5. PROCESORUL DE I/E I8089

5.1. Scopul lucrării

Lucrarea urmăreşte cunoaşterea arhitecturii interne a procesorului de I/E 8089, a funcționării acestuia, a modului de inițializare de către UCP, a setului de instrucțiuni şi a modului de realizare a unui sistem cu acest procesor.

5.2. Considerații teoretice

5.2.1. Arhitectura procesorului de I/E

Structura internă a procesorului de I/E (PIE) i8089 este prezentată în Figura 5.1 [15].

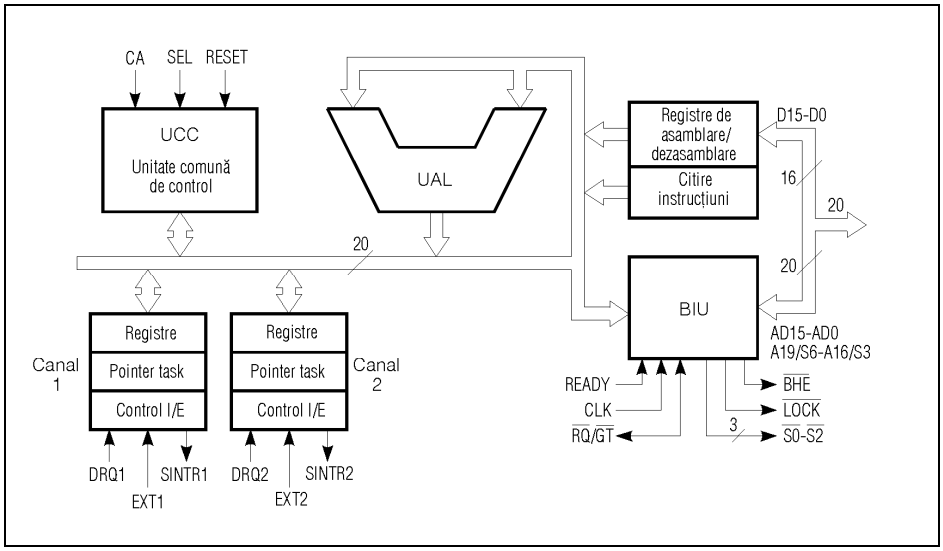

Figura 5.1. Structura internă a PIE 8089.

5.2.1.1. Unitatea comună de control (UCC)

Toate operațiile PIE (instrucțiuni, cicluri de transfer DMA, etc.) sunt compuse din secvențe de procese elementare numite *cicluri interne*. Un ciclu de magistrală necesită un ciclu intern; execuția unei instrucțiuni poate necesita mai multe cicluri interne. Există 23 de tipuri de cicluri interne, fiecare din ele necesitând 2-8 perioade de tact.

UCC coordonează activitățile PIE prin alocarea ciclurilor interne diferitelor unități ale procesorului. UCC determină deci care unitate va executa următorul ciclu intern. De exemplu, dacă ambele canale sunt active, UCC determină canalul care este mai prioritar; dacă ele au aceeaşi prioritate, programele vor fi executate în mod întrețesut. De asemenea, UCC inițializează procesorul.

5.2.1.2. Unitatea aritmetică şi logică (UAL)

Poate executa operații aritmetice binare cu numere de 8 sau 16 biți, fără semn. Rezultatele sunt reprezentate pe 20 de biți. Instrucțiunile aritmetice disponibile sunt cele de adunare, incrementare şi decrementare, iar cele logice sunt ŞI, SAU, NU.

5.2.1.3. Registrele de asamblare / dezasamblare

Toate datele sunt transferate prin intermediul acestor registre. Dacă transferul are loc între magistrale cu dimensiuni diferite, 8089 utilizează aceste registre pentru a efectua transferul într-un număr minim de cicluri de magistrală. Primul şi ultimul ciclu al unui transfer poate fi executat în mod diferit în cazul cuvintelor aflate la adrese impare.

5.2.1.4. Unitatea de citire a instrucțiunilor

Controlează citirea instrucțiunilor pentru canalul activ. Dacă pentru citirea instrucțiunilor se utilizează o magistrală de 8 biți, citirea se realizează octet cu octet, într-un ciclu de magistrală pentru fiecare. Dacă magistrala este de 16 biți, se utilizează o coadă de 1 octet pentru a reduce numărul ciclurilor de magistrală. Fiecare canal are propria coadă de 1 octet. În timpul execuției secvențiale, instrucțiunile sunt citite cuvânt cu cuvânt de la adrese pare; fiecare citire necesită un ciclu de magistrală. Dacă ultimul octet al unei instrucțiuni se află la adresă pară, primul octet al următoarei instrucțiuni, aflat la adresă impară, care a fost citit deja, este salvat în coadă. La începerea execuției următoarei instrucțiuni, primul octet va fi citit din coadă şi nu din memorie.

Dacă o instrucțiune de salt sau de apel are ca destinație o adresă impară, primul octet al instrucțiunii este citit singur, deoarece instrucțiunile de transfer invalidează conținutul cozii.

5.2.1.5. Unitatea de interfațare cu magistrala (BIU)

Controlează ciclurile de magistrală, transferând instrucțiuni şi date între PIE şi memoria sau perifericele externe. Fiecare acces la magistrală este asociat cu un *bit de marcaj al unui registru* care indică unității de interfațare dacă trebuie adresat spațiul sistem sau cel de I/E. BIU transmite tipul ciclului de magistrală pe liniile de stare $\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$. Controlerul de magistrală 8288 decodifică aceste linii şi generează semnalele necesare pentru o magistrală sau alta.

BIU face distincție între dimensiunile fizice şi cele logice ale magistralelor sistem şi de I/E.

Dimensiunile fizice ale magistralelor sunt comunicate unității BIU la inițializare.

- În *configurația locală*, ambele magistrale trebuie să aibă aceeaşi dimensiune, 8 sau 16 biți, în funcție de dimensiunea magistralei UCP.
- În *configurația la distanță*, magistrala sistem a PIE trebuie să aibă aceeaşi dimensiune fizică ca şi magistrala pe care o partajează cu UCP. Dimensiunea magistralei de I/E, locală pentru PIE, poate fi selectată independent. Dacă există periferice de 16 biți în spațiul de I/E, trebuie să se utilizeze o magistrală de 16 biți. Dacă există numai periferice de 8 biți, poate fi selectată o magistrală de 8 biți sau de 16 biți.

Dimensiunea logică a magistralei sistem şi de I/E este specificată de un program de canal pentru un anumit transfer DMA. Dimensiunea logică a unei magistrale fizice de 8 biți poate fi de numai 8 biți. O magistrală de 16 biți poate fi utilizată ca magistrală logică de 8 sau 16 biți. Aceasta permite conectarea perifericelor de 8 sau 16 biți.

Dimensiunea logică se referă numai la transferurile DMA. Instrucțiunile sunt citite pe octet sau cuvânt în funcție de dimensiunea fizică a magistralei.

BIU execută şi *arbitrajul magistralei locale*. În configurația locală, utilizează linia $\overline{RQ}/\overline{GT}$ pentru obținerea magistralei de la UCP şi pentru cedarea acesteia. În configurația la distanță, BIU utilizează linia $\overline{RQ}/\overline{GT}$ pentru a coordona utilizarea magistralei locale de I/E cu un alt PIE sau o

unitate centrală locală, dacă există. Arbitrajul magistralei sistem este realizat în acest caz de către arbitrul de magistrală 8289.

BIU activează semnalul $\overline{\text{LOCK}}$ (*Bus Lock*) în timpul execuției unei instrucțiuni TSL (*Test and Set Lock*), şi poate activa acest semnal pe durata unui transfer DMA dacă aceasta se specifică de către un program de canal.

5.2.1.6. Unitățile de control a operațiilor de I/E

Fiecare canal conține propria sa unitate de control a operațiilor de I/E.

Dacă transferul este *sincronizat*, canalul aşteaptă un semnal pe linia DRQ (*DMA Request*) înainte de a executa următoarea secvență de citire/scriere în cadrul transferului.

Dacă transferul este *terminat de un semnal extern*, canalul testează semnalul EXT1 sau EXT2 şi opreşte transferul dacă acest semnal devine activ.

Între ciclurile de citire şi scriere canalul poate contoriza, translata sau testa data transferată, şi poate termina transferul în funcție de rezultatele acestor operații.

Fiecare canal are şi o linie SINTR (*System Interrupt*) care poate fi activată prin program pentru a genera o cerere de întrerupere către UCP.

5.2.1.7. Registrele

Fiecare canal are un set propriu de registre, care sunt accesibile numai de canalul respectiv. Cele mai multe registre au roluri diferite în timpul execuției programelor de canal şi în timpul transferurilor DMA. Programele de canal trebuie să salveze aceste registre în memorie înaintea unui transfer DMA, dacă ele sunt necesare după transfer. Registrele sunt prezentate în Figura 5.2.

GA (*General Purpose* A)

Un program de canal poate utiliza acest registru ca:

- *registru general*: utilizat pentru păstrarea operanzilor instrucțiunilor de I/E
- *registru de bază*: utilizat pentru adresarea operanzilor de memorie

Înaintea unui transfer DMA acest registru se încarcă cu adresa sursă sau destinație a transferului.

GB (*General Purpose* B)

Este interschimbabil cu registrul GA. Dacă GA indică sursa unui transfer DMA, GB indică destinația, şi invers.

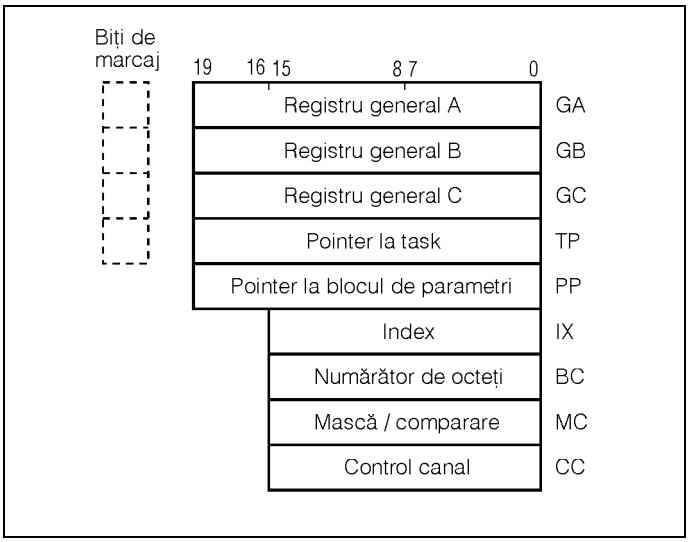

Figura 5.2. Registrele PIE 8089.

GC (*General Purpose* C)

Se poate utiliza ca registru general sau ca registru de bază de către un program de canal. La un transfer DMA, registrul GC este încărcat cu adresa primului octet al unei tabele de translatare înaintea inițierii transferului.

TP - Pointer la blocul taskului (*Task Pointer*)

Unitatea comună de control (UCC) încarcă acest registru din blocul de parametri atunci când lansează sau reia un program de canal. În timpul execuției, canalul actualizează automat registrul TP pentru a indica următoarea instrucțiune de executat, deci este utilizat ca pointer de instrucțiuni (contor de program). O procedură revine la programul apelant prin încărcarea registrului TP cu o adresă salvată anterior de către instrucțiunea CALL. Este accesibil programului de canal, care poate utiliza acest registru ca registru general sau registru de bază, deşi o asemenea utilizare nu este recomandată.

PP - Pointer la blocul de parametri (*Parameter Block Pointer*)

UCC încarcă acest registru cu adresa blocului de parametri înaintea lansării unui program de canal. Nu poate fi modificat de programul de canal, dar poate fi utilizat ca registru de bază pentru accesul la datele din blocul de parametri. Nu este utilizat în timpul unui transfer DMA.

IX (*Index*)

Poate fi utilizat ca registru de bază sau ca registru index pentru adresarea operanzilor de memorie (conținutul registrului IX este adunat cu cel al unui registru de bază). Dacă este utilizat ca registru index, conținutul lui poate fi autoincrementat (opțional). Nu este utilizat pentru transferurile DMA.

BC (*Byte Count*)

Într-un program de canal, poate fi utilizat ca registru general. Dacă un transfer DMA trebuie terminat după un număr specificat de octeți, acest număr trebuie încărcat în registrul BC înaintea transferului. BC este decrementat după fiecare octet transferat, indiferent de condiția de terminare. Dacă BC ajunge la zero, transferul este oprit numai dacă s-a indicat condiția de terminare după un număr de octeți. În caz contrar, decrementarea acestuia continuă.

MC (*Mask / Compare*)

Poate fi utilizat ca registru general într-un program de canal, sau pentru a efectua o comparație mascată cu o valoare de un octet, atât într-un program de canal, cât şi la un transfer DMA. Pentru aceasta se încarcă o valoare de comparare în octetul c.m.p.s. al registrului şi o valoare mască în octetul c.m.s. Un bit de 1 din valoarea mască selectează bitul din poziția corespunzătoare a valorii de comparare, iar un bit de 0 maschează poziția corespunzătoare.

CC (*Channel Control*)

Conținutul acestui registru controlează un transfer DMA. Programul de canal încarcă acest registru cu valoarea corespunzătoare înaintea începerii transferului. Unul din biții acestui registru (bitul 8) se referă la programul de canal. Dacă bitul 8 este 0, programul de canal rulează cu prioritatea normală. Dacă acest bit este 1, prioritatea programului de canal va fi aceeaşi cu cea a transferurilor DMA; rezultă o *execuție înlănțuită* a programului de canal.

Biții de marcaj

Registrele GA, GB, GC şi TP se numesc *registre pointeri*, deoarece se pot utiliza ca pointeri în spațiul sistem sau în spațiul de I/E. Bitul de marcaj asociat cu fiecare din aceste registre determină dacă registrul respectiv conține un *pointer în spațiul sistem* (marcajul este 0) sau un *pointer în spațiul de I/E* (marcajul este 1).

UCC setează sau şterge bitul de marcaj al registrului TP după cum primeşte de la UCP comanda *"Start program de canal în spațiul sistem"* sau *"Start program de canal în spațiul de I/E"*.

Programele de canal modifică biții de marcaj ai registrelor prin utilizarea diferitelor instrucțiuni de încărcare a registrelor. De exemplu, o instrucțiune de încărcare a unui pointer (*"Load Pointer"*) şterge bitul de marcaj, o instrucțiune *"Move"* setează bitul de marcaj, iar o instrucțiune *"Move Pointer"* transferă o valoare 0 sau 1 în bitul de marcaj.

PSW - Cuvântul de stare al programului

Fiecare canal are propriul cuvânt de stare. PSW memorează starea canalului astfel încât funcționarea canalului poate fi suspendată şi apoi reluată. Structura cuvântului de stare este următoarea:

7	6	5	4	3	2	1	0
P	XF	B	IS	IC	TB	S	D

- D — dimensiunea logică a magistralei destinație: 0 - 8 biți, 1 - 16 biți
- S — dimensiunea logică a magistralei sursă: 0 - 8 biți, 1 - 16 biți
- TB — indică prin valoarea 1 un program de canal (*Task Block*) în execuție
- IC — controlul întreruperilor: 0 - invalidate, 1 - validate
- IS — starea cererilor de întrerupere (*Interrupt Service*): 0 - SINTR inactiv, 1 - SINTR activ
- B — limita de încărcare a magistralei (*Bus Load Limit*)
- XF — indică prin valoarea 1 un transfer în curs
- P — bit de prioritate

Un program de canal poate avea prioritatea 1 sau 3, determinată de bitul de înlănțuire din registrul de control CC. Dacă acest bit este resetat, programul are prioritatea normală 3; dacă bitul este setat, programul este înlănțuit şi are aceeaşi prioritate cu un transfer DMA.

Dacă ambele canale execută activități cu aceeaşi prioritate, UCC testează biții de prioritate din cuvintele de stare. Dacă biții sunt diferiți, are prioritate canalul cu bitul P setat. Dacă şi biții de prioritate sunt egali, activitățile canalelor se vor executa alternativ. Bitul de prioritate din PSW nu are efect dacă activitățile canalelor au priorități diferite. Valoarea bitului de prioritate se încarcă dintr-un bit al cuvântului de comandă CCW; astfel UCP poate controla care canal va opera atunci când canalele au aceeaşi prioritate.

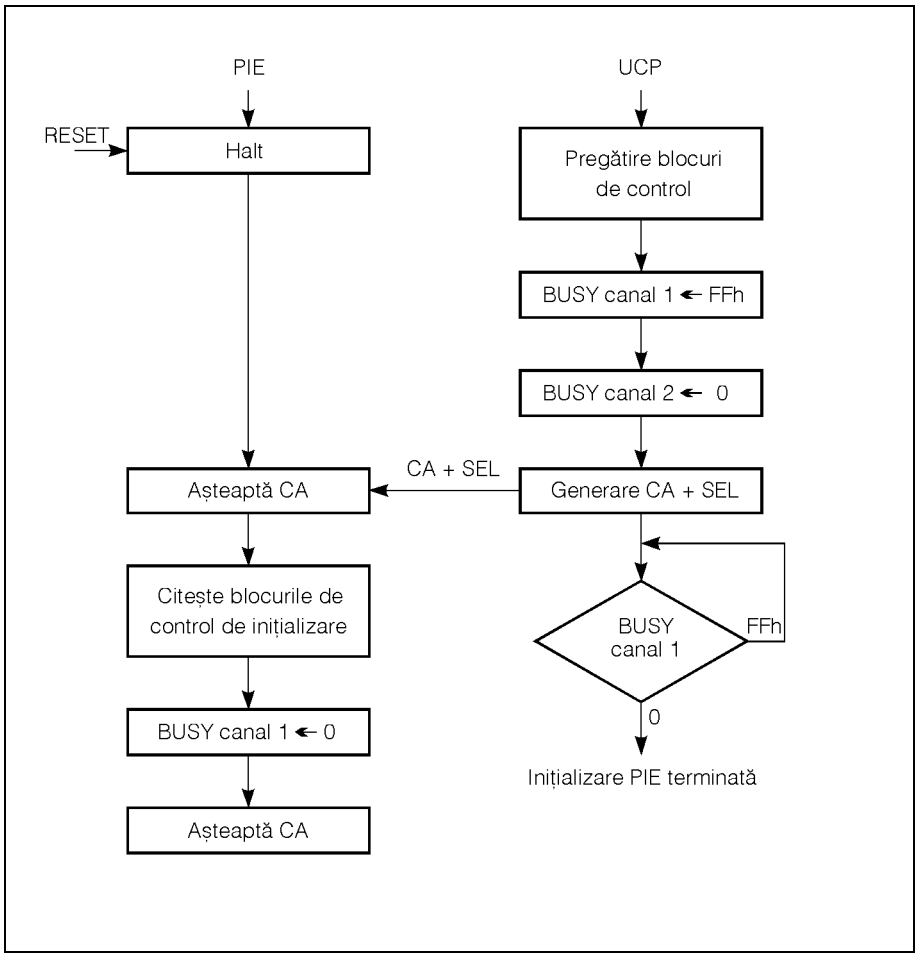

Figura 5.3. Secvența de inițializare a PIE 8089.

Dacă UCP lansează o comadă de suspendare, canalul salvează cuvântul de stare PSW, pointerul taskului TP şi bitul de marcaj al TP în primii 4 octeți ai blocului de parametri ai canalului. La recepția unei co-

menzi de reluare, se reface PSW, TP şi bitul de marcaj al TP, execuția fiind reluată.

5.2.2. Inițializarea şi comanda procesorului de I/E

Secvența de inițializare a PIE 8089 este prezentată în Figura 5.3 [15].

Secvența de inițializare începe cu activarea semnalului RESET a PIE, ceea ce opreşte orice operație în curs, dar nu afectează registrele, cu excepția bitului de înlănțuire din registrul de control CC, care este şters.

PIE aşteaptă activarea semnalului CA, după care va testa semnalul SEL. Deoarece pot exista două PIE care partajează o magistrală de I/E, unul dintre ele fiind "master", iar celălalt "slave", PIE trebuie să determine dacă este "master" sau "slave". Dacă este selectat canalul 1 (SEL = 0), PIE se va considera "master", iar dacă este selectat canalul 2 (SEL = 1), se va considera "slave".

Dacă PIE este "master", presupune că are acces la magistrală imediat. Dacă este "slave", va solicita magistrala de la UCP (în configurația locală) sau de la celălalt PIE (în configurația la distanță). Solicitarea se face printr-un impuls negativ pe linia $\overline{RQ}$ / $\overline{GT}$, iar confirmarea cedării magistralei se detectează prin apariția unui impuls pe aceeaşi linie.

După ce obține magistrala, PIE presupune că aceasta are dimensiunea de 8 biți. PIE citeşte informațiile din blocurile de control de inițializare aflate în spațiul sistem. Acestea sunt prezentate în Figura 5.5.

PIE citeşte octetul SYSBUS de la adresa FFFF6h, care indică dimensiunea fizică a magistralei sistem (Figura 5.4).

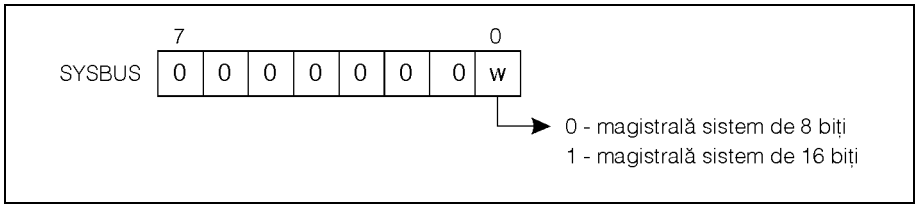

Figura 5.4. Structura octetului SYSBUS.

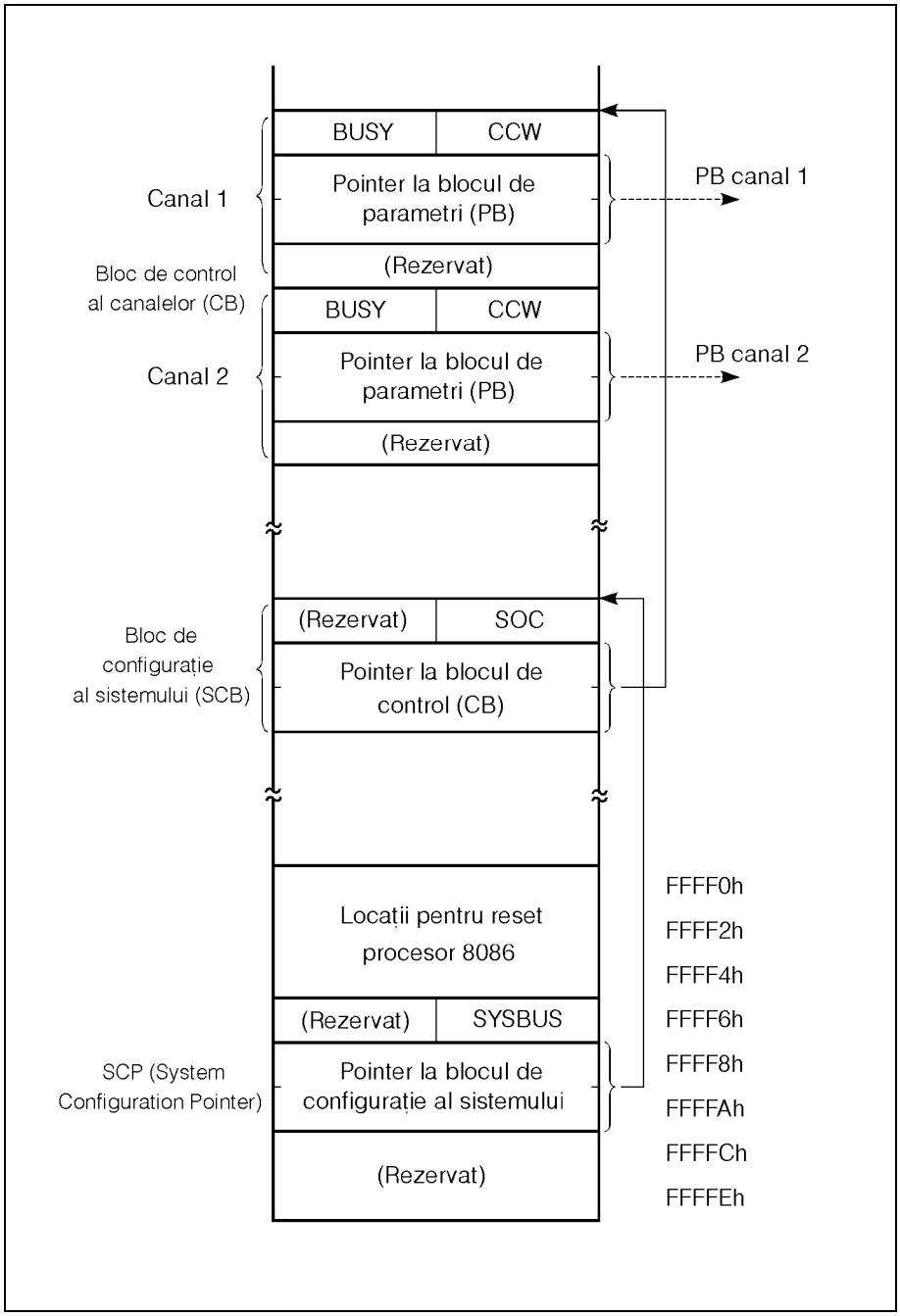

Figura 5.5. Blocurile de control din spațiul sistem.

În continuare, PIE citeşte pointerul la *blocul de configurație al sistemului* (SCB), aflat în cuvântul dublu de la adresa FFFF8h, din care construieşte o adresă fizică de 20 de biți. Având această adresă, PIE citeşte octetul SOC (*System Operation Command*). Acest octet indică modul de utilizare al liniei $\overline{RQ}/\overline{GT}$ şi dimensiunea magistralei de I/E (Figura 5.6).

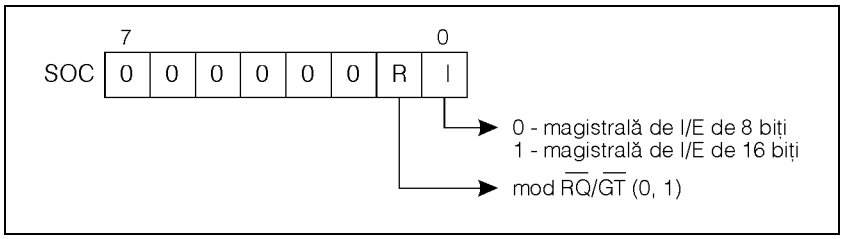

Figura 5.6. Structura octetului SOC.

Există două moduri de utilizare a liniei $\overline{RQ}/\overline{GT}$, modul 0 fiind compatibil cu cel al procesoarelor 80x86.

PIE citeşte în continuare pointerul la *blocul de control* (CB), îl converteşte într-o adresă de 20 de biți şi îl memorează într-un registru intern. Acest registru nu este accesibil programelor de canal, deci blocul de control nu poate fi mutat fără reinițializarea PIE. Indicatorul BUSY este setat, celelalte câmpuri din blocul de control CB fiind utilizate la activarea unui canal.

Unitatea centrală, după pregătirea blocurilor de control în memorie şi selecția canalului (s-a presupus aici canalul 1), testează indicatorul BUSY al canalului pentru a determina dacă secvența de inițializare s-a terminat. Pentru aceasta, indicatorul BUSY al canalului 1 este setat în prealabil la FFh.

După inițializare, orice semnal CA este interpretat ca o comandă. La recunoaşterea semnalului CA, canalul setează indicatorul BUSY din blocul de control CB la FFh, citeşte cuvântul de comandă CCW din CB şi execută comanda din câmpul de comandă. Structura cuvântului de comandă este prezentată în Figura 5.7.

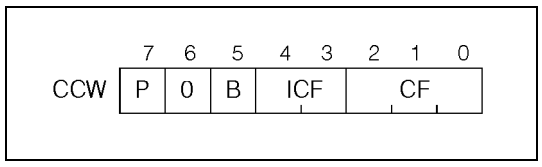

Figura 5.7. Structura cuvântului de comandă.

CF câmp de comandă (*Command Field*)

- 000 actualizare PSW
- 001 start program de canal din spațiul de I/E
- 010 (rezervat)
- 011 start program de canal din spațiul sistem
- 100 rezervat
- 101 reluare operație suspendată
- 110 suspendare operație de canal
- 111 oprire

ICF câmp de control al întreruperilor (*Interrupt Control Field*)

- 00 nici un efect
- 01 elimină cererea de întrerupere; cererea este recunoscută
- 10 validare întreruperi
- 11 invalidare întreruperi

B limita de încărcare a magistralei (*Bus Load Limit*)

- 0 fără limită de încărcare
- 1 cu limită de încărcare

P bit de prioritate

Un PIE în configurația locală are o prioritate mai mare decât UCP, deoarece magistrala va fi cedată la cererea PIE. Unul sau două PIE pot monopoliza magistrala, ceea ce este normal dacă operațiile PIE sunt critice. Există însă şi programe de canal cu prioritate mai redusă, care necesită performanțe mai reduse.

În asemenea cazuri, UCP poate seta un bit al cuvântului de comandă CCW care indică limita de încărcare a magistralei, pentru a reduce utilizarea magistralei de către programele de canal normale (neînlănțuite). Dacă bitul B este setat, canalul decrementează un contor de la 127 la 0 la fiecare instrucțiune executată, la fiecare perioadă de ceas. Deci, canalul va aştepta cel puțin 127 de cicluri înaintea executării următoarei instrucțiuni.

Cele două comenzi de "start program" diferă prin modul în care afectează bitul de marcaj al TP. Pointerul la blocul taskului se încarcă în TP, se actualizează PSW după cum se specifică în câmpurile ICF, B şi P ale CCW şi se lansează programul.

5.2.3. Setul de instrucțiuni 8089

Setul de instrucțiuni nu face diferențiere între adresele de memorie şi cele ale dispozitivelor de I/E. Instrucțiunile care acceptă operanzi de memorie pot fi utilizate şi pentru comunicarea cu dispozitivele de I/E.

5.2.3.1. Instrucțiuni de transfer

- `MOV` *`destinație, sursă`*

  Există patru instrucțiuni `MOV` diferite, cu mnemonicele:

`MOV`	(cuvânt)
`MOVB`	(octet)
`MOVI`	(cuvânt imediat)
`MOVBI`	(octet imediat)

  Dacă destinația este un registru pointer, bitul său de marcaj este setat. Aceste instrucțiuni se pot utiliza deci pentru încărcarea adreselor spațiului de I/E în registre pointeri.

  La transferul unui octet sau cuvânt într-un registru pointer (de 20 biți), se extinde semnul pe pozițiile neutilizate (12 biți, respectiv 4 biți).

  Exemple

  ```
  MOV   [GA],[GB]
  MOVB  BC,[PP].contor
  MOVI  BC,0
  MOVBI MC,'A'
  ```

- `MOVP` *`destinație, sursă`* (*Move Pointer*)

  Transferă o variabilă de tip adresă fizică între un registru pointer şi memorie. Dacă sursa este un registru pointer, conținutul acestuia şi bitul de marcaj este convertit într-un pointer de adresă fizică. Dacă sursa este memoria, cei 3 octeți sunt convertiți într-o adresă fizică de 20 biți şi o valoare de marcaj.

  Exemple

  ```
  MOVP  TP,[GC+IX]
  MOVP  [GB].ADR1,GC
  ```

- LPD *destinație, sursă* (*Load Pointer with Doubleword*)

Converteşte un cuvânt dublu într-o adresă fizică şi o încarcă într-un registru pointer. Bitul de marcaj al registrului pointer este şters, indicând o adresă din spațiul sistem.

LPD	încarcă o variabilă de tip cuvânt dublu
LPDI	încarcă o valoare imediată

Exemple

```
LPD   GA,[PP].adr_start
LPDI  GB,adr_perif
```

5.2.3.2. Instrucțiuni aritmetice

Operanzii sunt interpretați ca numere fără semn. Nu se detectează depăşirea de capacitate.

- ADD *destinație, sursă*

ADD	(adunare cuvânt)
ADDB	(adunare octet)
ADDI	(adunare cuvânt imediat)
ADDBI	(adunare octet imediat)

Exemple

```
ADD   [GB],GC
ADDBI MC,10
```

- INC *destinație*

INC	(incrementare cuvânt)
INCB	(incrementare octet)

Exemplu

```
INC   [GB].pointer
```

- DEC *destinație*

DEC	(decrementare cuvânt)
DECB	(decrementare octet)

Exemplu

```
DEC   [GA+IX].cont_b
```

5.2.3.3. Instrucțiuni logice şi pe biți

Cei 4 biți mai semnificativi ai unui registru destinație de 20 biți vor fi nedefiniți după o operație logică. Dacă destinația unei operații pe octet este un registru, biții 8-15 sunt rescrişi cu bitul 7 al rezultatului. Biții 8-15 pot fi păstrați după o instrucțiune AND sau OR prin utilizarea operațiilor pe cuvânt în care octetul c.m.s. al operandului sursă este FFh, respectiv 00h.

- AND *destinație, sursă*

```
AND
ANDB
ANDI
ANDBI
```

Exemple

```
AND   [GC].stare,MC
ANDBI [GC+IX],0Fh
```

- OR *destinație, sursă*

```
OR
ORB
ORI
ORBI
```

Exemple

```
OR    MC,[GC].masca
ORBI  [GB].comanda,0Ch
```

- NOT *destinație*
- NOT *destinație,sursă*

Complementează biții unui operand. Dacă există un singur operand, rezultatul înlocuieşte valoarea inițială. Dacă sunt doi operanzi, biții complementați ai sursei înlocuiesc valoarea operandului destinație, care trebuie să fie un registru.

```
NOT
NOTB
```

Exemple

```
NOT   MC
NOT   [GA]
NOTB  IX,[GB].stare
```

- SETB *destinație, bit*

Este setat bitul specificat al destinației, care trebuie să fie un octet de memorie.

Exemplu

```
SETB  [GA].reg_param,3
```

- CLR *destinație, bit*

Bitul selectat este şters.

Exemplu

```
CLR   [GA],0
```

5.2.3.4. Instrucțiuni de salt şi apel

Modifică conținutul registrului TP prin adunarea unui deplasament cu semn, de 8 sau 16 biți, reprezentat în C2.

Dacă numele instrucțiunii începe cu litera 'L', deplasamentul va fi de 16 biți. În caz contrar, asamblorul generează un deplasament de 8 sau 16 biți, în funcție de context.

- CALL/LCALL *salvare_TP,adr*

Salvează registrul TP şi bitul său de marcaj în operandul *salvare_TP* (o variabilă de tip adresă fizică) şi efectuează saltul la adresa destinație formată prin adunarea *adr* la conținutul registrului TP. Reîntoarcerea la instrucțiunea următoare instrucțiunii CALL se poate face prin utilizarea une instrucțiuni MOVP pentru a încărca TP cu operandul *salvare_TP*.

O stivă poate fi implementată utilizând ca indicator de stivă un registru de bază.

Exemplu

```
LCALL [GC].save,init
```

- JMP/LJMP *adr*

Se efectuează un salt necondiționat la adresa *adr*.

- JZ/LJZ *sursă,adr*

Se efectuează saltul dacă operandul *sursă* este zero.

```
JZ/LJZ      (salt dacă cuvântul este zero)
JZB/LJZB    (salt dacă octetul este zero)
```

Exemple

```
JZ    BC,write
LJZ   [GB].contor,term
```

- JNZ/LJNZ *sursă,adr*

  Se efectuează saltul dacă operandul *sursă* este diferit de zero.

```
JNZ/LJNZ    (salt dacă cuvântul este diferit de zero)
JNZB/LJNZB  (salt dacă octetul este diferit de zero)
```

- JMCE/LJMCE *sursă,adr* (*Jump if Masked Compare Equal*)

Se efectuează saltul dacă sursa (un octet de memorie) este egală cu octetul c.m.p.s. al registrului MC mascat cu octetul c.m.s. al registrului MC. Se utilizează pentru testarea mai multor biți ai registrelor de 8 biți.

Exemplu

```
JMCE  [GB],oct_gasit
```

- JMCNE/LJMCNE *sursă,adr* (*Jump if Masked Compare Not Equal*)

  Instrucțiuni similare cu JMCE/LJMCE.

- JBT/LJBT *sursă,bit,adr* (*Jump if Bit True*)

Testează bitul indicat al operandului *sursă* şi efectuează saltul dacă bitul este 1. Sursa trebuie să fie un octet de memorie (sau un registru al unui dispozitiv). Dacă saltul este la aceeaşi instrucțiune JBT, se aşteaptă până când bitul devine 0.

Exemplu

```
LJBT  [GA].rezultat,1,eroare
```

- JNBT/LJNBT *sursă,bit,adr* (*Jump if Not Bit True*)

Instrucțiuni similare cu JBT/LJBT, dar saltul se efectuează dacă bitul este 0.

Exemplu

```
LJNBT [GC],6,ok
```

5.2.3.5. Instrucțiuni de control ale procesorului

- TSL *destinație,valoare,adr* (*Test and Set while Locked*)

  Execuția acestei instrucțiuni este prezentată în Figura 5.8.

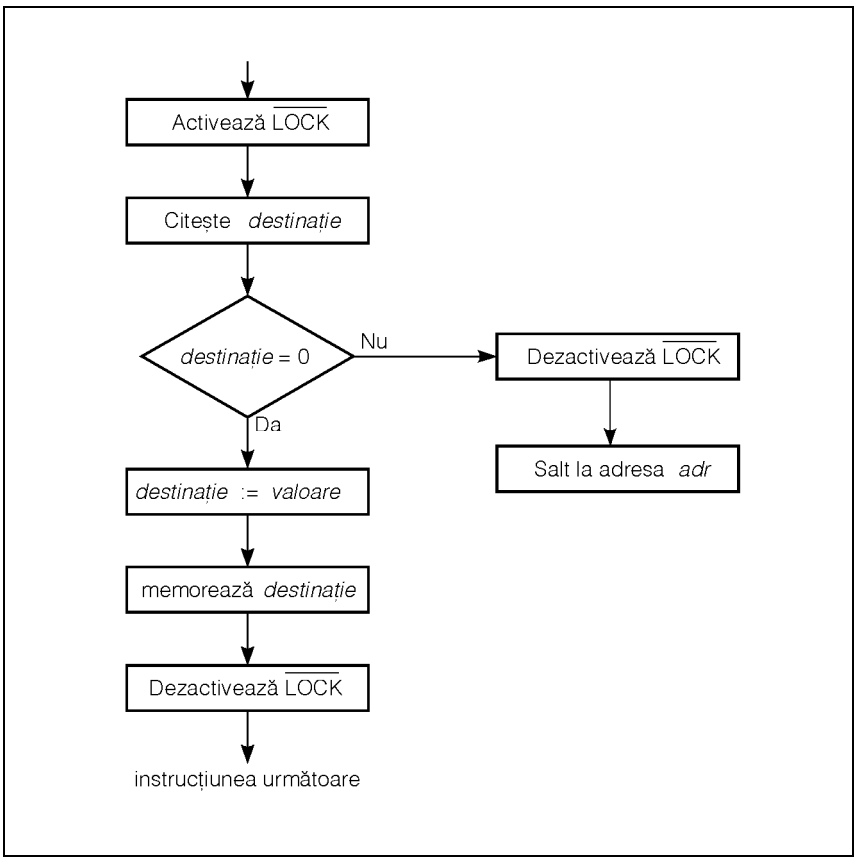

Figura 5.8. Execuția instrucțiunii TSL.

Instrucțiunea TSL se poate utiliza pentru implementarea unei variabile de tip semafor care controlează accesul la o resursă partajată dintr-un sistem multiprocesor. Dacă *adr* specifică adresa instrucțiunii TSL, această instrucțiune este executată repetat până când semaforul (destinația) devine 0 (resursa devine liberă).

Exemplu

```
TSL   [GA].semafor,0FFh,not_ready
```

- `WID` *`dim_sursă,dim_destinație`* (*Set Logical Bus Widths*)

Modifică biții 0 şi 1 din PSW, specificând dimensiunile logice ale magistralelor pentru un transfer DMA. Operanzii pot specifica 8 sau 16 biți, cu restricția ca dimensiunea logică să nu depăşească dimensiunea fizică.

Exemplu

```
WID   8,16
```

- `XFER`

Pregăteşte canalul pentru un transfer DMA, care va începe după terminarea instrucțiunii următoare. Aceasta poate fi orice instrucțiune, cu excepția uneia care modifică registrele GA, GB sau BC. Pentru un transfer sincronizat, această instrucțiune poate transmite o comandă de 'start' sau ultimul dintr-o serie de parametri.

- `SINTR`

Setează bitul IS (*Interrupt Service*) din PSW şi activează linia SINTR a canalului, dacă bitul de validare a întreruperilor din PSW este setat. În caz contrar setează numai bitul IS.

- `NOP`

Nici o operație

- `HLT`

Termină un program de canal. Canalul şterge indicatorul BUSY şi devine inactiv.

5.2.4. Pregătirea unui transfer DMA

Un program de canal pregăteşte un transfer DMA în două etape: pregătirea controlerului şi pregătirea canalului [15].

5.2.4.1. Pregătirea controlerului de dispozitiv

Controlerele care execută transferuri DMA pot efectua diferite tipuri de operații. De exemplu, un controler de disc poate citi/scrie un sector, poate căuta o pistă etc. Registrele controlerului sunt vizibile de către programul de canal ca o serie de locații de memorie. Adresa de bază a

dispozitivului poate fi plasată într-un registru pointer, comunicarea cu aceste registre realizându-se prin operații de I/E programate.

Anumite controlere încep transferul DMA imediat după recepționarea ultimului parametru. În acest caz, instrucțiunea programului de canal care transmite ultimul parametru trebuie să urmeze după instrucțiunea XFER. Această instrucțiune plasează canalul în modul DMA după următoarea instrucțiune.

5.2.4.2. Pregătirea canalului

Informațiile care descriu operația trebuie transferate în registrele de canal. Aceste informații sunt descrise în continuare.

Pointerii sursă şi destinație. Pot fi plasați în registrele GA sau GB. Un bit al registrului de control trebuie poziționat pentru a indica pointerul sursă.

Pointerul tabelei de translație. Dacă datele trebuie translatate pe măsură ce sunt transferate, registrul GC trebuie încărcat cu adresa primului octet al unei tabele de translație de 256 octeți. Tabela poate fi amplasată în spațiul sistem sau cel de I/E, iar registrul GC trebuie încărcat cu o instrucțiune care setează sau resetează bitul său de marcaj după cum este necesar.

Operația de translatare este definită numai pentru datele de tip octet. Dimensiunile logice ale magistralelor sursă şi destinație trebuie să fie setate la 8 biți. Octetul de transferat este considerat ca un număr fără semn. Acest număr este adunat la conținutul registrului GC pentru a forma o adresă de memorie (numărul este considerat ca un deplasament în tabela de translație). Octetul de la această adresă este încărcat din memorie, înlocuind octetul sursă.

Contorul de octeți. Dacă transferul trebuie terminat după un anumit număr de octeți, acest număr trebuie încărcat în registrul BC. Canalul decrementează acest registru pe măsura transferului, indiferent dacă se utilizează sau nu această condiție de terminare.

Valori mască şi de comparare. Dacă transferul trebuie terminat atunci când un octet (eventual translatat) devine egal sau neegal cu o valoare de căutare, registrul MC trebuie încărcat cu valoarea măştii (în octetul c.m.s.) şi valoarea de comparare (în octetul c.m.p.s.). Registrul MC nu este modificat în timpul transferului.

Dimensiunea logică a magistralei. Se utilizează instrucțiunea `WID` pentru a seta dimensiunea logică a magistralelor sursă şi destinație. Dimensiunile logice sunt independente pentru fiecare canal.

Pentru magistrale cu dimensiunea fizică de 8 biți, dimensiunea logică poate fi numai de 8 biți. Pentru magistrale de 16 biți, dimensiunea logică trebuie selectată conform cu perifericul, de 8 sau 16 biți. Transferurile cu memoria de 16 biți se execută la viteza maximă dacă dimensiunea logică este de 16 biți.

Dimensiunea logică trebuie setată la 8 biți dacă:

- data trebuie translatată;
- data trebuie comparată printr-o mască, iar memoria de 16 biți este destinație a transferului.

Instrucțiunea `WID` setează ambele dimensiuni logice, care rămân setate până la următoarea instrucțiune `WID`. Această instrucțiune trebuie utilizată cel puțin o dată după inițializarea procesorului (prin semnalul RESET).

Registrul de control al canalului. Biții registrului CC specifică modul de execuție al transferului DMA (Figura 5.9).

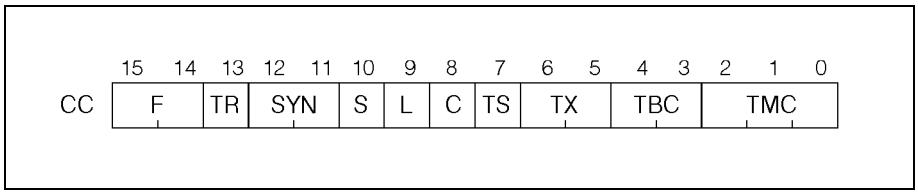

Figura 5.9. Structura registrului de control al canalului CC.

Câmpurile acestui registru sunt descrise în continuare.

- **F** (*Function*)

Identifică sursa sau destinația ca memorie sau un port de I/E. Canalul incrementează registrele pointeri sursă şi destinație care se referă la memorie. Pointerii care se referă la dispozitivele de I/E rămân constanți. Valorile acestui câmp pot fi:

00	transfer port → port
01	transfer memorie → port
10	transfer port → memorie
11	transfer memorie → memorie

- **TR** (*Translate*)

0	fără translatare
1	cu translatare

- **SYN** (*Synchronization*)

  Specifică modul de sincronizare al transferului:

  00 fără sincronizare
  01 sincronizare la sursă
  10 sincronizare la destinație
  11 (rezervat)

Transferurile nesincronizate sunt utilizate de obicei între zone de memorie. Următorul ciclu de transfer începe imediat după terminarea ciclului curent. Memoriile lente pot extinde ciclurile de magistrală cu ajutorul circuitului 8284, care va insera stări de aşteptare.

Sincronizarea la sursă este selectată în mod tipic atunci când sursa este un dispozitiv de I/E iar destinația este memoria. Dispozitivul de I/E lansează următorul ciclu de transfer prin activarea semnalului DRQ (DMA *Request*).

Sincronizarea la destinație este utilizată de obicei dacă sursa este memoria şi destinația este un dispozitiv de I/E. Atunci când este gata să primească următorul octet sau cuvânt, dispozitivul activează semnalul DRQ.

- **S** (*Source*)

Identifică registrul GA sau GB ca pointer sursă (şi celălalt registru ca pointer destinație):

0 GA este pointerul sursă
1 GB este pointerul sursă

- **L** (*Lock*)

Se utilizează pentru activarea semnalului $\overline{\text{LOCK}}$ al magistralei în timpul transferului.

- **C** (*Chain*)

Nu este utilizat pentru un transfer DMA. Se utilizează pentru a creşte prioritatea programului de canal.

- **TS** (*Terminate on Single Transfer*)

Se utilizează pentru a se executa un singur ciclu de transfer (un octet sau un cuvânt), reluând apoi programul de canal. În acest caz, orice altă condiție de terminare este ignorată. Un exemplu de utilizare este pentru dispozitive lente, ca tastaturi sau linii de comunicație, pentru translatarea şi/sau compararea octetului transferat.

Ultimele trei câmpuri din registrul CC indică condițiile de terminare a transferului, presupunând că nu a fost selectat transferul singular. Se pot specifica trei condiții de terminare şi combinații ale acestora.

- **TX** (*Terminate on External Signal*)

  00 fără terminare externă
  01 terminare la semnalul extern activ, offset = 0
  10 terminare la semnalul extern activ, offset = 4
  11 terminare la semnalul extern activ, offset = 8

- **TBC** (*Terminate on Byte Count*)

  00 fără terminare după un număr de octeți
  01 terminare dacă BC = 0, offset = 0
  10 terminare dacă BC = 0, offset = 4
  11 terminare dacă BC = 0, offset = 8

- **TMC** (*Terminate on Masked Compare*)

  000 fără terminare la comparare mascată
  001 terminare la identitate, offset = 0
  010 terminare la identitate, offset = 4
  011 terminare la identitate, offset = 8
  100 fără efect
  101 terminare la diferență, offset = 0
  110 terminare la diferență, offset = 4
  111 terminare la diferență, offset = 8

La terminarea unui transfer DMA, canalul adună o valoare numită *offset de terminare* la conținutul registrului TP şi reia execuția programului de canal de la acest punct. Terminarea la un transfer singular are întotdeauna valoarea offsetului egală cu 0.

Offseturile de terminare se pot utiliza ca indici într-o tabelă de salturi pentru identificarea condiției de terminare. Este posibil să apară simultan două sau trei condiții de terminare. În acest caz, canalul indică terminarea rezultată din condiția de terminare cu offsetul având valoarea cea mai mare.

5.3. Desfăşurarea lucrării

5.3.1. Se consideră următoarea configurație a unui sistem:

- Unitatea centrală este un procesor 8086, cu o magistrală sistem de 16 biți, conectat în modul maxim.
- Există un singur procesor de I/E 8089, conectat în configurația la distanță.

- Magistrala locală de I/E este de 8 biți.
- Programele de canal se află în spațiul sistem.
- Pentru activarea semnalelor CA şi SEL se utilizează două porturi din spațiul de I/E al procesorului 8086, cu adresele FCh (canalul 1) şi FDh (canalul 2).
- Se presupune că pointerul la blocul de configurație al sistemului şi octetul SYSBUS se află în memoria de tip ROM. Deci pointerul este fix, citindu-se valoarea acestuia de la adresa FFFF8h pentru a putea inițializa blocul de configurație SCB şi celelalte blocuri, care se presupun amplasate în memoria RAM.

  Se va desena circuitul pentru generarea semnalelor CA şi SEL.

Se va scrie un program pentru inițializarea procesorului de I/E 8089 pentru configurația de mai sus. Canalul 1 va controla afişarea unor mesaje pe ecran, iar canalul 2 va controla citirea unor comenzi de la tastatură.

5.3.2. Se va scrie un program de canal pentru transferul între două blocuri de memorie din spațiul sistem, de maxim 64 KB. Se presupune o magistrală sistem de 16 biți. Pentru o viteză maximă de transfer, programul va bloca magistrala pe timpul fiecărui ciclu de transfer. Transferul se va termina după un număr specificat de octeți.

5.3.3. Se va realiza o schemă pentru un sistem cu un procesor 8086 şi un procesor de I/E 8089 în configurația locală. Cele două procesoare partajează magistrala sistem, circuitele "latch" de adrese, amplificatoarele bidirecționale de date şi controlerul de magistrală 8288. Se va figura şi modul de conectare al unor periferice la sistem.