# DISPOZITIVE CU ACCES LA MEMORIE

## 1. Memento teoretic

Deoarece microprocesorul ar consuma foarte mult timp pentru transferurile de date dintre periferice (datele stocate pe hard disc, flopy disc, într-un proces de achizitie de date etc.) s-a recurs la folosirea de circuite specializate care sa preia acesta sarcina de transfer intre periferic si memorie, cunoscând adresa sursa, adresa destinatie si numarul de octeti de transferat.

## 2. Controler-ul DMA în arhitectura IBM – PC

Firma IBM a folosit controler-ul DMA 8237A fabricat de firma Intel, controler care se foloseste si în ziua de astazi, în structura PC. Se va studia acest circuit în continuare.

Urmasii lui 8237A sunt 8237A-4 si 8237A-5 care este cel mai recent contoler si totodata cel care respecta compatibilitatea cu cele mai vechi. Acestea au urmatoarele caracteristici:

- Posibilitate de E/D pentru fiecare canal a cereri de DMA (DMA Request)
- Patru canale independente / chip
- Posibilitatea de transfer memorie - memorie
- Posibilitate de incrementare sau decrementare a adresei sursa - destinatie
- Posibilitate de transfer cu o viteza de maxim 1.6 MB/s cu 5 MHz 8237A-5
- Posibilitate de de expandare directa la orice numar de canale
- Posibilitate de de cerere soft pentru DMA Request
- Posibilitate de de control independent pentru DREQ sI DACK
- Poate accesa maxim un segment de memorie (64 Ko).

Schema bloc a controlerului DMA 8237A-5 este prezentata în anexa 1.

Descrierea pinilor :

**RESET**: intrare activa în "1" logic si care reseteaza pe 0 registrii de comanda, stare, cereri, temporar si masca. Dupa RESET circuitul este inactiv un ciclu. Tip: I.

**CLK** : linie de ceas cu frecventa de 5MHz pentru 8237A-5 (3MHz pentru 8237A).

**CS** : (Chip Select) este activ in "0" logic si este utilizat pentru selectia circuitului în mod I/O, permitând programarea circuitului în timpul ciclurilor inactive. Tip: I.

**REDY** : este utilizata pentru sincronizarea circuitului cu memoriile si cu dispozitivele periferice lente. Tip: I.

**AEN** : (Adress Enable) iesire folosita pentru selectarea circuitului latch. Cind AEN=1 bitii $A_8$ - $A_{15}$ sunt trecuti pe magistrala de adrese. Se poate folosi pentru dezactivarea altor periferice ce utilizeaza magistrala in mod DMA. Este activ in "1" logic. Tip: O.

**EOP** : ( End Of Process ) este un semnal bidirectional care indica terminarea unui serviciu DMA. Acest semnal se poate activa intern sau extern. Totodata este în strânsa legatura cu TC (Terminal Count ) care indica ca un canal a terminat transferul. Tip: I/O.

**ADSTB** : (Adress Strobe) este activ în "1" logic si este utilizat pentru a selecta bitii $A_8$ - $A_{15}$ într-un latch extern. Tip: O.

**MEMR** : (Memory Read) iesire TS activa în "0" logic folosita pentru a citi din memorie de la adresa selectata într-un ciclu de DMA Read sau transfer memorie - memorie. Tip: O.

**MEMW** : ( Memory Write ) iesire TS activa în "0" logic folosita pentru a scrie în memorie de la adresa selectata într-un ciclu de DMA Write sau transfer memorie - memorie. Tip: O.

**IOR** : (I/O Read) linie bidirectionala TS prin intermediul careia CPU poate citi registrii de control în timpul ciclurilor inactive. În timpul ciclurilor active este folosita de DMA pentru a accesa date de la periferice care utilizeaza DMA Write transfer. Tip: I/O.

**DRQ0-DRQ3** : ( DMA Request ) intrari pe care perifericele transmit asincron semnale cu semnificatia ca este necesar un transfer DMA pentru canalul specificat. Aceste cereri pot fi servite în mod cu prioritati fixe unde DRQ0 este cea mai prioritara sau cu prioritati ciclice. Linia DRQ corespunzatoare cereri trebuie mentinuta în stare activa pâna când linia DACK corespunzatoare devine activa (pâna când cererea devine recunoscuta de CPU). Tip: I.

**DACK0-DACK3**: (DMA Acknowledge) iesire utilizata pentru a semnaliza perifericului ce a cerut transfer DMA ca i s-a alocat un ciclu DMA. Nivelul activ poate fi programat. Dupa RESET toate liniile DACK sunt active în "0" logic. Tip: O.

**HRQ** : (Hold Request) linie utilizata de 8237A pentu a cere CPU controlul asupra magistralelor. Acest semnal se poate valida sau nu printr-o masca. Tip: O.

**HLDA**: (Hold Acknowledge) linie de raspuns ce vine de la CPU ca magistralele au fost eliberate in vederea unui transfer DMA. Tip: I.

**$A_0$ ... $A_3$**: linii de adrese bidirectionale TS. Ele sunt iesiri în ciclul activ si contin cei mai semnificativi patru biti ai adresei. În ciclul pasiv sunt folosite de CPU pentru a adresa registrii interni. Tip: O.

**$A_4$ ... $A_7$** : linii de adresa care sunt valabile doar în timpul transferului DMA si furnizeaza bitii $A_4$ ... $A_7$ ai adresei. Tip: O.

**$D_0$ ... $D_7$** : linii bidirectionale TS conectate la magistrala de date. În "starea program" pe aceste linii se citesc sau se scriu registrii de adresa, stare, numaratori de catre CPU. În ciclul inactiv aceste linii au semnificatia de biti de adresa $A_8$ ... $A_{15}$. Tip: I/O.

**Descrierea functionala**

În schema bloc a circuitului se observa urmatoarele module :

I) Blocul de control al comenzilor : decodificarea diferitelor comenzi transmise circuitului de catre CPU

II) Blocul de determinare a prioritatiilor : rezolva problema ordinii în care se achita cererile de transfer DMA pentru mai multe canale care o cer simultan

III) Canalele de transfer DMA : dispune de 4 canale independente programabile. În cadrul fiecarui canal exista cinci registrii (adresa de baza, numarator cuvinte, adresa curenta, numarator cuvinte curente, registru de mod).

Se poate observa ca circuitul 8237A dispune de un spatiu de memorie interna de 344 biti.

Vom prezenta, în continuare, registrii interni ai circuitului 8237A

| Nume registru | Lungime | Numar |
|---|---|---|
| Base Adress Registers | 16 biti | 4 |
| Base Word Count Registers | 16 biti | 4 |
| Curent Address Registers | 16 biti | 4 |
| Curent Word Count Registers | 16 biti | 4 |
| Temporary Address Register | 16 biti | 1 |
| Temporary Word Count Register | 16 biti | 1 |
| Status Register | 8 biti | 1 |
| Command Register | 8 biti | 1 |
| Temporary Register | 8 biti | 1 |
| Mode Register | 6 biti | 4 |
| Mask Register | 8 biti | 1 |
| Request Register | 8 biti | 1 |

**Functionarea DMA**

Circuitul 8237A functioneaza în doua moduri de ciclii. Primul dinte ei este ciclul inactiv când DMA-ul nu executa nici o cerere DMA si al doilea mod cel în care DMA ( cel putin un canal) executa o cerere si se numeste ciclu activ. În ciclu pasiv CPU poate inspecta si modifica valorile din registrii interni ai DMA-ului.

***Ciclul pasiv***

În acest ciclu controler-ul verifica daca nu apare o cerere de întrerupere. Daca linia CS si HLDA sunt în "0" logic atunci cotrolerul intra în starea de Progam Condition în care asteapta ca CPU sa inspecteze, citeasca sau sa scrie registrii interni. Registrul intern flip-flop este utilizat pentru a determina care este octetul

“low” si care este cel “hi” când se programeaza adresa si numarul de octeti care sunt reprezentati pe 16 biti. Bitul flip-flop este resetat de catre comenzile Reset si “ master clear”.

***Ciclul activ***

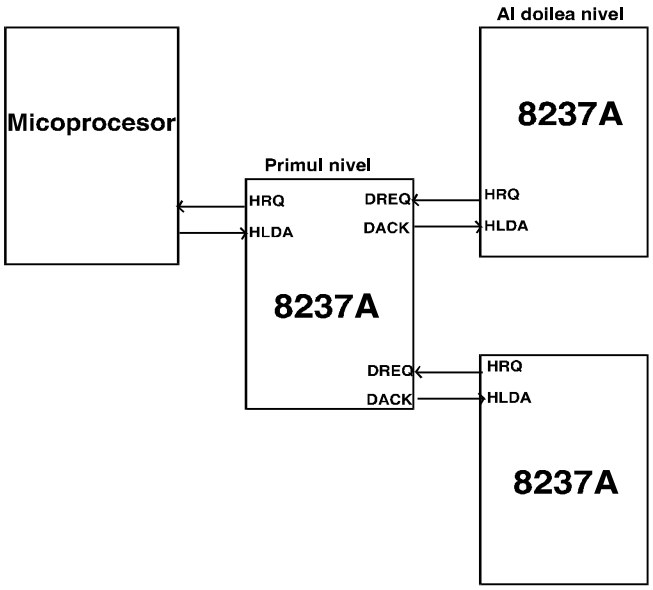
Când circuitul 8237A este în ciclul pasiv si pe una din liniile de cerere DMA nemascata vine o cerere atunci va activa linia HRQ pentru CPU si va intra în ciclul activ. În acest ciclu poate intra în unul din cele patru moduri :

*Single Transfer Mode*: în acest mod este programat sa faca doar un singur transfer. Numaratorul (Word count) va fi decrementat sau incrementat depinde cum a fost programat iar când ajunge la zero sau la FFFFH, un TC va fi transmis pentru canalul respectiv si va cauza procesul de autointializare daca canalul a fost programat pentru asa ceva.

*Block Transfer Mode* : în acest mod circuitul DMA este trebuie activat de DREQ dupa ce a fost un TC sau de un EOP extern. Din nou procesul de initializare va fi apare daca a fost programat.

*Demand Transfer Mode*: este programat sa continue sa transfere dupa ce un semnal TC a aparut sau un EOP extern, sau dupa ce DREQ a devenit inactiv.

*Cascade Mode* : În acest mod se pot conecta în cascada mai multe circuite 8237A. Acestea se pot conecta cum este aratat în figura.




**Tipuri de transfer**

Exista trei moduri de transfer care la rândul lor pot activa în trei moduri distincte: de citire, scriere si verificare. Transferul cu scriere muta date de la un port de I/O în memorie activând liniile MEMW si IOR. Transferul cu citire muta date din memorie la un port de I/O activând liniile MEMR sI IOW. Transferul cu verificare este un proces de pseudo transfer.

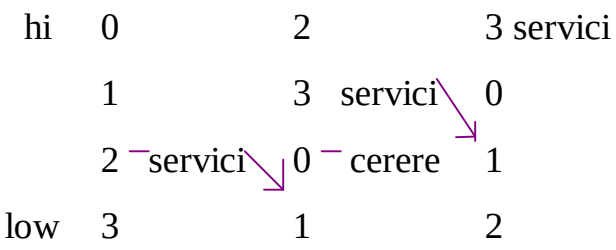
*Memorie-Memorie* - pentru a muta un bloc de date de la o adresa la o alta adresa cu un minim de efort si timp se poate utiliza facilitatea circuitului 8237A de a transfera blocuri de date memorie în memorie. Pentru acest lucru este necesar setarea unui bit din registrul de comanda pentru a selecta canalul 0 si 1 de a lucra in mod de transfer memorie în memorie. Acest proces este initializat de catre o cerere DERQ soft trimisa pentru canalul 0. În registrul care contine adresa curenta pentru canalul 0 este sursa care este incrementata sau decrementata depinde de cum a fost programata. Octetii care trebuie transferati se vor citi din memorie în registrul temporar din 8237A dupa care se va activa un ciclu de scriere în memorie pe canalul

1. Când în registrul “Word count” se va afla FFFFH se va genera un TC care va cauza un EOP.

Canalul 0 poate fi programat astfel încât sa retina aceeasi valoare pentru a putea umple o zona de memorie cu aceasi valoare.

*Autoinitializare* - Programând un bit în registrul de mod se va activa acesta optiune. În acest mod se va restaura automat adresa curenta (Curent Adress) si numarul de octeti de transferat (Curent Word Count) când un EOP este activ. Astfel DMA-ul este gata sa înceapa un nou transfer când un semnal DERQ este detectat.

*Prioritati* - 8237A suporta doua tipuri de prioritati selectabile soft. Prima dinte ele este prioritatea fixa care fixeaza fiecarui canal o prioritate de la 0 (cea mai prioritara) la 3(cea mai putin prioritara). Al doilea tip de prioritati este cel cu prioritati rotative. Aceasta presupune urmatorul algoritm: ultimul canal servit va primii cea mai mica prioritate. În acest fel se va putea evita monopolizarea serviciului DMA de catre un singur canal.




*Compress Timing* - în traducere ar însemna ca 8237A poate suprima un tact în timpul de transfer al datelor.

**Descrierea Registrilor**

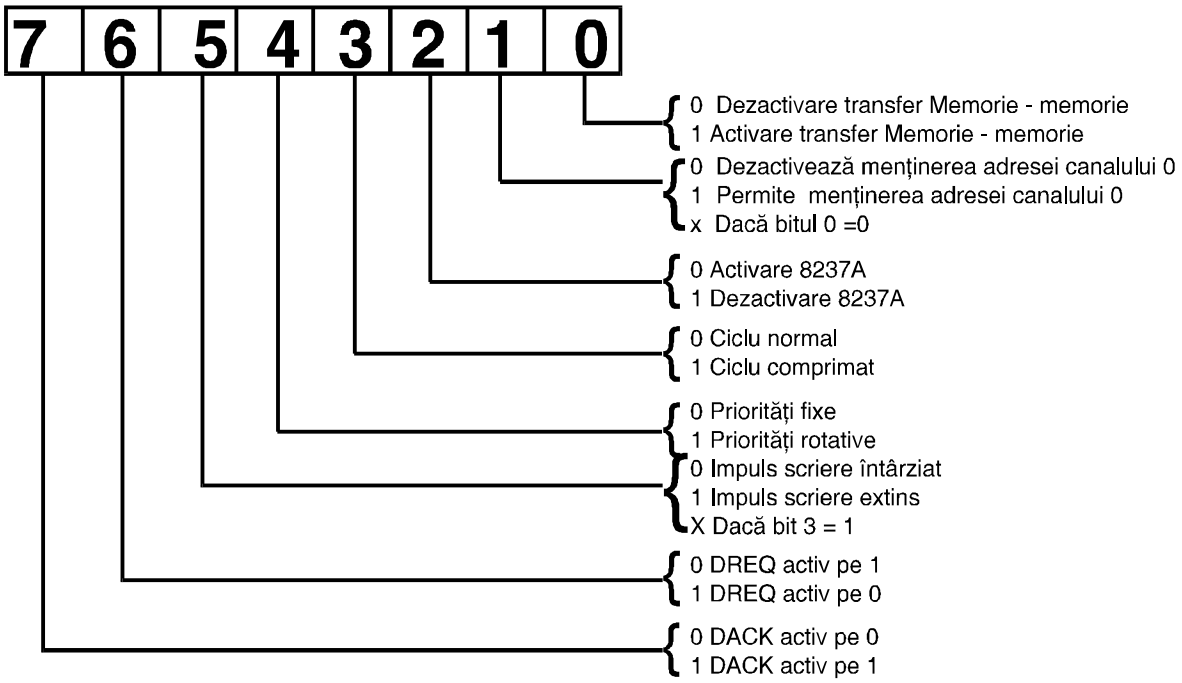
*Registrul cu adresa curenta* (Curent Adress Register): este un registru pe 16 biti care contine adresa de unde se vor transfera date. Aceasta adresa se va incrementa sau decremnta automat dupa fiecare transfer. Daca canalul este

programat cu autoinitializare adresa curenta va fi recopiata în registru doar dupa un EOP.

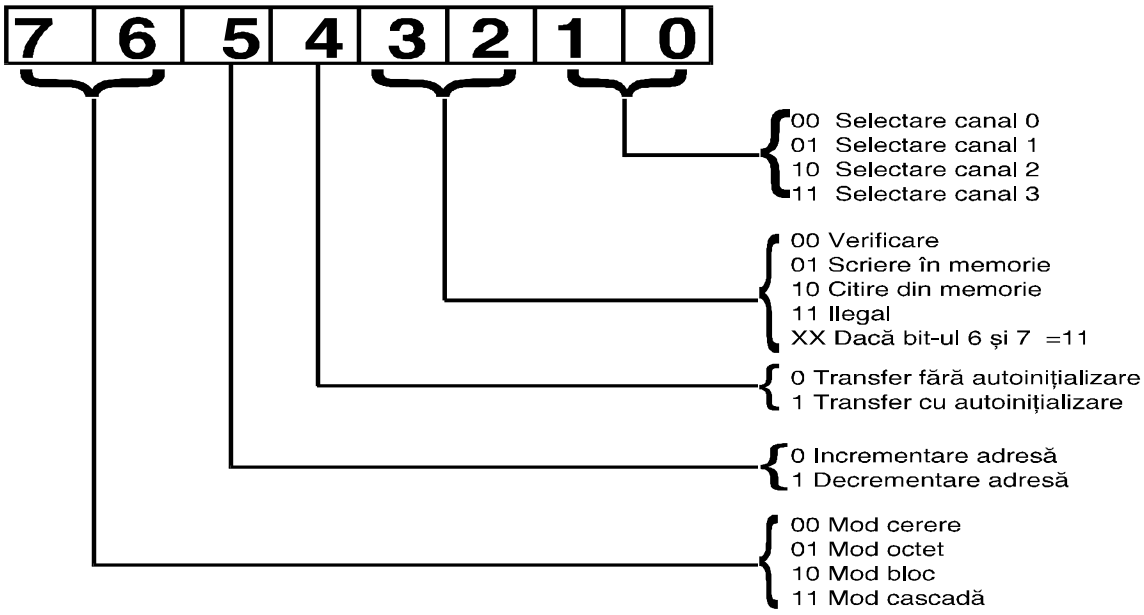
*Registru numarator de cuvinte* (Curent Word Register): este un registru pe 16 biti care stabileste câte transferuri DMA au mai ramas de efectuat. **Atentie**: numarul de transferuri efectuate va fi cu unu mai mult decât numarul înscris (Ex. daca se înscrie 100 se vor transfera 101 octeti) La trecerea din 0 în FFFFh se va genera TC. Daca canalul a fost programat cu autoinitializare valoarea originala va fi inscrisa în momentul aparitiei unui EOP (se initiaza procesul de autoinitializare)

*Registrul cu adresa de baza si cu numarul de octeti de transferat* (Base Adress and Base Word Count Registers): fiecare canal are asociat acest registru care este utilizat intern pentru procesul de autoinitializare de unde se vor copia informatiile originale. Acest registru se înscrie simultan cu scrierea primilor doi registrii descrisi mai sus, si nu poate fi citit de catre microprocesor.

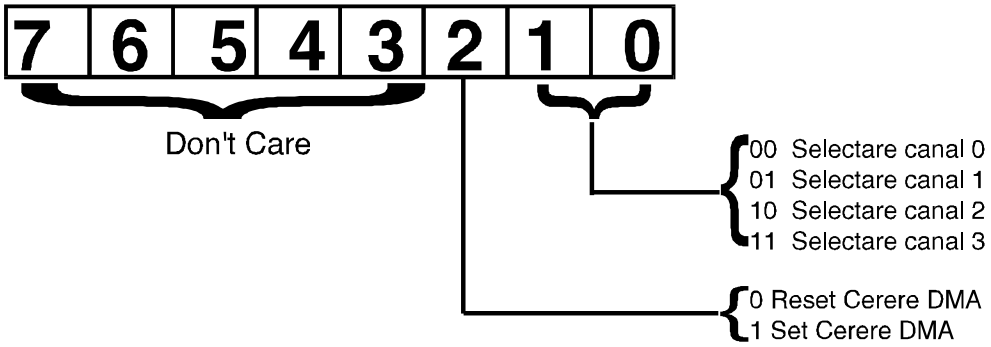
*Registru de comanda* (Command Register): Este un registru pe 8 biti care contine informatii referitoare la operatiile efectuate de 8237A. Este perogramat de CPU în timpul starii de Program Condition si este sters de RESET sau MASTER Clear. În tabelul alaturat se specifica functiile bitilor din acest octet.



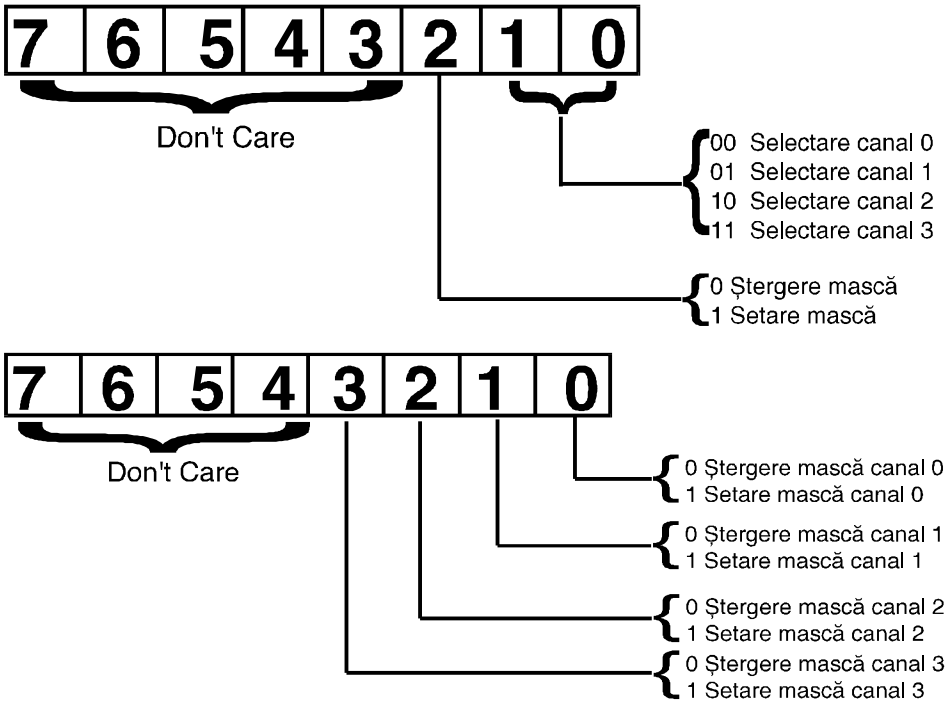
*Registrul de mod* (Mode Register): fiecarui canal îi corespunde un registru de 6 biti în care se specifica felul transferului. A se vede figura alaturata.



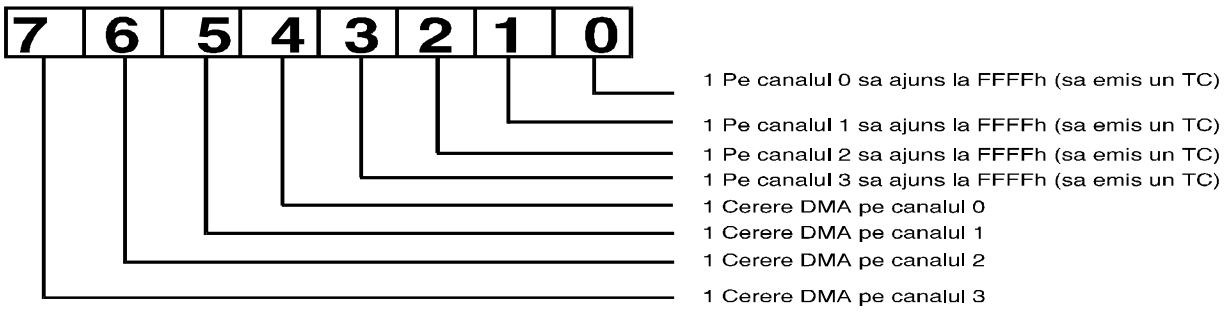

*Registrul de cerere servici DMA* (Request Register): 8237A poate raspunde unei cereri de transfer DMA care este initiata prin soft. Fiecare canal are asignata o pozitie (un bit) în registrul de cerere care arata daca se cere servici DMA pentru canalul respectiv sau nu. Acest registru nu se poate masca si respecta regula stabilita de logica de prioritati. Pentru a se vedea cum se poate seta - reseta bitul asociat uniu canal a se vedea figura alaturata. Pentru a putea un canal sa execute o cerere DMA soft acest canal trebuie sa fie programat in **mod bloc** (**Block Mode**)



*Registrul masca* (Mask Register): fiecarui canal îi este atribuit un bit cu ajutorul caruia se specifica daca acest canal este inactiv (disable) sau activ în momentul activarii liniei DREQ. Se pot stabili mastile pentru fiecare canal separat sau pentru mai multe canale în acelasi timp. A se vedea figura.




*Registrul de stare* (Status Register): este un registru care poate fi citit de CPU si care furnizeaza informatii despre fiecare canal în parte în acel moment. Aceste informatii ne spun daca, canalul respectiv este solicitat sau daca s-a emis un TC.



*Registru temporar* (Temporary Register): este un registru folosit pentru a memora octetul curent în procesul de transfer memorie - memorie. Dupa un transfer complet CPU poate citi ultimul octet transferat. Acest registru este sters dupa un RESET.

**Comenzi soft** (Software Commands): Mai exista câteva comenzi soft care pot fi executate în starea de Program Condition si care nu se reflecta direct asupra unui bit anume. Aceste comenzi sunt:

*Clear First/Last Flip Flop*: Aceasta comanda este necesara în momentul în care se doreste scrierea unui registru de 16 biti. Dupa executarea acestei comenzi, 8237A respecta urmatoarea regula: primul octet înscris va fi octetul "low" urmatorul fiind octetul "hi" s.a.m.d.

*Master Clear*: Aceasta comanda are acelasi efect cu un reset hard. Dupa executarea acestei comenzi registrii de comanda, stare, cerere, temporar, si Flip/Flop vor fi resetati si registrul masca va fi setat, dupa care 8237A va intra în *ciclul pasiv*.

*Clear Mask Register*: Acesta comanda sterge toate mastile de la cele *patru* canale si din acel moment vor putea accepta cererile de transfer.

**Programarea**

Vom prezenta în continuare modul de programare a lui 8237A doar pe 8 biti; pe 16 biti fiind absolut identic, doar porturile fizice asociate canalelor DMA fiind diferite (vezi anexa). În tabelul urmator se vor indica porturile prin intermediul caruia se poate programa 8237A.

| Port | Sens | Actiune |
|---|---|---|
| 00h | R/W | Adresa canal 0 DMA |

| | | |
|---|---|---|
| 01h | R/W | Contor canal 0 DMA |
| 02h | R/W | Adresa canal 1 DMA |
| 03h | R/W | Contor canal 1 DMA |
| 04h | R/W | Adresa canal 2 DMA |
| 05h | R/W | Contor canal 2 DMA |
| 06h | R/W | Adresa canal 3 DMA |
| 07h | R/W | Contor canal 3 DMA |
| 08h | R | Registru de stare |
| 08h | W | Registru de comanda |
| 09h | W | Registru cereri soft |
| 0Ah | W | Registru masti (invidual) |
| 0Bh | W | Registru mod |
| 0Ch | W | Stergere Flip-Flop intern |
| 0Dh | R | Registru temporar |
| 0Eh | W | Reset 8237A |
| 0Fh | W | Registru masti (toate canalele) |

Pentru a putea accesa un spatiu de adrese de 1MB sunt necesare 20 linii de adresa, dar fiindca 8237A nu dispune doar de 16 linii de adresa s-a gasit solutia urmatoare. Din adresa pe 20 de biti se separa primii 4 care se vor înscrie în asa numitul registru pagina (registru intern). O pagina este un bloc de memorie de 64 Ko (analog cu adresa de segment). Astfel s-a împartit primul megaoctet de memorie dupa cum se poate vedea în urmatorul tabel:

| Pagina | Segment : Ofset |
|---|---|
| 0 | 0000:0000 – 0000:FFFF |
| 1 | 1000:0000 – 1000:FFFF |
| 2 | 2000:0000 – 2000:FFFF |
| 3 | 3000:0000 – 3000:FFFF |
| 4 | 4000:0000 – 4000:FFFF |
| 5 | 5000:0000 – 5000:FFFF |
| 6 | 6000:0000 – 6000:FFFF |
| 7 | 7000:0000 – 7000:FFFF |
| 8 | 8000:0000 – 8000:FFFF |
| 9 | 9000:0000 – 9000:FFFF |
| A | A000:0000 - A000:FFFF |
| B | B000:0000 - B000:FFFF |
| C | C000:0000 - C000:FFFF |
| D | D000:0000 - D000:FFFF |
| E | E000:0000 - E000:FFFF |
| F | F000:0000 - F000:FFFF |

Porturile de pagina asociate canalelor DMA sunt:

| Canal DMA | Port asociat |
|---|---|
| 0 | 87h |
| 1 | 83h |
| 2 | 81h |
| 3 | 82h |
| 4 | 8Fh |

| | |
|---|---|
| 5 | 8Bh |
| 6 | 89H |
| 7 | 8Ah |

Ei bine daca cunoastem pagina si segmentul de unde dorim sa transferam date, numarul de octeti de transferat putem trece la programarea propriu-zisa. Pasii necesari programarii cu succes sunt urmatorii :

1. Dezactivare canal
2. Reset octet F/F
3. Set mode
4. Set page
5. Set offset
6. Set lungime bloc
7. Enable channel
8. Programarea perifericului de a permite transfer DMA

În continuare se vor prezenta doua exemple de programare a lui 8237A. Primul arata cum s-ar putea testa daca un canal DMA functioneaza corect sau nu prin soft. Cel de-al doile exemplu arata cum se poate initia un transfer DMA memorie - memorie.

*Exemplul 1* :

```
Prog        segment at 0FFF0h
            assume cs:prog

Start       Label       FAR
```

```
DMA     EQU             0
        Mov al,0
        Out 83h,al      ;Initializare registru pagina
        Mov al,4
        Out DMA+8,al    ;Dezactivare DMA
        Out DMA+0Dh,al  ;Reset DMA
        Mov al,0FFh
R1:     Mov bl,al
        Mov bh,al
        Mov cx,8
        Xor dx,dx
R2:     Out dx,al       ;Scrie octet Low
        nop
        Out dx,al       ;Scrie octet High
        Mov ax,0101h
        In al,dx        ;citire High
        Mov ah,al
        In al,dx        ;Citire Low
        Cmp bx,ax       ;Verificare înscriere
        Jz C1
        Jmp Eroare
C1:     Inc dx          ;urmatorul registru DMA
        Loop R2
        Inc al          ;Configuartie de test 0000h
        Jz R1           ;Test O.K.
R3:     In al,0A0h      ;Seexecuta o citire de la portul 0A0h
        Mov cx,10h
```

```
C2:         Loop c2          ;Bucla de întârziere
            Jmp R3           ;Bucla în caz de eroare
Eroare:     In al,0a0h       ;se executa doua citiri de la portul 0a0h
            nop
            In al,0a0h
            Mov cx,10h
C3:         Loop C3          ;Bucla de întârziere
            Jmp Eroare

            Org 0F0h
            Jmp Start
Prog        ends
end start
```

*Exemplul 2* **:**

```
            .model small

            .data

mesage      db      10,13,'11111111111111111111111111','$'

clear_fl    equ     0ch
 mask01     equ     0ah
 un_mask    equ     0ah
 ch0        equ     00h
 ch1        equ     02h
```

```
ch1_cnt     equ     03h
ch0_cnt     equ     01h
page0       equ     087h
page1       equ     083h
mode        equ     0bh
cmmd        equ     08h
dma_request equ     09h

page_0      db      '0'
page_1      db      '0'
off0        dw      '0'
off1        dw      '0'

message1    db      10,13,'22222222222222222222222','$'

            .code
pause_      proc
            push ax
            push cx
            push dx
;Motivul acestui macro este acela de a lasa un timp minim între accesul la porturile
;hard.
;Desi pe unele masini foarte rapide s-ar putea sa nu existe nici o diferenta importanta
;de timp
            mov  cx,500
            mov  dx,61h
IAR:        in   al,dx      ;in   al,dx
```

```
                loop IAR
                pop dx
                pop cx
                pop ax
endp

; pt. intrare dx:ax=segment:offset adress
; out  DH = Page(0-F)
; AX = Offset

MakePage proc                   ;procedura care calculeaza pagina si ofsetul
                push bx
                mov bl,dh
                shr bl,1
                shr bl,1
                shr bl,1
                shr bl,1
                shl dx,1
                shl dx,1
                shl dx,1
                shl dx,1
                add ax,dx
                adc bl,0
                mov dh,bl
                pop bx
                ret
endp MakePage
```

```
start:
        mov ax,@data
        mov ds,ax
        mov dx,ax           ;sursa segmentul de date curent
        mov ax,offset mesage
        call MakePage
        mov byte ptr ds:[page_0],dh
        mov word ptr ds:[off0],ax
        mov dx,@data        ; destinatia
        mov ax,offset message1
        call MakePage
        mov byte ptr ds:[page_1],dh
        mov word ptr ds:[off1],ax

;Scriu pe ecran sirurile initiale
        mov ah,09h
        xor al,al
        mov dx,offset mesage
        int 21h
        mov dx,offset message1
        int 21h

;Pregatesc DMA
;se fac canalele Disable pt. a putea fi programate
        mov al,0100b
        out mask01,al       ;setez maska pt canalul 0 deci îl fac disable =>
                            ;acum îl pot programa
```

```
 pause_
            mov al,05h
            out mask01,al              ;acelasi lucru pt. canalul 1
pause_
            mov al,0;
            out clear_fl,al            ;clear FL primul pas care trebe facut
pause_
            mov al,10001000b ;modul pt. canal 0
            out mode,al
pause_
            mov al,10000101b ;modul pt. canal 1
            out mode,al
pause_
            mov byte ptr al,ds:[page_1] ;setez pagina pt. canalul 1
            out page1,al
pause_
            mov byte ptr al,ds:[page_0] ; setez pagina pt. primul canal 0
            out page0,al
pause_
            mov word ptr ax,ds:[off0]
            out ch0,al
pause_
            mov al,ah
            out ch0,al          ;programez adresa pt. primul canal 0
pause_
            mov word ptr ax,ds:[off1þ
            out ch1,al
```

```
pause_
        mov al,ah
        out ch1,al          ;acelasi lucru pt. canalul 1
pause_
        mov ax,27
        out ch0_cnt,al
pause_
        mov al,ah
        out ch0_cnt,al      ;programez numarul de octeti de mutat se time
                            seama de ;canalul 1 si nu de canalul 0
pause_
        mov ax,27
        out ch1_cnt,al
pause_
        mov al,ah
        out ch1_cnt,al      ;programez numarul de octeti de mutat se time
                            ;seama de canalul
                            ;1 si nu de canalul 0
pause_
        mov al,00001001b ;enable memory to memory transfer
        out cmmd,al
pause_
        mov al,001b
        out un_mask,al      ;demaschez calalele DMA 0 si 1
pause_
        mov al,000b
        out un_mask,al
```

```
pause_
        mov al,100b         ;pt canalul 0
        out dma_request,al ;cerere DMA
pause_
        mov ah,09h          ;Scriere mesaj dupa transfer
        xor al,al
        mov dx,offset mesage
        int 21h
        mov dx,offset message1
        int 21h
        mov ax,4c00h
        int 21h
end start
end code
```

**Bibliografie**


[1] **Dobrota V. si altii** - *Aplicatii în sisteme cu microprocesoare din familia intel 80x86*, vol.2.

[2] **Athanasiu I., Panoiu A.** – *Calculatoare Personale*: *Microprocesoarele 8086, 286, 386*, Editura Teora, Bucuresti, 1993.