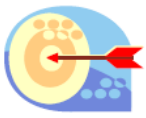
2.Transferul de date

Cuprins

Cuprins modul

Introducere

Modulul "Transferul de date" abordează problema circulației datelor între procesor şi dispozitivul de I/O la nivel fundamental. Sunt tratate cele trei moduri de transfer, programat, prin întreruperi şi prin DMA. Transferul este exemplificat la familia de procesoare x86 prin gestionarea întreruperilor cu controllerul specializat I8259 şi gestionarea transferului DMA cu controllerul I8257. O prezentare generală a programelor de gestionare a transferului şi câteva aspecte particulare a sistemelor de întreruperi şi DMA la microcontrollere încheie acest modul



Obiective

După parcurgerea acestui modul studenții vor şti modurile fundamentale de transfer şi vor putea să asimileze modulele următoare cu mai mare uşurință. Studenții vor putea să:

- Lucreze cu modul de transfer programat, şi în special cu modul de interogare, des utilizat în aplicații;
- Conceapă o aplicație cu transfer prin întreruperi şi prin DMA;
- Aprecieze diferențele între modurile de transfer şi să aleagă modul de transfer optim pentru o anumită aplicație.

Obiective specifice:
1.Învățarea principiilor fundamentale de transfer de date
2.Învățarea noțiunii de protocol cu aplicare la transferul de date prin interfețe
3.Înțelegerea noțiunilor prin exemplificări practice

Durata medie de studiu individual

Durata medie de studiu individual este de 2 ore.

2.1. Clasificare

Transferul de date studiat în acest modul este cel care are loc între un dispozitiv de I/O şi unitatea centrală. Transferul poate fi:

1. Programat;
2. Prin întreruperi;
3. Prin acces direct la memorie, DMA (Direct Memory Access).

Software-ul care controlează transferul de date se numeşte driver soft. Funcțiile acestui software sunt:

1. Inițializarea dispozitivului de I/O;
2. Inițierea transferului de date;
3. Transferul de date;
4. Terminarea transferului şi raportarea rezultatelor transferului.

Inițializarea dispozitivului de I/O se face la punerea sub tensiune a dispozitivului sau la un RESET generat de unitatea central. Să ne aducem aminte cum face o imprimantă cu jet la pornire- un set de operații de inițializare şi calibrare, un hard disc- testul de integritate a suportului, care se manifestă vizibil sau prin sunete caracteristice.

Comanda software a unui dispozitiv de intrare ieşire este realizată de:

1. Driver-ul software din calculatorul gazdă;
2. Programul executat de microprocesorul sau microcontrollerul dispozitivului de I/O, dacă acesta dispune de o unitate centrală.

Împărțirea sarcinilor între driver şi programul din dispozitivul de I/O depinde de fiecare aplicație, tendința actual fiind de preluare a cât mai multe sarcini de către dispozitivul de I/O pentru eliberarea timpului consumat de calculatorul gazdă.

În cazul unui sistem de calcul din familia x86 transferul poate fi înțeles pornind de la schema bloc din figura 2.1. Transferurile efectuate de procesor prin program pot fi:

A.Citire din memorie, MOV AL, [BX]

Pe magistrala de adrese este pusă adresa de memorie din registrul BX, care identifică o locație de memorie. Pe magistrala de comenzi este activat semnalul MEMR. Conținutul locației de memorie adresată este pus pe magistrala de date şi intră în registrul AL al microprocesorului (linii cu roşu).

B.Scriere în memorie, MOV [BX], AL

Pe magistrala de adrese este pusă adresa de memorie din registrul BX, care identifică o locație de memorie. Pe magistrala de comenzi este activat semnalul MEMW. Conținutul registrului AL este pus pe magistrala de date şi este salvat în locația adresată.

C.Citire de la un dispozitiv de I/O, IN AL, DX

Pe magistrala de adrese este pusă adresa dispozitivului din registrul DX, care identifică dispozitivul. Pe magistrala de comenzi este activat semnalul IOR. Conținutul portului de intrare este pus pe magistrala de date şi intră în registrul AL al microprocesorului (linii cu albastru).

D.Scriere la un dispozitiv de I/O, OUT DX, AL

Pe magistrala de adrese este pusă adresa dispozitivului din registrul DX, care identifică dispozitivul. Pe magistrala de comenzi este activat semnalul IOW. Conținutul registrului AL al microprocesorului este pus pe magistrala de date şi este trimis la portul de ieşire.

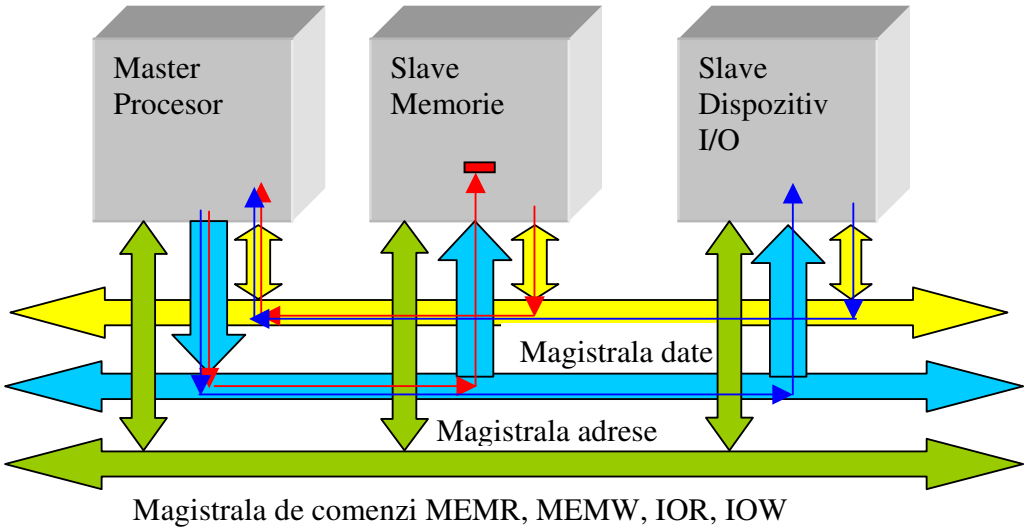


Figura 2.1. Schema bloc principială a unui sistem x86

2.2.Transferul programat

În acest mod de transfer toate operațiile de I/O sunt sub controlul programului. Transferul programat are două variante, direct şi prin interogare (polling).

a.Transferul programat direct

În transferul programat direct calculatorul gazdă citeşte sau scrie un port de intrare / ieşire în mod direct. De exemplu secvența de program scrisă în limbajul de asamblare x86:

MOV DX, adresa port	se încarcă în DX adresa portului de I/O
IN AL, DX	se citesc date de la portul adresat
MOV [BX], AL	se salvează datele în memorie, la adresa specificată de BX
MOV AL, date	se încarcă date în acumulator

OUT DX,AL datele se trimit la portul cu adresa specificată în DX

Secvența anterioară citește date de la un port cu adresa specificată în registrul DX, le salvează în memorie și scrie apoi date în același port.

b.Transferul programat prin interogare

Fiecărui dispozitiv de I/O i se atribuie un bit (fanion) care indică faptul că acesta este gata pentru transfer. Presupunem 8 dispozitive de I/O care au atașate 8 porturi de date și un port pentru citirea fanioanelor, structura fiind dată în figura 2.2.

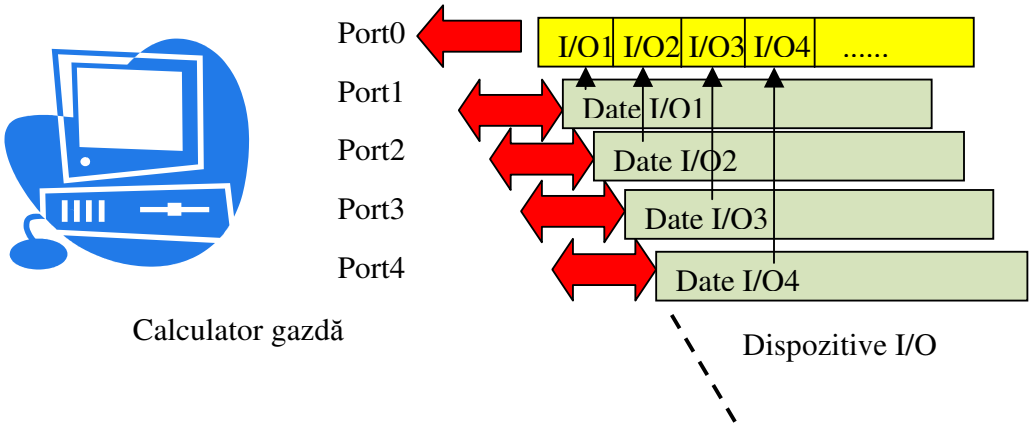


Figura 2.2. Transferul prin interogare

Fanioanele sunt interogate ciclic și se servește acel periferic care solicită un transfer de date prin schimbarea stării fanionului. Dacă de exemplu cererea se face cu nivel logic 1 al fanionului, un exemplu de program este:

```
Start:  MOV DX, adresa Port0
        IN AL,DX          se citesc fanioanele
        JN adresa1        la adresa1 este secvența de program de transfer cu I/O1
        RCL AL,01         deplasare AL la stânga
        JN adesa2         la adresa2 este secvența de program de transfer cu I/O2
        ……
        JMP start
```

Se citește portul care conține fanioanele, apoi se verifică primul bit. Dacă cel mai semnificativ bit este 1 numărul binar din acumulator este negativ și dispozitivul solicită

un transfer. După testarea tuturor celor 8 fanioane bucla de citire şi testare este reluată. Secvenţele de program de transfer pentru fiecare dispozitiv sunt secvenţe de transfer programat direct, aşa cum au fost descrise anterior. Desigur că acest program este unul principial, care trebuie completat şi extins pentru a deveni un program funcţional.

Prioritatea dispozitivelor de I/O care sunt servite poate fi stabilită prin poziţia fanionului corespunzător în octetul de fanioane. Dacă mai multe dispozitive solicită un transfer în acelaşi timp, servirea se face în ordinea poziţiei fanioanelor. Această ordine se poate modifica software.

Dacă un dispozitiv de I/O este mai rapid decât celelalte fanionul lui poate fi testat mai des în bucla de citire şi testare, caz în care citirea portului cu fanioane trebuie realizată înainte de testarea fiecărui fanion.

Prin software se poate **masca** un dispozitiv de I/O prin excluderea din bucla de testare a fanionului.

Pentru a evita o testare continuă, printr-o logică combinaţională se poate semnaliza când un dispozitiv de I/O are nevoie de un transfer de date, printr-un SAU logic realizat hard între cele 8 fanioane şi care să fie citit de procesor, figura 2.3.

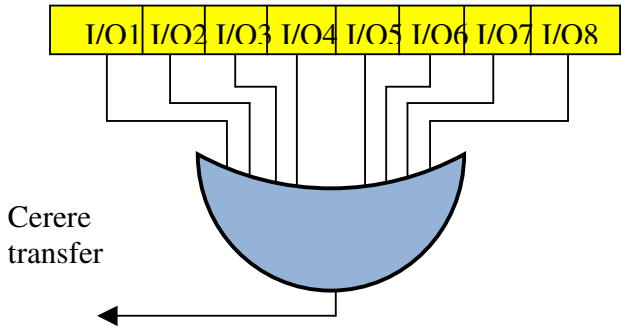


Figura 2.2. Semnalizarea unei cereri de transfer

Acest mod de lucru este foarte des folosit atunci când se combină transferul programat cu cel prin întreruperi. În acest caz semnalizarea unei cereri de transfer se face printr-o întrerupere, după care unitatea centrală identifică dispozitivul care are nevoie de transfer prin interogare.

Dacă există mai mult de 8 dispozitive de I/O care pot solicita un transfer sau sunt mai puţine linii în portul de citire a fanioanelor, fanioanle pot fi codificate, procesorul citind astfel codul binar al dispozitivului care solicită un transfer.

Avantajele transferului programat sunt simplitatea hardware şi software şi faptul că există controlul strict al timpului în care se face un transfer de date.

Dezavantajele transferului programat sunt:

1. Sistemul se ocupă în cea mai mare parte a timpului cu dispozitivele de I/O, devenind dedicat acestei sarcini;
2. Timpul de răspuns la o solicitare este mare, programul de interogare consumând timp.

Ce exemplu cunoasteți în care timpul de execuție al unei secvențe de program este foarte important?

La achiziția semnalelor, la sistemele în timp real eşantionarea trebuie să fie realizată periodic, rutina având aceeaşi durată, altfel conversia va fi eronată.

2.3.Transferul prin întreruperi

Transferul prin întreruperi are **avantajul** că gestionează mai bine timpul procesorului sau microcontrollerului. Procesorul rulează un program care este întrerupt la apariția unei cereri de transfer şi este reluat după ce transferul s-a terminat. Un alt **avantaj** este servirea mai rapidă pentru că programul principal se poate întrerupe în orice moment. Transferul efectiv durează la fel de mult ca şi prin transfer programat.

Familiile de procesoare au structuri diferite ale sistemului de întreruperi. Familia de procesoare x86 dispune de protocolul de dialog din figura 2.4.

Dispozitivul care cere o întrerupere va lansa un semnal INT. Procesorul verifică dacă întreruperile sunt validate cu semnalul INTE. Dacă întreruperile nu sunt validate cererea de întrerupere se ignoră. Dacă sunt validate atunci termină instrucțiunea în curs şi salvează în stivă conținutul registrului PC (Program Counter) şi a registrului de stare. Procesorul generează semnalul de acceptare a întreruperii INTA şi aşteaptă ca dispozitivul de I/O să pună pe magistrala de date adresa de salt la care se află rutina de servire a întreruperii.

Activarea şi dezactivarea sistemului de întreruperi se face prin program cu instrucțiunile STI şi CLI. Dezactivarea sistemului de întreruperi se face automat la RESET-ul sistemului sau după semnalul de acceptare a unei cereri de întrerupere INTA.

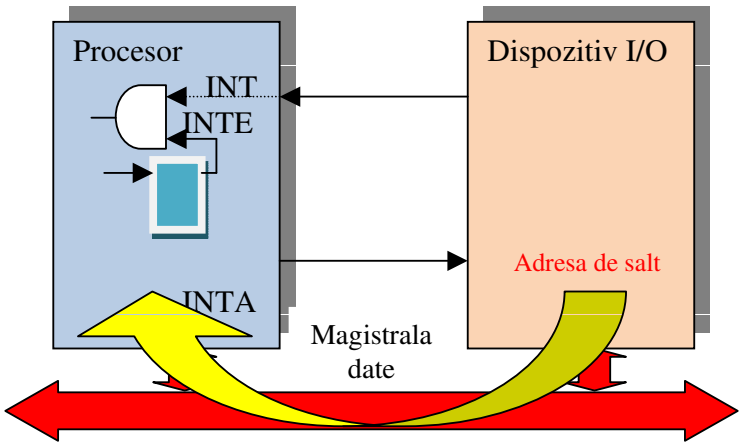


Figura 2.4. Protocolul de cerere şi acceptare a întreruperii

Pentru gestionarea mai multor întreruperi se poate folosi un controller specializat, aşa cum este de exemplu Intel 8259, care poate gestiona 8 nivele de întrerupere dar care permite şi cascadarea pentru mai mult de 8 nivele, figura 2.5.

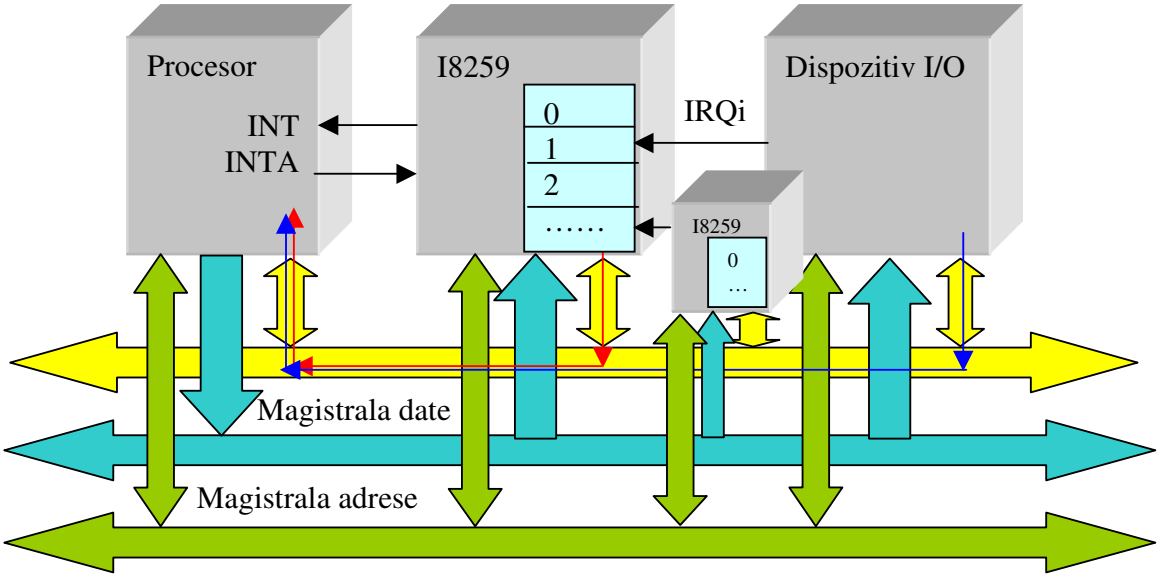


Figura 2.5. Sistemul de întreruperi gestionat de controllerul I8259

Dispozitivul de I/O cere o întrerupere IRQi. I8259 analizează cererea şi dacă nu este mascată şi nu este în curs de servire o întrerupere mai prioritară cere o întrerupere INT către procesor. Procesorul analizează cererea şi dacă o acceptă răspunde cu INTA către I8259. Controllerul I8259 pune pe magistrala de date adresa de salt la rutina de servire a întreruperii (traseul cu săgeți roşii). În rutina de servire a întreruperii datele se transferă între dispozitivul de I/O şi procesor (traseul cu săgeți albastre).

Punerea adresei de salt de către dispozitivul de I/O avea dezavantajul că limita portabilitatea dispozitivelor de I/O, acestea putând fi funcționale pe anumite sisteme şi nefuncționale pe altele. Utilizarea controllerului I8259 care are sarcina de a pune adresele de salt rezolvă această problemă.

Prioritatea cererilor de întrerupere se poate programa. Implicit IRQ0 este cel mai prioritar. Se poate programa **mascarea** unor cereri de întrerupere cu un registru de măşti. Controllerele de întrerupere pot fi cascadate, în acest caz numărul de linii de întrerupere poate creşte. Dacă se cuplează două controllere cascadate, numărul liniilor de întrerupere disponibile devine 15.

Din punct de vedere software diferența între numărul de linii de adresă şi de date la un procesor implică dificultăți la transmisia unei adrese de salt pe magistrala de date. În general salturile se fac într-o zonă redusă de memorie, numită tabel al vectorilor de întrerupere. În acest tabel nu se află câte o rutină de servire pentru fiecare nivel de întrerupere ci câte o instrucțiune de salt la o astfel de rutină, din cauza lipsei de spațiu.

Sarcinile controllerului I8259 şi datele cu care se programează sunt date în următorul tabel:

Sarcini I8259	**Programare**
Generare protocol de cerere şi acceptare întrerupere	-
Gestionare priorități	registru de definire a priorităților
Mascare selectivă	registru de măşti
Punere adrese de salt pe magistrala de date	adrese de salt

2.4.Transferul prin DMA

La transferul programat şi la cel prin întreruperi datele circulă între dispozitivul de I/O, acumulatorul procesorului şi memorie. Dacă dispozitivele de I/O sunt rapide, cu o viteză comparabilă cu cea a procesorului transferul se poate face prin DMA (Direct Memory Access). Acesta este cel mai rapid mod de transfer de date şi se poate face doar prin intermediul unui controller specializat, figura 2.6.

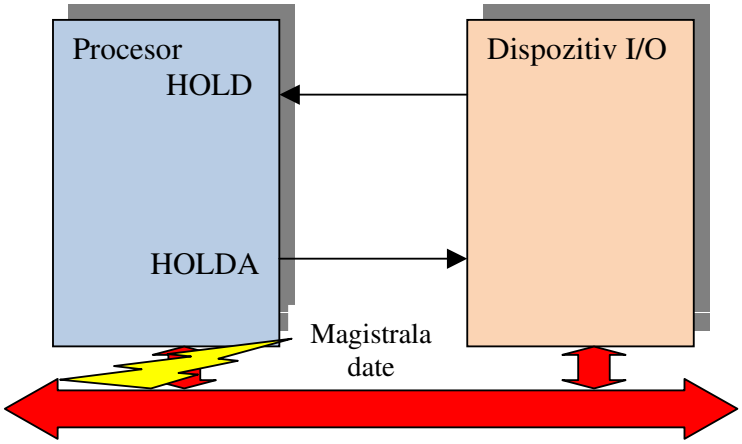


Figura 2.6. Protocolul de cerere şi acceptare de transfer DMA

Dispozitivul de intrare ieşire solicită un transfer DMA prin semnalul HOLD solicitând ca procesorul să-şi suspende activitatea prin trecerea magistralelor de date în înaltă impedanță. Când procesorul acceptă această cerere generează un semnal de HOLDA (HOLD Acknowledgement) şi trece magistralele în înaltă impedanță. Deoarece în acest caz nu mai există un dispozitiv master pe care să pună adrese şi semnale de comandă pe magistrală, acest mod de transfer are nevoie de un controller specializat care să preia rolul de master.

O schemă bloc a transferului DMA în familia x86 gestionat de controllerul Intel 8257 este dată în figura 2.7.

La primirea unei cereri de transfer DMA (numită DRQ) de la dispozitivul de I/O pe unul dintre cele 4 canale ale I8257, controllerul analizează cererea. Dacă cererea nu este mascată şi dacă nu este în curs de de execuție un transfer DMA, controllerul cere procesorului suspendarea activității cu semnalul HOLD. În momentul acceptării cererii procesorul activează semnalul HOLDA şi controllerul răspunde dispozitivului I/O că cererea a fost acceptată cu semnalul HOLDA.

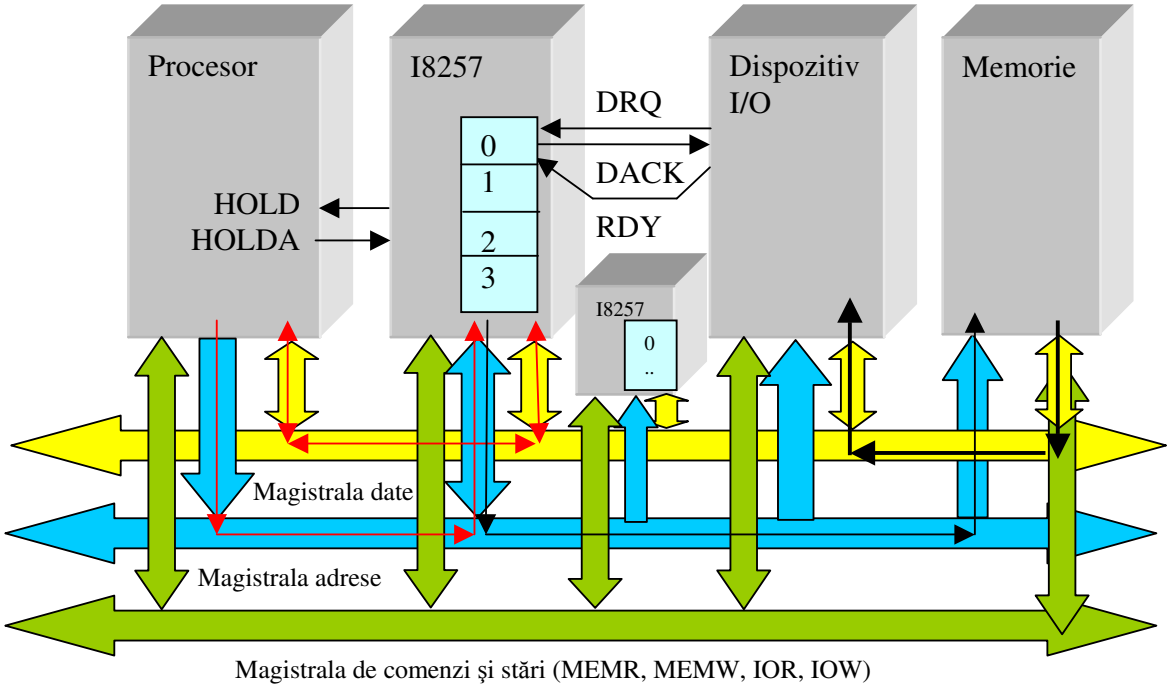


Figura 2.7. Sistemul DMA gestionat de controllerul I8257

După acceptarea cererii datele se transferă între dispozitivul de I/O şi memorie (linia neagră), fără a mai trece prin procesor. Controllerul DMA devine master pe magistrală şi pune adresele de memorie, precum şi semnalele de comandă MEMR, MEMW, IOR, IOW) (linii negre). Atunci când procesorul este master pe magistrală, controllerul DMA este slave şi poate fi adresat în vederea programării lui. În acest caz procesorul pune adrese şi comenzi (linii roşii). Viteza dispozitivului de I/O este diferită de a memoriei (mai mică) şi sincronizarea datelor transferate se face cu semnalul RDY. Arbitrarea magistralelor se face doar la primul cuvânt, transferul având loc în continuare în salvă.

Prioritatea cererilor de DMA se poate programa. Implicit DRQ0 este cel mai prioritar. Se poate programa **mascarea** unor cereri de DMA cu un registru de măşti. Controllerele de DMA pot fi cascadate, în acest caz numărul de linii poate creşte. Dacă se cuplează două controllere cascadate, numărul liniilor disponibile devine 7.

Sarcinile controllerului I8257 şi datele cu care se programează sunt date în următorul tabel:

Sarcini I8257	Programare
Generare protocol de cerere şi acceptare DMA	-
Gestionare priorităţi	registru de definire a priorităţilor
Mascare selectivă	registru de măşti
Punere adrese pe magistrala de adrese şi comenzi pe magistrala de comenzi	Adresa de început a zonei de memorie şi numărul de cuvinte de transferat

2.5.Programe de comandă a transferului

În principiu, oricare ar fi metoda de transfer, programat, prin întreruperi sau prin DMA, programul care comandă transferul trebuie să aibă trei părţi:

1. Iniţializare transfer date
2. Transfer date
3. Terminarea transferului de date

Transferul de date programat

1.În segmental de iniţializare se scrie într-un contor lungimea zonei de memorie din care se citesc, sau în care se scriu date, fixează adresa de început şi porneşte lucrul cu dispozitivul de I/O.

2.În segmental de transfer de date începe scrierea în memorie sau citirea din memorie şi trimiterea la dispozitivul de I/O. Se decrementează contorul de cuvinte transferate, se incrementează adresa de memorie şi se verifică dacă citirea / scrierea s-a făcut corect (cu bit de paritate sau CRC). Dacă s-a detectat o eroare se reia de la ultimul cuvânt corect sau se abandonează transferul. Se verifică dacă s-au scris /citit toate datele printr-o metodă care ţine cont de natura datelor (un caracter special la codificarea ASCII etc.)

3.Dacă s-a întâlnit caracterul de terminare se comunică raportul transferului- număr de cuvinte transferate, existenţa unor erori. La sfârşit se comandă oprirea transferului.

Transferul prin întreruperi

1.În plus faţă de operaţiile de la transferul programat se iniţializează controllerul de întreruperi şi se validează întreruperile. La sfărşitul etapei porneşte transferul.

2.Începe când perifericul generează o cerere de întrerupere. Se salvează în stivă datele programului principal şi se trece la servirea întreruperii. Transferul are loc ca şi la transferul programat.

3.După comunicarea raportului se reiau din stivă datele programului întrerupt.

Transferul DMA

1.În plus faţă de operaţiile de la transferul programat se iniţializează controllerul DMA. La sfărşitul etapei porneşte transferul.

2.De transfer se ocupă controllerul DMA.

3.Printr-o cerere de întrerupere controllerul DMA anunţă că a terminat transferul. Se citeşte starea dispozitivului de I/O, dacă transferul a avut erori şi numărul de cuvinte transferate.

Un program de inventariere a resurselor unui PC (http://www.sysinfolab.com/) arată zona de memorie alocată unui dispozitiv de I/O, în figura 2.8. în cazul interfeţei SATA. Alte informaţii arată numărul liniei de întrerupere şi modul DMA de lucru cu hard discul prin interfaţa SATA.

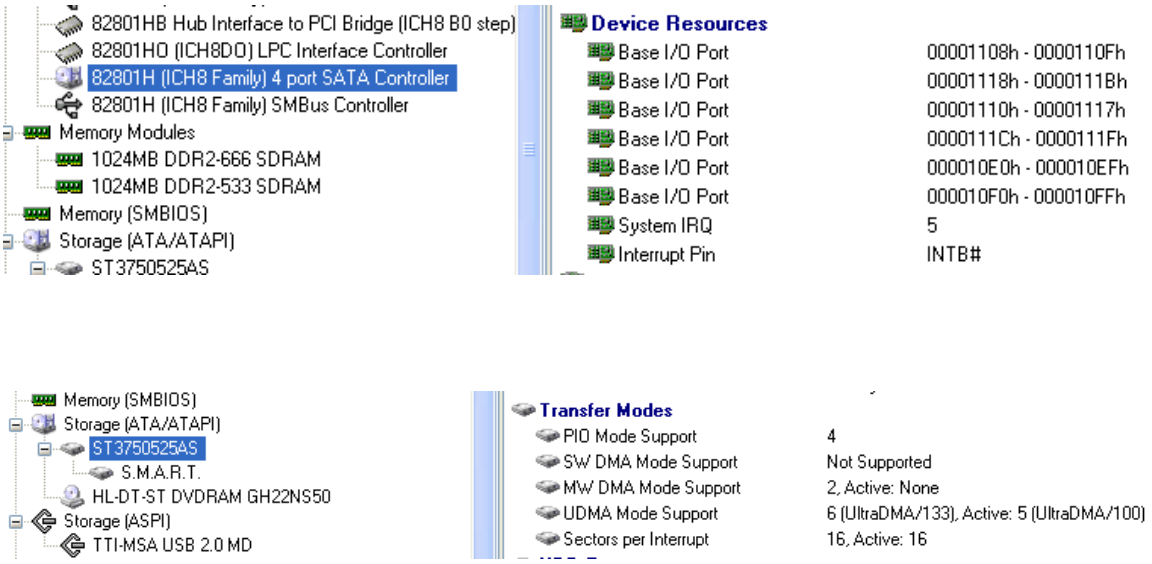


Figura 2.8.Configurarea unui hard disc cu interfaţă SATA şi alocarea zonei de memorie, nivele de întrerupere şi DMA

2.6.Sistemele de întreruperi şi DMA în microcontrollere

La un microcontroller sursele de întrerupere pot fi externe (semnale cuplate la pini), figura 2.9. sau por fi interne, de la interfețele integrate în microcontroller, cum sunt convertorul analog digital, timerul sau interfața serială. Cererile de întrerupere pot fi mascate cu un registru de măşti programat anterior în microcontroller de programul utilizatorului. O cerere nemascată va fi transmisă unității centrale care întrerupe programul curent şi face un salt la o adresă dintr-un tabel de adrese, modul particular de salt fiind specific diverselor familii de microcontrollere.

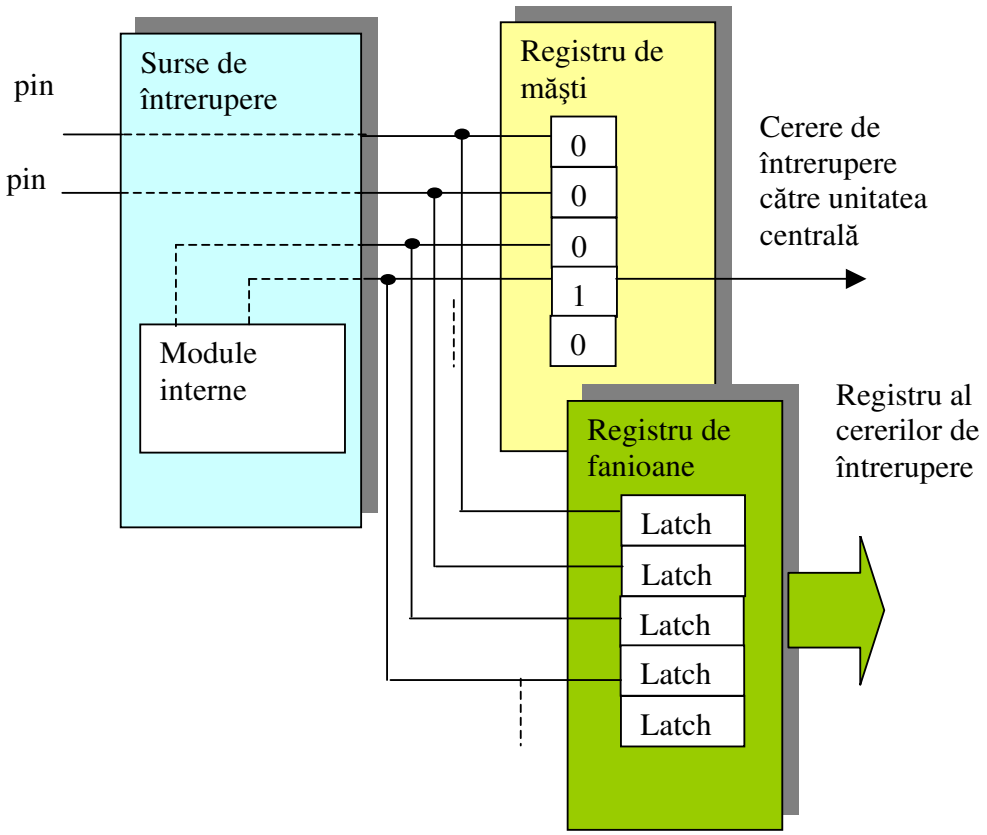


Figura 2.9. Sistemul de întreruperi la microcontrollere

Activarea unei întreruperi are ca efect şi poziționarea unui fanion în registrul de fanioane, permițând microcontrollerului să identifice sursa întreruperii. Registrul de fanioane este citit de programul utilizator prin transfer programat şi prin interogare se determină sursa întreruperii. Acesta este modul combinat între transferul programat prin interogare şi transferul prin întreruperi.

Este posibilă utilizarea exclusiv a transferului prin interogare prin interogarea în buclă a surselor de întrerupere în programul principal. În cazul unei aplicații de termometru electronic citirea senzorului de temperatură se poate face prin interogare, periodic. Dacă cumva programul principal durează mai mult, de exemplu pentru că s-a detectat apăsarea unui buton prin care se modifică programarea termometrului, temperatura este citită mai târziu, ceea ce nu afectează utilizatorul. Dacă într-un alt exemplu se citeşte starea unui buton prin interogare, trebuie ca programul principal în nicio situație să nu depăşească timpul de apăsare scurtă a butonului de către utilizator, deci trebuie făcute unele calcule. Dacă se utilizează sistemul de întreruperi nu trebuie făcut niciun calcul.

Transferul DMA este implementat în microcontrollerele mai complexe, care trebuie să asigure viteze mari de transfer şi să prelucreze cantități mari de date. O descriere sumară a avantajelor transferului DMA la microcontrollerele din familia STM32, cu schema bloc din figura 2.10.

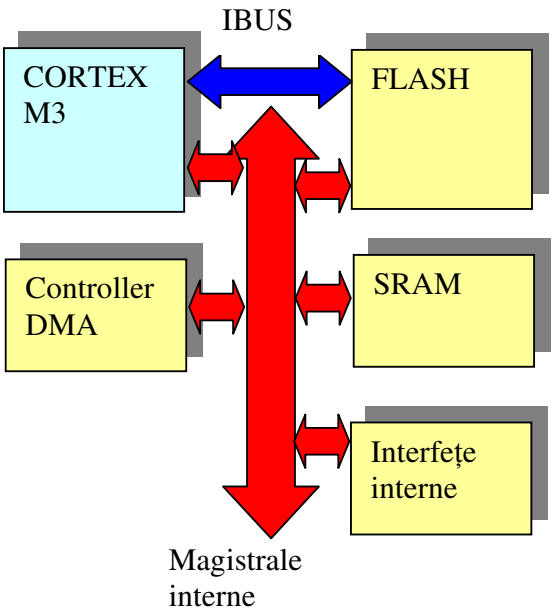


Figura 2.10. Schema bloc a familiei de microcontrollere STM32

Controllerul DMA are 7 canale, iar la modelele mai complexe există 2 controllere DMA cu 12 canale independente. Transferul DMA se poate executa între memorie şi dispozitivele de I/O dar şi între memorie şi memorie sau între două dispozitive de I/O.

O particularitate a transferului DMA este că dialogul de acces la magistrală (cerere şi acceptare de DMA, programarea controllerului DMA cu adresele de transfer) se face în

acelaşi timp cu transferul propriu zis, deci la terminarea unui transfer DMA poate începe imediat altul.

Programarea unui canal DMA este simplă şi se face cu 4 regiştri, unul conține adrese de memorie, unul adrese pentru dispozitivul de I/O, unul numărul de cuvinte de transferat şi unul date de configurare.

O altă particularitate este faptul că în timpul transferului unitatea centrală poate lucra. Un ciclu de magistrală are 8 tacte, din care un transfer DMA ocupă doar 5. În timpul rămas liber unitatea centrală poate prelua instrucțiuni din memoria FLASH (transferul se face pe o magistrală specială IBUS) şi le poate executa.

În figura 2.11. se arată că un transfer de 32 de biți prin DMA (stânga) durează 214µs şi un transfer programat (dreapta) durează 544µs.

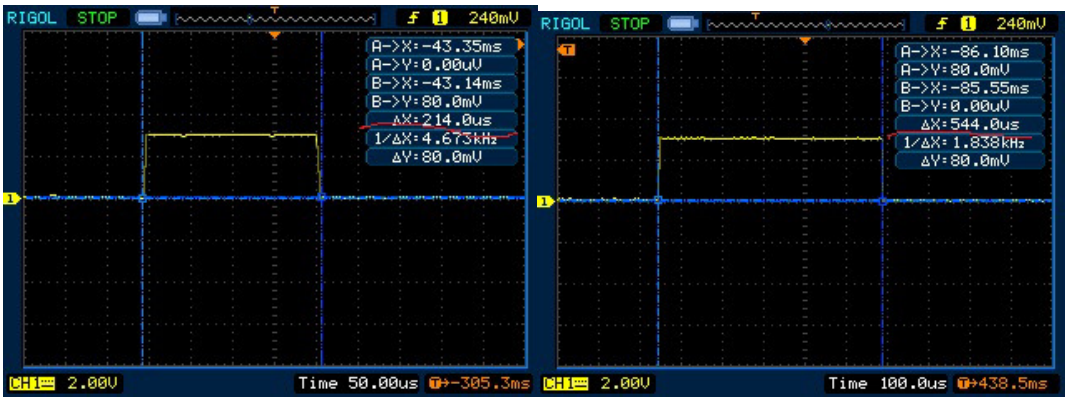


Figura 2.11. Comparație între transferul DMA şi programat (sursa: http://www.embedds.com/using-direct-memory-access-dma-in-stm23-projects/)

Rezumat

În acest modul sunt tratate aspectele fundamentale ale transferului de date între o unitate centrală, memorie şi un dispozitiv de I/O. Sunt descrise şi analizate transferul programat, prin întreruperi şi prin DMA, cu avantajele şi dezavantajele fiecăruia. Se prezintă modul de gestionare a întreruperilor şi a transferului DMA gestionate de controllere specializate din familia procesoarelor Intel x86. Sunt descrise apoi câteva particularităţi ale transferului prin întreruperi şi prin DMA la microcontrollere. Aspectele prezentate în acest modul vor fi completate în modulul de magistrale cu diagrame de timp de acces pentru fiecare mod de transfer în parte.

Bibliografie

1. P. Ogrutan, *Interfeţe şi echipamente periferice*, Editura Universitatii Transilvania Brasov, 1994, pag. 7-12
2. *A hardware interrupt tutorial*, http://www.best-microcontroller-projects.com/index.html
3. *Using Direct Memory Access (DMA) in STM32 projects*, http://www.embedds.com/using-direct-memory-access-dma-in-stm23-projects/
4. P. Borza, C. Gerigan, P. Ogrutan, Gh.Toacse Microcontrollere. Aplicatii, Editura Tehnica, Bucuresti 2000, http://vega.unitbv.ro/~ogrutan/Microcontrollere/curs.pdf

Test de autoevaluare

1.Sarcinile controllerului de întreruperi sunt:

R

- ☐ (a) să transmită spre procesor adresele de salt puse de dispozitivul de I/O
- ☐ (b) să stabilească prioritățile printr-un registru de priorități
- ☐ (c) să asigure mascarea selectivă a cererilor de întrerupere
- ☐ (d) să asigure activarea semnalului INT la sosirea unui semnal IRQ

I. Vezi pagina 8

2.Controllerul DMA trebuie programat cu:

R

- ☐ (a) un cuvânt de mascare selectivă a cererilor
- ☐ (b) numărul de linii de adresă pe care se pune adresa
- ☐ (c) cuvinte de stabilire a priorităților
- ☐ (d) intervalul de timp după care cedează magistralele procesorului

I. Vezi pagina 11

3. Avantajul transferului prin întreruperi este faptul că durează mai puțin decât cel programat

R

- ☐ adevărat
- ☐ fals

I.Vezi pagina 6

4. Cascadarea a 2 controllere DMA cu 4 canale la intrare extinde numărul de canale disponibile la 8 canale

R

☐ adevărat

☐ fals

I. Vezi pagina 10

5.De ce săgeata de adrese de la controllerul DMA din figura 2.7. este bidirecțională şi toate celelalte săgeți sunt unidirecționale?

R

☐ (a) Controllerul DMA este cel mai complex circuit

☐ (b) Controllerul DMA trebuie să pună o adresă pe magistrala de date

☐ (c) Controllerul DMA pune adrese în ciclul DMA şi este adresat în momentul programării lui

☐ (d) Dispozitivul de I/O pune adrese pentru controllerul DMA

I. Vezi pagina 10

6.Aplicații care se folosesc de faptul că transferul programat asigură controlul timpului de transfer sunt:

R

☐ (a) Scrierea pe hard disc, pentru ca scrierea unui fişier mai mic să dureze mai puțin

☐ (b) Achiziția de date din procese, la care timpul de eşantionare trebuie să fie constant

☐ (c) Trimiterea datelor către controllerul video pentru ca fiecare cadru de imagine să dureze la fel

☐ (d) La aplicații de laborator, pentru simplitatea soft şi hard

I. Utilizați cunoştințele anterioare şi gândirea proprie

7.Utilizarea întreruperilor şi a transferului DMA aduc următoarele avantaje la microcontrollere:

- ☐ (a) Transferul DMA permite unității centrale să lucreze în timpul transferului
- ☐ (b) Transferul programat prin interogare poate fi utilizat în timp ce unitate centrală serveşte o întrerupere
- ☐ (c) transferul DMA asigură o viteză mai mare de transfer
- ☐ (d) Mascarea unei întreruperi permite ca aceasta să fie executată indiferent de momentul apariției

I. Vezi paginile 13, 14, 15

Răspunsuri corecte:

1. b, c, d, pagina 8
2. a, c, pagina 11
3. fals, transferul durează la fel, pagina 6
4. fals, se extinde la 7, unul fiind ocupat cu cascadarea, pagina 10
5. c, pagina 10
6. b şi d. Dacă scrierea pe HDD nu ar fi prin întreruperi şi DMA ar dura foarte mult, iar asigurarea timpului de cadru video se face hardware printr-un circuit specializat.
7. a, c, paginile 13, 14, 15