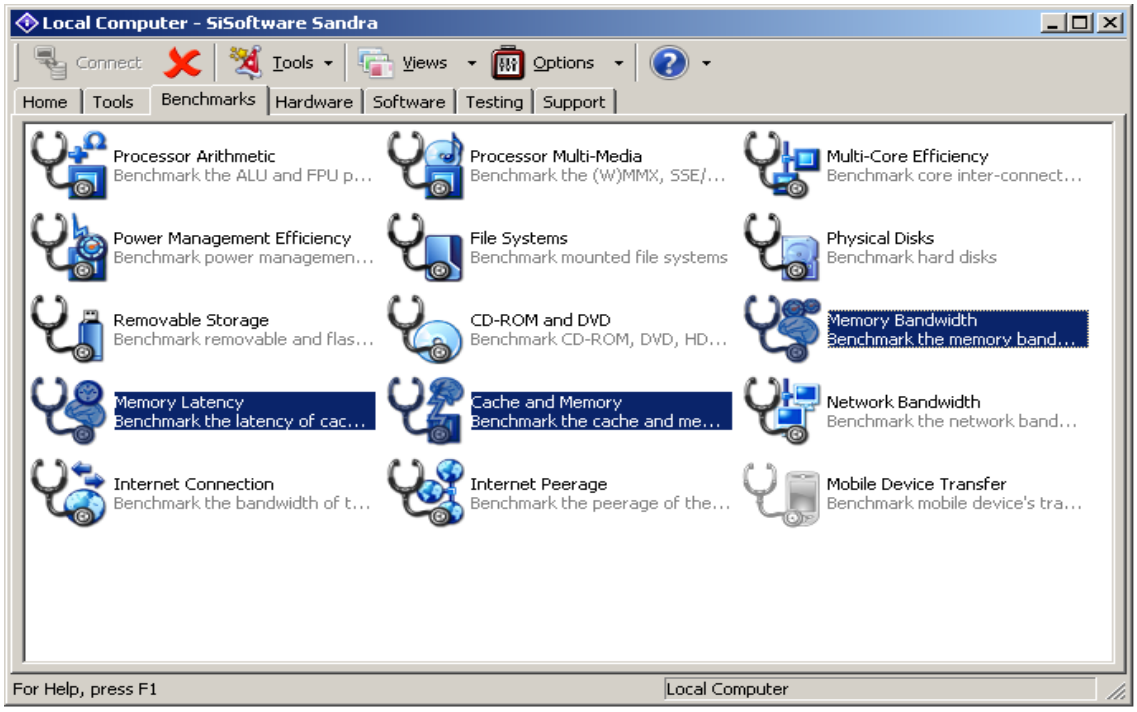
sau utilizând aceeaşi comandă sub sistemul de operare MS XP bifând în fereastră opțiunea de includere a fişierelor sistem după operația de formatare a dischetei.

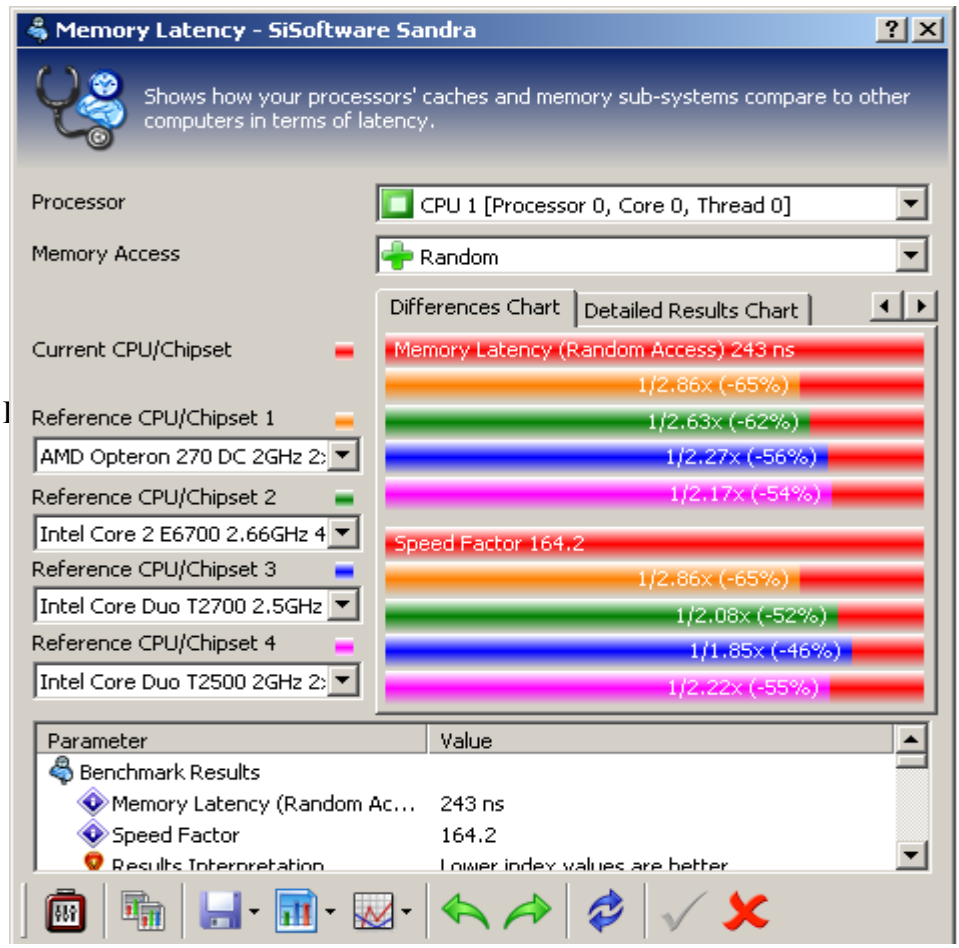
Pentru a putea rula aplicația de pe dischetă este nevoie să se modifice în fereastra de setări a BIOS-ului, opțiunea de căutare prima dată a sistemului de operare pe unitatea de dischetă, astfel încât la o repornire a calculatorului, acesta să încarce sistemul de pe ea şi apoi să se lanseze aplicația respectivă de verificare a integrității memoriei interne.

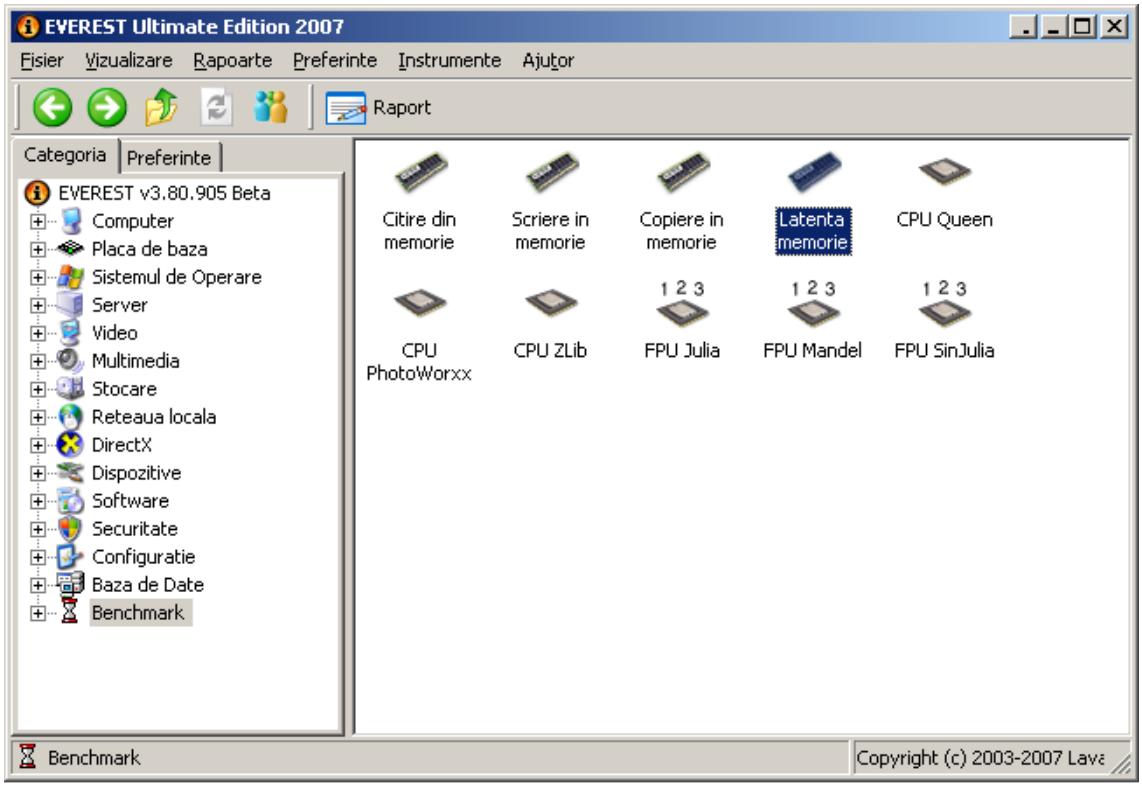
Deoarece pentru execuția acestei aplicații se folosesc mai mulți algoritmi, durata testului poate dura între zeci de minute şi ore, fiind influențată şi de caracteristicile hardware ale sistemului de calcul. La terminare testării aplicația comunică utilizatorului dacă memoria RAM a sistemului este în stare bună de funcționare sau dacă au apărut erori în funcționarea acesteia.

Din punct de vedere al performanțelor memorie interne a sistemului pot fi utilizate cu succes şi programele prezentate în lucrarea de laborator precedentă, EVEREST şi SiSoftware Sandra. Ambele programe includ şi aplicațiile Benchmark ce permit testarea memorie interne.

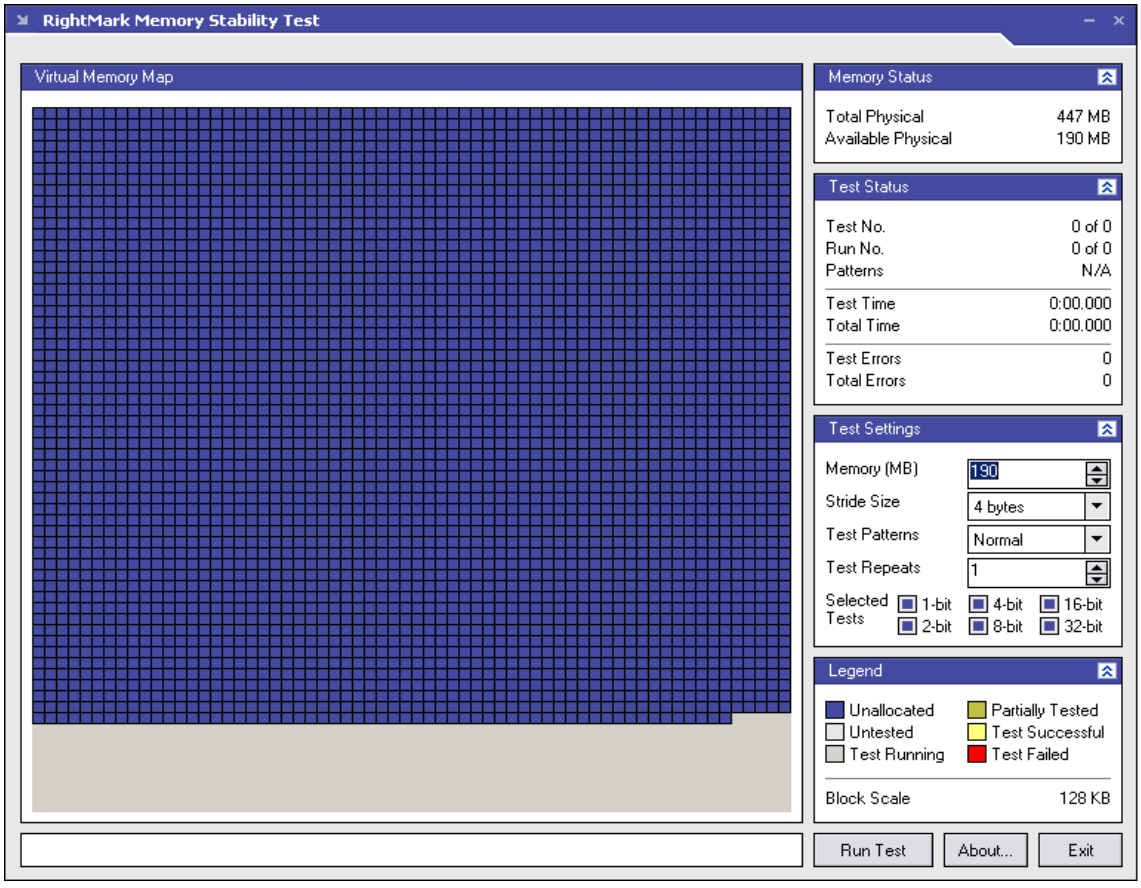
Ferestrele de lansare a acestei aplicații pentru ambele programe precum şi un exemplu de rulare a unui test pentru verificarea latenței memoriei sunt prezentate în figurile următoare.

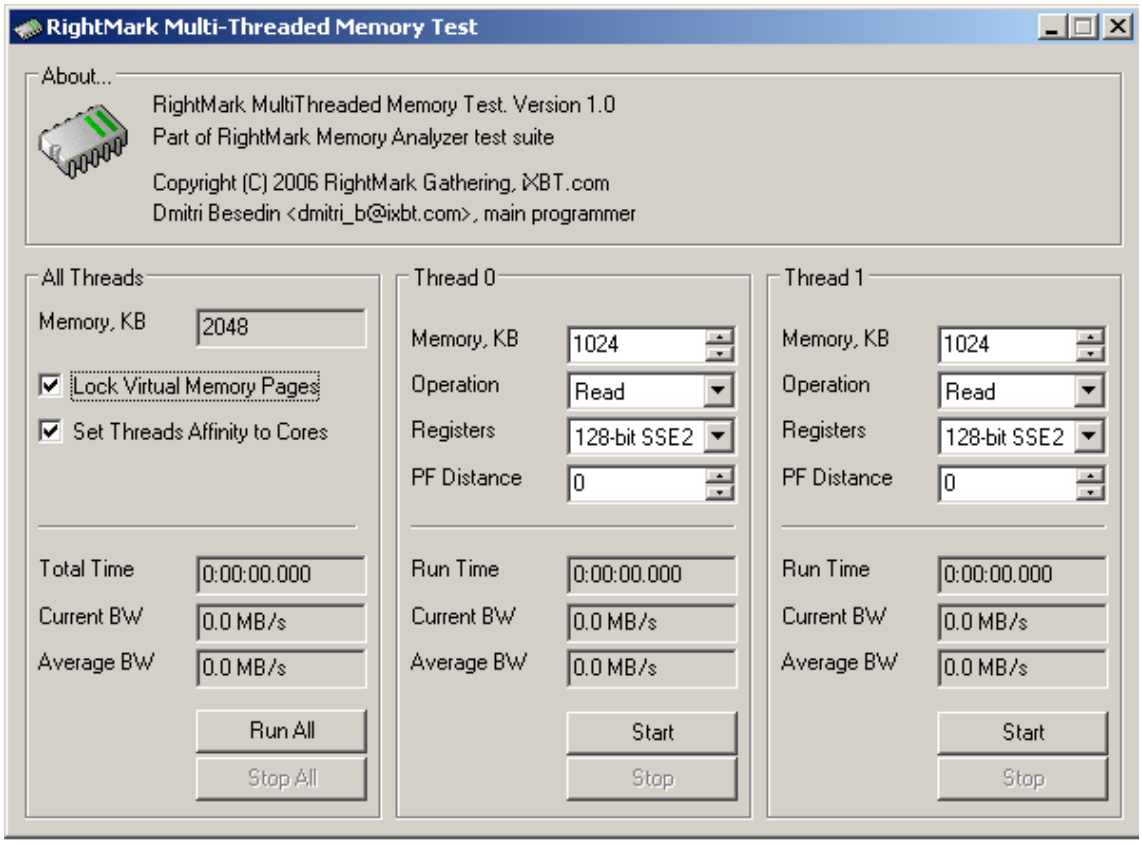

Local Computer - SiSoftware Sandra
Connect
Tools
Views
Options
Home
Tools
Benchmarks
Hardware
Software
Testing
Support
Processor Arithmetic
Benchmark the ALU and FPU p...
Processor Multi-Media
Benchmark the (W)MMX, SSE/...
Multi-Core Efficiency
Benchmark core inter-connect...
Power Management Efficiency
Benchmark power managemen...
File Systems
Benchmark mounted file systems
Physical Disks
Benchmark hard disks
Removable Storage
Benchmark removable and flas...
CD-ROM and DVD
Benchmark CD-ROM, DVD, HD...
Memory Bandwidth
Benchmark the memory band...
Memory Latency
Benchmark the latency of cac...
Cache and Memory
Benchmark the cache and me...
Network Bandwidth
Benchmark the network band...
Internet Connection
Benchmark the bandwidth of t...
Internet Peerage
Benchmark the peerage of the...
Mobile Device Transfer
Benchmark mobile device's tra...
For Help, press F1
Local Computer


Memory Latency - SiSoftware Sandra
Shows how your processors' caches and memory sub-systems compare to other computers in terms of latency.
Processor
CPU 1 [Processor 0, Core 0, Thread 0]
Memory Access
Random
Differences Chart
Detailed Results Chart
Current CPU/Chipset
Memory Latency (Random Access) 243 ns
Reference CPU/Chipset 1
AMD Opteron 270 DC 2GHz 2x
1/2.86x (-65%)
1/2.63x (-62%)
1/2.27x (-56%)
1/2.17x (-54%)
Reference CPU/Chipset 2
Intel Core 2 E6700 2.66GHz 4
Speed Factor 164.2
Reference CPU/Chipset 3
Intel Core Duo T2700 2.5GHz
1/2.86x (-65%)
1/2.08x (-52%)
1/1.85x (-46%)
Reference CPU/Chipset 4
Intel Core Duo T2500 2GHz 2x
1/2.22x (-55%)
Parameter
Value
Benchmark Results
Memory Latency (Random Ac...
243 ns
Speed Factor
164.2
Results Interpretation
Lower index values are better



Un alt program ce poate fi utilizat pentru testarea memorie interne este RightMark Memory Analyzer, special realizat pentru acest lucru. În figurile următoare fiind prezentate ferestrele de execuție a aplicațiilor de testare:





În timpul orei de laborator studenții vor utiliza programele prezentate pentru testarea memorie interne, şi vor consemna în referatul de laborator rezultatele obținute, privind performanțele memoriei sistemului de calcul, raportate de fiecare program utilizat.

Bibliografie

1. Athanasiu Irina, Panoiu Alexandru, - Microprocesoarele 8086, 286, 386, Editura TEORA, Bucuresti, 1992;
2. Adrian Petrescu ş.a. – Microcalculatoarele M18, M18B, M118, Editura Tehnică, Bucureşti, 1984;
3. Andronescu Gh., - Sisteme Digitale, Editura MatrixRom, Bucuresti, 2002;
4. Baluta Gheorghe, - Circuite logice si structuri numerice. Proiectare si aplicatii. Editura MatrixRom, Bucuresti, 2002;
5. Blakeslee Thomas, Proiectarea cu circuite logice MSI si LSI standard, EdituraTehnica, Bucuresti, 1988;
6. Bogdanov Ivan, - Microprocesorul in comanda actionarilor electrice, Editura FACLA, Timisoara, 1989;
7. Budiu Mihai, Cache-uri, Cornell University, USA, 1999;
8. Capatina Octavian, - Proiectarea cu microcalculatoare integrate, Editura Dacia, Cluj, 1992;
9. Cristian Lupu, Stefan Stancescu, - Microprocesoare Circuite Proiectare., Editura Militara, Bucuresti, 1986;
10. Cuculescu I., - Analiza numerica, Editura Tehnica, Bucuresti, 1967;
11. Dancea Ioan, - Microprocesoare. Arhitectura interna, programare, aplicatii. Editura Dacia, Cluj-Napoca, 1979;
12. Davidoviciu A., s.a., - Minicalculatoarele si microcalculatoarele in conducerea proceselor industriale, Editura Tehnica, Bucuresti, 1983;
13. Ionescu D. - Codificare si coduri, Editura Tehnica, Bucuresti, 1981;
14. Lupu C., s.a. - Microprocesoare. Aplicatii. Editura Militara, Bucuresti 1982;
15. Marinescu D. Naicu S., - Microcontrolerul 80C32. Manual de utilizare. Editura Tehnica, Bucuresti, 1998;
16. Milici Dan, - Circuite numerice. Introducere in sistemele de calcul, Editura MatrixRom, Bucuresti, 2003;
17. Pop Eugen, s.a. - Metode in prelucrarea numerica a semnalelor, Editura FACLA, Timisoara, 1989;
18. Popescu D., Popescu C., - Circuite digitale elementare, Editura MatrixRom, Bucuresti, 2003;
19. Potorac D.A., - Bazele proiectarii circuitelor numerice, Editura MatrixRom, Bucuresti, 2002;

20. Puiu-Berizinţu M., Rotar Dan – An Optimal Control Method of the PWM Inverter used in Electrical Drives with Induction Motor - MIPRO'99 CONFERENCE, IEEE Region 8, CROAŢIA 1999.
21. Puiu Berizinţu Mihai, Rotar Dan – Using DSP for PWM Inverter Command by the Generatrix Wave Sampling Principle, Conferinţa Naţională de Acţionări Electrice "CNAE 2000", Iaşi, 12-14 octombrie 2000, publicată în Buletinul Institutului Politehnic Iaşi, Tomul XLVI (L), Fasc. 5, ISSN 0258-9109, pp. 72-77
22. Radu O., Sandulescu Gh., - Filtre numerice. Aplicatii, Editura Tehnica, Bucuresti, 1979;
23. Rotar Dan - Harmonic analysis based on microcomputers, Efficiency, Cost, Optimization, Simulation and Environmental Aspects of Energy Systems and Processes Congress ECOS98, ISBN 2-905-267-29-1, Nancy, France, pp. 1173-1180, 1998.
24. Rotar Dan - Protection of the Microcomputer-based Pulse-Width Modulated Inverters, 17th International Conference on COMPUTERS IN TECHNICAL SYSTEMS, Proceedings Volume 2, ISBN 953-6042-57-6, pp. 67-70, CROAŢIA 1998.
25. Rotar Dan, Ababei Ştefan - Determinarea consumului energetic prin contorizare numerică, Conferinţa Naţională de Energetică Industrială, Bacău, 1998, Editura Plumb, ISBN 973-9362-16-8, pp. 170-173.
26. Rotar Dan – Sisteme de măsură digitale a energiei electrice – Probleme de management şi conservare a energiei, Craiova, ISBN 973-0-00917-1, pp. 21-28, 1999
27. Rotar Dan – Programarea DSP, Conferinţa Naţională de Energetică Industrială CNEI 2000 MILENIUM, 10-11 noiembrie 2000, Bacău, Editura ALMA MATER, ISBN 973-99703-4-6, pp. 84-87
28. Rotar Dan – Regulator numeric pentru procesorul digital de semnal TMS320F240, Conferinţa Naţională de Energetică Industrială CNEI 2000 MILENIUM, 10-11 noiembrie 2000, Bacău, Editura ALMA MATER, ISBN 973-99703-4-6, pp. 88-91
29. Rotar Dan, Ababei Stefan, Sorin Popa, Communication system for DSP and PC compatible computer, Romanian Academy, Branch office of Iasi, MCOM-8, 2002, ISSN 1224-7480, pp. 413-418.
30. Dan Rotar, Petru Livinţi, Ababei Stefan, Digital filtering with digital signal processing controller, Romanian Academy, Branch office, MCOM-9 vol. 2, 2003, ISSN 1224-7480, pp. 207-210.
31. Somnea Dan, Vladut Teodor, - Programarea in assembler. Editura Tehnica, Bucuresti, 1992;
32. Stanasila Octavian, - Notiuni si tehnici de matematica discreta, Editura Stiintifica si Enciclopedica, Bucuresti, 1985;
33. Stanomir D., Stanasila O., - Metode matematice in teoria semnalelor, Editura Tehnica, Bucuresti 1980;
34. Suciu Marcel, Popescu Dumitru, Ionescu Traian, - Microprocesoare, microcalculatoare si roboti in automatizari industriale, Editura Tehnica, Bucuresti, 1986
35. Sztojanov I., s.a. - De la poarta TTL la microprocesor vol I, II, Editura Tehnica, Bucuresti, 1987;

36. Tanase Ady, Gaitan V., - Familia de procesoare pentru prelucrarea numerica a semnalelor ADSP-21, Editura MatrixRom, Bucuresti, 2004;
37. Teodorescu Dan, - Introducere in microelectronica, Editura Facla, Timisoara, 1985;
38. Teodorescu Dan, - Automatizari microelectronice, EdituraTehnica, Bucuresti, 1988:
39. Toacse Ghe., - Introducere in microprocesoare, Editura Stiintifica si Enciclopedica, Bucuresti, 1986;
40. Toacşe Gheorghe, Nicula Dan – Electronică digitală. Dispozitive. Circuite. Proiectare., Editura Tehnică, Bucuresti, 2005;
41. Zoican Sorin - Arhitectura sistemelor de calcul,Universitatea Politehnica, Bucuresti, 1998;
42. Zoican Sorin, Popovici C. Eduard - Arhitectura microprocesoarelor. Indrumar de laborator, Universitatea Politehnica, Bucuresti, 1997;
43. *** – TMS320C24x DSP Controllers - Reference Set: Vol.1, Texas Instruments Inc, 1997;
44. *** – TMS320C24x DSP Controllers - Reference Set: Vol.2, Texas Instruments Inc, 1997;
45. *** – AT90S3213 Microcontroller, Atmel, 1998;
46. *** – PIC 16F97x Microcontroller, Microchip, 2005;