2.MICROCONTROLLERE PE 8 BIȚI (F^2MC-8FX)

2.1.Regiştrii, maparea memoriei, adresarea, instrucțiuni şi comportarea la RESET

Spațiul de adresare este de 64K şi este divizat în spațiu I/O, de date şi de program. Regiştrii MC sunt de două tipuri: regiştrii speciali şi regiştrii de uz general.

Regiştrii speciali

Sunt 7 regiştrii de 16 biți, astfel:

Acumulatorul A , folosit la stocarea temporară a datelor. Pentru operații de transfer se folosesc cei 8 biți mai puțin semnificativi.

Acumulator temporar T, folosit la operații între regiştrii acumulator. Când se execută o instrucțiune MOV, valoarea anterioară a acumulatorului A se transferă automat în T.

Registru de index IX, este folosit împreună cu un octet de offsset (PC-128 la PC+128)(PC-Program Counter).

Registru extrapointer EP, pentru a indica o adresă de memorie la un tip de adresare (EP).

Registru indicator de stivă SP, indică poziția (adresa) curentă a stivei în memorie.

Registru Program Counter PC indică adresa de program care se execută de CPU. Valoarea după RESET este FFFDH.

Registru de stare a programului PS este format din registrul de bankuri (Register Bank Pointer RP), registru indicator al paginii (Direct Bank Pointer DP) şi registrul de flaguri (Condition Code Register CCR) astfel:

	15 — 11	10 — 8	7 — 0 Bit (LSB)
PS	RP	DP	CCR

RP şi DP se folosesc pentru a avea acces la regiştrii de uz general. Cei 5 biți mai semnificativi (RP) reprezintă adresa bankului de regiştrii, iar ceilalți 3 biți (DP)se folosesc la adresarea directă pentru selecția registrului în bank. Există 32 de bankuri de registre de uz general cu adresa 0-31 specificată de RP (pe 5 biți). Fiecare bank are 8 registre de 8 biți selectate de DP (0-7).

Structura CCR:

CCR	H	I	IL1	IL0	N	Z	V	C

H =1 dacă apare transport între biții 3 şi 4 ai octetului de date

N =1 dacă rezultatul unei operații este negativ (primul bit este 1)

V =1 indică o depăşire (overflow)

C =1 dacă apare transport la bitul 7 (MSB) ca rezultat al unei operații (CARRY)

I =1 validare întreruperi

IL0, IL1 arată nivelul mascării întreruperii. O cerere de întrerupere este acceptată doar dacă nivelul ei este mai mare decât cel indicat. IL1 =1 şi IL0 =1 înseamnă invalidarea întreruperilor.

Regiştrii de uz general

Sunt situaţi (mapaţi) în memorie între adresele 0100H şi 0200H. Au adresa specificată de RP şi DP.

RP 00000B		**RP 00001B**	……..
Registru de uz general, DP (binar)		Registru de uz general, DP (binar) ……	
R0	000	R0	000
R1	001	R1	001
R2	010	R2	010
……..		……..	
R7	111	R7	111

Exemplu de utilizare: instrucţiunea MOV A,R6

R6 are DP=110B. RP a fost iniţializat cu 01000B. RP şi DP formează 46H. Octetul mai semnificativ al adresei este 01H din cauza mapării regiştrilor începând cu 0100H.

Rezultatul instrucţiunii este: conţinutul registrului R6 aflat la adresa 0146H este mutat în acumulator.

Adresarea

1.Adresarea directă (dir)[1]

MOV 32H,A conţinutul A este mutat la adresa 0032H. Zona de adresare este 0000H-047FH. (La operaţii de transfer se foloseşte implicit A partea Low)

2.Adresare extinsă (ext)

MOVW A,1234H conţinutul adresei 1234 se încarcă în AH şi conţinutul adresei 1235 se încarcă în AL. Zona de adresare este 0000H-FFFFH.

3.Adresare directă pe bit (dir:b)

SETB 34H:2 se setează bitul 2 al octetului de la adresa 0034H

4.Adresare indexată (@IX+off)

MOVW A,@IX+5 IX=1234H, IX+5=1239H, conţinutul adresei 1239H se încarcă în AH, conţinutul adresei 123AH se încarcă în AL

5.Adresare cu registrul EP (Extra Pointer) (@EP)

MOVW A,@EP EP=1234H, conţinutul adresei 1234H se încarcă în AH, conţinutul adresei 1235H se încarcă în AL

6.Adresare cu regiştrii de uz general (Ri), a fost exemplificată anterior

7.Adresare imediată (#imm)

MOV A,#56H acumulatorul (AL) se încarcă cu 56H

MOVW A,#1256H acumulatorul se încarcă cu 1256H

8.Adresare vectorizată (#k), cu specificarea adresei în tabela de vectori

CALLV #5 se apelează subrutina nr. 5 din tabela de vectori

9.Adresare relativă (rel), în acest mod de adresare în zona între PC+128 şi PC-128 se specifică o valoare care se adună cu semn la PC (Program Counter)

Instrucţiuni speciale

Microcontrollerul are instrucţiuni de multiplicare /divizare.

[1] Prescurtările din paranteze sunt aceleaşi cu cele din foile de catalog ale microcontrollerului.

MULU A multiplică fără semn AL cu TL şi stochează rezultatul în A (16 biţi). Nu sunt afectaţi biţii de stare

DIVU A divizează valoarea din T cu valoarea din A şi stochează rezultatul în A şi restul în T.

Observaţie: setarea sau resetarea pe bit SETB şi CLRB operează cu octeţi prin citire, poziţionarea de bit şi rescriere. Prin urmare nu se pot face astfel de poziţionări dacă unii biţi sunt Read Only sau Write Only.

Maparea memoriei

Este figurată în figura 2.1:

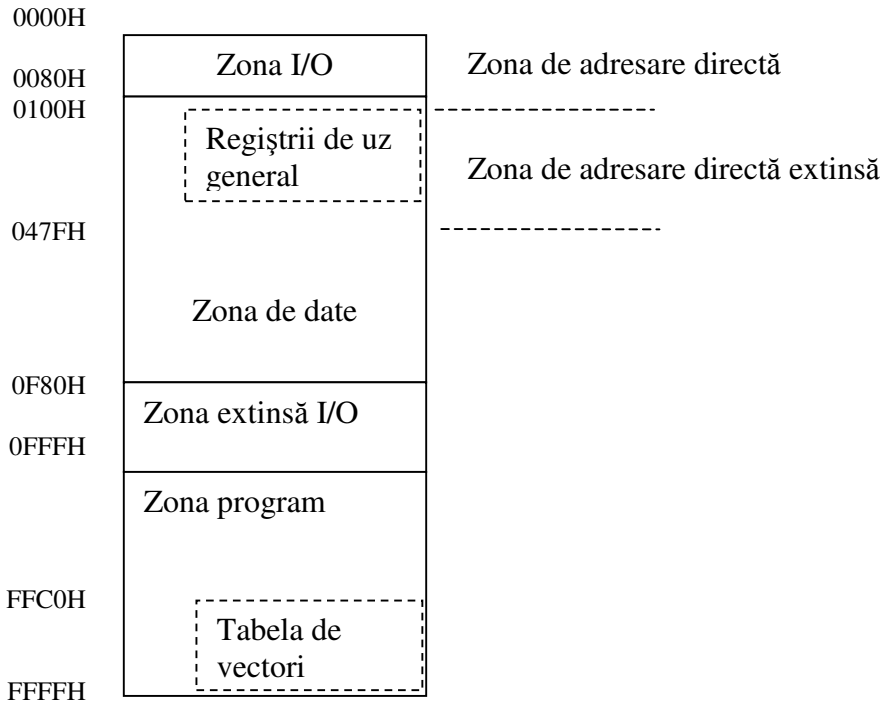


Figura 2.1: maparea memoriei

Zona regiştrilor de uz general este între 0100 şi 01FFH. Zona de adresare directă se află între adresele 0000H şi 00FFH. Această zonă este cel mai des accesată, accesul fiind realizat de instrucţiuni de numai 2 octeţi. Zona conţine şi selecţia prin adresare a interfeţelor de I/O.

Zona de program conţine tabel instrucţiunilor de salt şi tabela vectorilor de RESET şi întrerupere. Zona între FFC0H şi FFFFH este folosită ca tabelă a instrucţiunilor de salt folosite de adresarea vectorizată (CALLV #). Adesele de salt se introduc în tabelă conform unei corespondenţe:

CALLV #_	**Adresa de salt**	
	Octet superior	**Octet inferior**
#0	FFC0H	FFC1H
#1	FFC2H	FFC3H
----------	----------	----------
#6 (IRQ23)	FFCCH	FFCDH
#7 (IRQ22)	FFCEH	FFCFH

Zona între FFD0H şi FFFFH este folosită ca tabelă de salt la întreruperi şi RESET:

Întrerupere IRQ	Adresa de salt	
	Octet superior	Octet inferior
RESET	FFFEH	FFFFH
MOD	FFFCH	FFFDH
IRQ0	FFFAH	FFFBH
IRQ1	FFF8H	FFF9H
----------	----------	----------
IRQ21	FFD0H	FFD1H

Comportarea la RESET

La adresa FFFDH se stochează 00H, ceea ce înseamnă mod *single chip*. Alte variante sunt rezervate pentru dezvoltări ulterioare.

Pinul exterior MOD trebuie conectat la masă, ceea ce înseamnă citirea vectorului de RESET din memorie. În urma RESET-ului PC se încarcă cu valorile de la adresele FFFFH (octet mai puțin semnificativ) şi FFFEH. Regiştrii A, T, SP, EP, IX, regiştrii de uz general şi de fapt toată memoria RAM îşi păstrează valorile. În CCR I devine 0 şi biții IL devin 11B, ceea ce înseamnă invalidarea întreruperilor.

Observații:

1.Microcontrollerul poate executa transferuri de date şi operații aritmetice pe 16 biți, tratând cuvintele pe 16 biți ca 2 octeți, salvând octetul mai semnificativ la adresa specificată de instrucțiune, iar octetul mai puțin semnificativ la următoarea adresă (mai mare).

MOVW 0500H,A salvează A la adresa 0500H

Dacă A=1234H, 12H (AH) se salvează la adresa 0500H iar 34H (AL) la adresa 0501H

2.Importanța acumulatorului temporar este evidențiată de următorul exemplu:

MOVW A,1200H Conținutul adresei 1200H intră în AH iar conținutul adresei 1201H în AL.

MOVW A, 0040H Conținutul acumulatorului se transferă în T (AH în TH, AL în TL). Conținutul adresei 0040H intră în AH iar 0041 în AL.

ADDCW A Adunarea cu transport se efectuează între A şi T cu rezultatul în A.

Fără acumulatorul temporar, pentru a efectua această adunare era nevoie de utilizarea a încă unui registru.

2.2.Funcționarea în întreruperi

Sunt posibile 24 de cereri de înterupere cu prioritate programabilă. Prioritatea întreruperilor se programează în 6 registre de 8 biți ILR0,1,2,3,4 şi 5 în care la fiecare cerere de întrerupere i se ataşează 2 biți. Astfel prioritatea poate fi 0,1,2 sau 3. Nivelul de prioritate al cererii este comparat cu nivelul de mascare memorat în registrul CCR, biții IL0 şi IL1 şi cererea de întrerupere poate fi acceptată sau nu. Cererile pot fi invalidate global de bitul I din CCR.

Succesiunea de operații la primirea unei întreruperi:

- După RESET toate cererile sunt invalidate. În programul de inițializare a interfețelor se inițializează interfețele cu care se lucrează şi se programează biții corespunzători din registrele ILR0,1,2,3,4 pentru ataşarea priorității.Nivelul 0 este corespunzător priorității maxime. Nivelul 3 înseamnă invalidarea întreruperii.
- Se execută programul principal.

- Interfața (sau o sursă de întreruperi externă) cere o întrerupere, care dacă este validată de un flag propriu al interfeței este transmisă controllerului de întreruperi.
- Controllerul gestionează cererile de întrerupere, verificând dacă cererea primită poate fi acceptată, funcție de biții de mascare IL1 şi IL0 din CCR. Întreruperea este acceptată dacă nivelul de prioritate al cererii este mai mic decât cel al mascării.
- Dacă întreruperea este acceptată PC şi PS sunt salvate în stivă, adresa de salt la întrerupere este luată din tabela de vectori. IL1 şi IL0 din CCR se modifică preluând valoarea biților de prioritate din întreruperea acceptată.
- Se execută rutina de întrerupere.
- La instrucțiunea RETI se restaurează PC şi PS (inclusiv IL1 şi IL0) şi se continuă programul principal.

Observații

1.Flagul de cerere de întrerupere din interfață nu se resetează după acceptarea întreruperii. Se impune resetarea lui prin program, în rutina de tratare a întreruperii.

2.O întrerupere scoate MC din starea de *standby*.

3.Întreruperile multiple pot surveni dacă CPU tratează o întrerupere şi apare o cerere de întrerupere cu prioritate acceptabilă (mai mică decât nivelul de mascare stabilit în CCR). Dacă se doreşte ca programul să accepte întreruperi multiple se setează temporar IL0 şi IL1 la un nivel mic (00B de exemplu) în rutina de servire a întreruperii. Restaurarea lui CCR readuce nivelul de acceptare a întreruperilor la valoarea inițială.

4.Timpul între apariția unei cereri de întrerupere şi servirea ei, format din timpul necesar finalizării instrucțiunii în curs şi gestionarea cererii de întrerupere este de maxim 26 ciclii maşină.

5.Acumulatorul A şi acumulatorul temporar T nu sunt automat salvați în stivă la acceptarea unei întreruperi, aşa încât se folosesc instrucțiunile de PUSHW şi POPW.

Întreruperi externe

Blocul pentru prelucrarea întreruperilor externe detectează frontul selectat al semnalelor aplicate la pinii externi şi transmite cererea către CPU. Cererile de întrerupere de la pinii externi pot fi folosite şi pentru revenirea din modurile de lucru cu economie de energie.

Există 8 canale de întrerupere 0-7, la fiecare canal îi corespund 2 pini externi de întrerupere INT00-INT15. Fiecare canal are un registru EIC.

EIC (Registrul pentru întreruperi externe, External Interrupt Control Register) comandă:

- validează cu 2 biți cererea de întrerupere pentru 2 linii de cerere de întrerupere
- selectează polaritatea cererii: front crescător, descrescător sau ambele fronturi pentr 2 linii
- validează transmiterea cererii de întrerupere către CPU pentru ambele linii
- se poziționează un flag care semnifică că a apărut o cerere de întrerupere

2.3.Porturi de I/O

Porturile de I/O controlează liniile de I/O. Porturile I/O disponibile, numărul de biți al fiecărui port, semnificațiile duale ale liniilor şi caracteristicile electrice ale liniilor depind de tipul de MC. În general porturile de I/O sunt 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G şi funcționarea lor este comandată de următorii regiştrii:

- PDR (Registrul de date, Port Data Register) în care se înscriu datele care trebuie să apară la pinii de ieşire sau se citesc datelele de la pinii de intrare.
- DDR (Registrul de sens, Data Direction Register) în care un 0 comandă linia corespunzătoare ca linie de intrare iar un 1 linie de ieşire.

- PUL (Registrul pentru comanda rezistențelor de *pull-up*, prin care se poate comanda conectarea la pinii de intrare a rezistențelor care mențin nivel *High*.
- ILSR (Registrul de selecție a nivelelor de intrare, Input Level Selection Register prin care se pot selecta pentru anumite porturi sau linii din port nivelele logice folosite: CMOS sau intrare cu histerezis.
- AIDRH, AIDRL (Regştrii de invalidare a intrărilor analogice, A/D Input Disable Register) care selectează pentru unul sau două porturi funcția de intrări analogice sau pini de I/O digitali.

Toate porturile I/O pot avea semnificații duale, în funcție de tipul MC.

Porturile 0-9 şi A-G au regiştrii de date PDR0-9, PDRA-G, de sens DDR0-9, DDRA-G. Porturile 0-8, E şi G au regiştrii de *pull up*. Există un singur registru ILSR în care câte un bit programează pentru câte o linie din porturi nivelul de intrare. Există 2 regiştrii (AIDRH şi AIDRL) care invalidează intrările analogice.

Schema bloc care arată funcționarea portului de I/O cu proprietatea Read Back (se poate citi ceea ce s-a înscris în port), valabilă atât pentru MC pe 8 biți cât şi pentru cele de 16 biți este dată în figura 2.2:

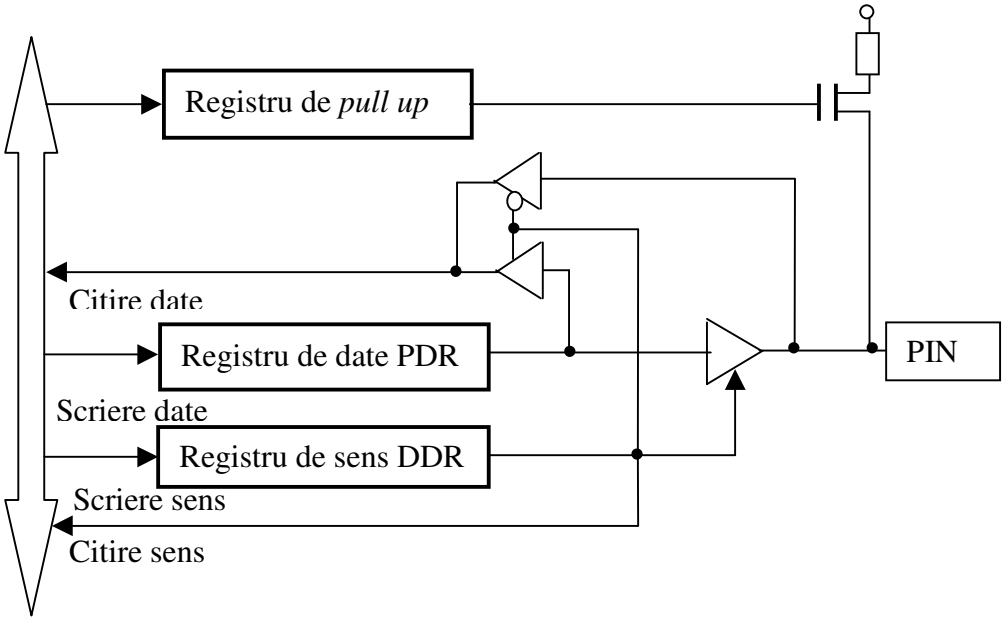


Figura 2.2: port I/O cu proprietatea Read Back

Operarea cu porturi de I/O

- În mod ieşire (bitul DDR corespunzător liniei este 1) datele trimise în PDR sunt scoase la ieşire şi păstrate în latch până la o nouă reînscriere. Citind PDR se obține valoarea scrisă în port.
- În mod intrare (bitul DDR corespunzător liniei este 0) datele sunt citite din PDR. Scrierea datelor în PDR este posibilă, dar datele nu apar pe pinii de intrare.

Observații:

1.Programarea regiştrilor DDR stabileşte semnificația pinilor ca I/O de uz general. Pentru folosirea ca intrări analogice de exemplu, se scrie în registrul AIDRL sau AIDRH un 1 pe bitul corespunzător liniei comandate. Dacă se doreşte atribuirea unei alte semnificații duale unui pin trebuie poziționat bitul de validare al dispozitivului care foloseşte linia.

2.La RESET regiştrii DDR se încarcă cu 0 şi liniile devin intrări (pentru un consum cât mai mic).

3.În modurile cu economie de energie (STOP sau WATCH) pinii intră în înaltă impedanță indiferent de DDR.

4.La fiecare model există linii de curent mare la care se pot lega direct LED-uri.

2.4.Generarea tactului, moduri de lucru cu economie de energie şi module de protecţie

Microcontrollerele Fujitsu sunt prevăzute cu 2 sisteme de tact independente pentru obţinerea unei game de frecvenţe de lucru mai largi şi un tact de rezervă pentru mărirea siguranţei în funcţionare. Variantele de MC cu S în denumire (Single Clock) au implementat doar tactul principal. Subtactul (Subclock) este folosit pentru obţinerea unei viteze mai mici de funcţionare şi implicit a unui consum mai redus. Subtactul este mult mai mic (4MHz tactul principal şi 32KHz subtactul). Tactul de rezervă este realizat cu componente RC interne şi înlocuieşte tactul principal în cazul unui defect, figura 2.3:.

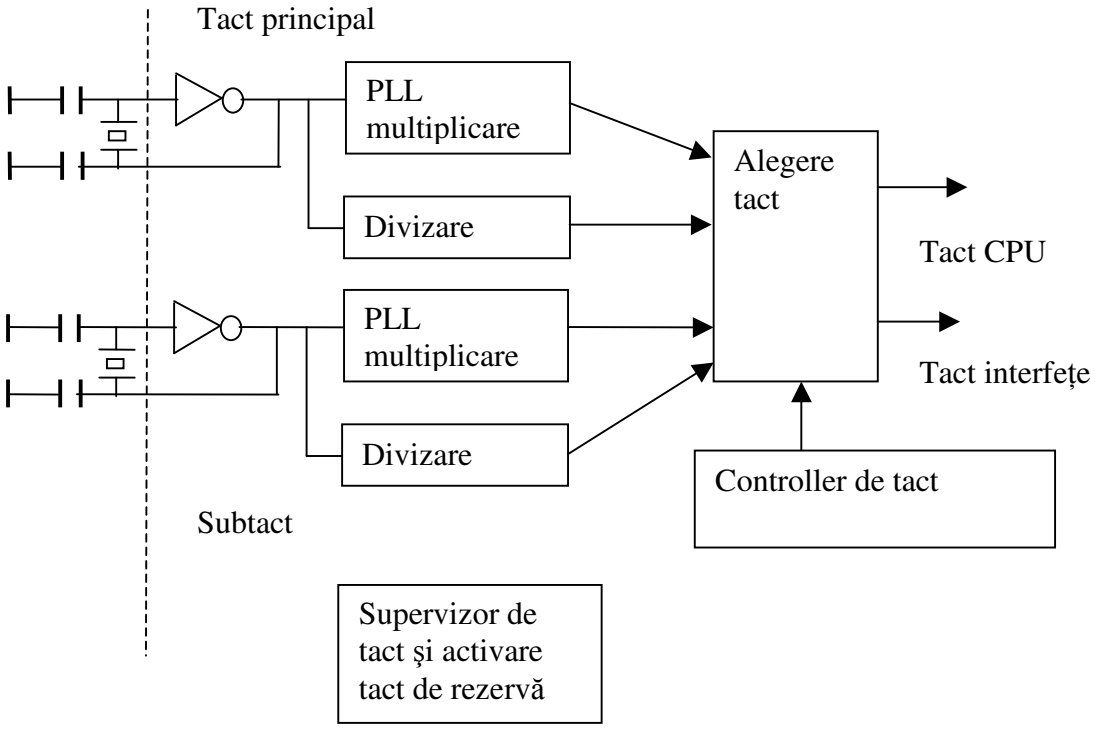


Figura 2.3: schema bloc a generării tactului

Controllerul de tact alege modul de tact şi modul de *standby*. Tactul pentru CPU şi pentru interfeţe poate fi diferit şi se obţine din tactul principal (sau din subtact) fiind posibile divizarea tactului sau multiplicarea cu buclă PLL.

Unele interfeţe din structura MC care sunt folosite la generarea unor secvenţe de timp nu sunt influenţate de divizarea sau multiplicarea tactului. Acestea sunt:

- Timer de bază (Timebase Timer) are tactul principal divizat cu 2.
- Timer (Ceas) de gardă (Watchdog Timer) are tactul principal sau subtact.
- Prescalare tact (Clock Prescaler) are subtactul divizat cu 2.
- Numărător de tact (Clock Counter) are subtact.

Modurile de lucru cu economie de energie

- Mod adormit (Sleep Mode), se opreşte tactul la CPU şi timerul de gardă. CPU se opreşte dar celelalte interfețe funcționează.
- Mod timer de bază, (Timebase Timer Mode) tactul este furnizat la timerul de bază, la prescalare tact şi numărătorul de tact. Rămân în funcțiune întreruperile externe şi blocul de RESET datorită tensiunii de alimentare prea mici. Este un mod de *standby* pentru tactul principal.
- Mod de observare (Watch Mode), tactul principal este oprit. Este un mod de *standby* pentru subtact.
- Modul STOP (Stop Mode) toate tactele sunt oprite. Funcționează doar întreruperile externe şi RESET datorită tensiunii mici.

Observație:

1.Câteva date orientative referitor la consumul de curent în diferite situații (consumul diferă în funcție de diferitele familii, tehnologie şi interfețele implementate:

- Operare normală cu tactul principal 50mA, operare normală cu subtact 300μA.
- Mod adormit cu tactul principal 25mA, mod adormit cu subtact 30μA.
- Mod STOP 7μA.

2.Timpul de stabilizare al oscilațiilor tactului este timpul între momentul pornirii unui tact şi momentul în care tactul este stabil. Controllerul de tact asigură acest timp prin numărarea oscilațiilor şi distribuie tactul stabil CPU şi interfețelor interne. Aşteptarea stabilizării tactului este realizată prin numărarea unor impulsuri de la Timerul de bază, iar pentru subtact se foloseşte prescalarea de tact. Stabilizarea după un RESET are o valoare inițială de 2^{14} /frecvența de tact. La fel se aşteaptă şi stabilizarea oscilațiilor buclei PLL.

Regiştrii de comandă pentru tact

SYCC (System Clock Control Register) comandă factorul de divizare între 1 şi 16, validarea tactului principal, validarea subtactului, modul de tact (tact principal, tact principal prin PLL, subtact, subtact prin PLL etc.).

PLCC (PLL Control Register) comandă pentru tactul principal şi pentru subtact factorul de multiplicare x2, x3 sau x4, validare /invalidare PLL şi indică starea tactelor: oprite (sau în curs de stabilizare) sau operaționale.

WATR (Asignarea timpului de stabilizare, Oscillation Stabiliyation Standby Time Assignment Register) stabileşte timpul de aşteptare a stabilizării între 4,10ms şi 0,5μs (pentru tactul principal) şi între 1s şi 61μs (pentru subtact).

STBC (Standby Control Register) controlează tranziția între modurile de operare cu economie de energie.

RSRR (Reset Source Register) arată cauzele unui RESET: software, hardware, la punerea sub tensiune, generat de Watchdog, extern, generat de supervizorul de tact.

- Un RESET extern înseamnă un nivel de 0 logic la intrarea /RST cu o anumită lungime minimă.
- RESET software se obține prin scrierea unui bit în registrul STBC (Standby Control Register).
- RESET generat de Watchdog se obține dacă Watchdog-ul nu este reinițializat în perioada de timp programată.
- RESET datorită subtensiunii de alimentare (opțiune posibilă doar la unele dintre modelele de MC alimentate la 5V) este echivalent cu RESET-ul extern. Se monitorizează tensiunea de alimentare şi se generează RESET dacă tensiunea scade sub 2,6V. Circuitul de detecție este realizat cu un comparator şi o referință. Dacă tensiunea revine peste 2,7V RESET-ul nu mai este menținut. Detecția este operațională în orice mod de lucru.

- Supervizorul de tact generează un RESET dacă oscilațiile se opresc în mod anormal şi nu datorită unor tranziții între modurile de lucru. După RESET funcționarea MC se realizează cu un tact intern RC (opțiune valabilă doar la unele din modelele alimentate la 5V). Supervizorul comută tactul intern dacă nu este detectat un front al tactului principal timp de 4 ciclii RC sau nu este detectat un front al subtactului timp de 32 de ciclii RC. Supervizorul este comandat de registrul CSVCR (Clock Supervisor Control Register) care validează detectarea pentru tactul principal sau subtact.

2.5.Programarea memoriei FLASH

Sunt posibile 3 metode de programare şi ştergere a memoriei FLASH:

- programare paralelă
- programare serială
- programare printr-un program intern

Programarea serială din exterior este comandată de pinii externi MD şi pinii canalului serial 0.

Dimensiunea memoriei este de 480Kbit şi este situată între adresele 1000H şi FFFFH, organizată în 2 bankuri. Nu se poate scrie şi citi un bank în acelaşi timp dar se poate scrie într-un bank şi citi din celălalt (operare dual㠖). Caracteristicile principale sunt:

- organizarea memoriei de 480Kbit (60K octeți) este în sectoare (şapte sectoare de 4K şi două de 16K), sectoarele fiind grupate în 2 bank-uri.
- algoritmul de scriere/ citire este integrat (Embedded Algoritm ™)
- cel puțin 10 mii de programări / ştergeri posibile
- timpul de citire este de un ciclu maşină
- terminarea scrierii/ ştergerii se poate realiza prin întreruperi.

Bank-urile sunt împărțite în sectoare astfel:
SA1 (4K octeți) 1000H-1FFFH (bank inferior)
SA2 (4K octeți) 2000H-2FFFH (bank inferior)
SA3 (4K octeți) 3000H-3FFFH (bank inferior)
SA4 (16K octeți) 4000H-7FFFH (bank superior)
SA5 (16K octeți) 8000H-BFFFH (bank superior)
SA6 (4K octeți) C000H-CFFFH (bank superior)
SA7 (4K octeți) D000H-DFFFH (bank superior)
SA8 (4K octeți) E000H-EFFFH (bank superior)
SA9 (4K octeți) F000H-FFFFH (bank superior)

Regiştrii de comandă pentru memoria FLASH

FSR (Starea memoriei FLASH, FLASH Memory Status Register) validează programarea /ştergerea, validează întreruperile la terminarea programării / ştergerii, un bit arată starea (în curs de programare sau programare terminată), un bit poate schimba alocarea adreselor sectoarelor SA3 cu SA9 (pentru operarea în mod dual).

SWRE 0, SWRE 1 (Controlul scrierii, FLASH Memory Sector Write Control Register) sunt 2 regiştrii care validează scrierea sectoarelor în mod individual.

Funcționarea

Sunt 4 tipuri de comenzi care folosesc algoritmul integrat de scriere / citire: citire/RESET, scriere, ştergere memorie şi ştergere sector. La scriere/ştergere se poate verifica starea comenzii după fiecare octet scris citind cuvântul de la adresa unde s-a scris. Se obține un octet care conține 5 flag-uri:

- DQ7, DQ6 arată dacă operația este în curs sau a fost executată (similar cu un bit din FSR) şi dacă operația se referă la întreaga memorie sau la un sector.

- DQ5 arată depăşirea timpului normal al operaţiei
- DQ3 arată depăşirea timpului normal al operaţiei de ştergere sector
- DQ2 arată că operaţia de ştergere a fost suspendată.

Din cele 4 comenzi se exemplifică pe scurt două:

Citirea din FLASH se poate face în 2 moduri:

1.normal, citind conţinutul unei adrese care se află în zona FLASH.

2.prin algoritmul integrat astfel (secvenţele de comandă se află în datele de catalog ale circuitelor):

- se programează sectorul din care dorim citirea în mod citire cu regiştrii de comandă (mod implicit după RESET)
- la adresa de unde dorim să citim se scrie F0H în mod continuu şi se citeşte DQ7 pentru a se vedea când se pot citi datele.

Scrierea în FLASH

- se programează sectorul în care vrem să scriem în mod scriere
- se scrie o secvenţă de program în mod continuu şi se citeşte DQ7 pentru a vedea când se pot scrie datele. Secvenţa este: AAH se trimite la adresa UAAAH, 55H se trimite la adresa U554H, A0H se trimite la adresa UAAAH, apoi datele se trimit la adresa dorită pentru înscriere. U (primii 4 biţi de adresă) trebuie să fie aceeaşi cu cei din adresa de scriere dorită).

Observaţie: un bit de 0 nu poate fi modificat în 1 prin scriere deoarece flag-urile vor indica eronat (trecerea din 0 în 1 a lui DQ5 indică o depăşire a timpului operaţiei). În acest caz este necesară ştergerea sectorului.

Securitatea memoriei FLASH

Dacă se scrie 01H la adresa 4000H se invalidează accesul din exterior. Singura operaţie admisă după protejarea memoriei este ştergerea sectorului.

Operarea duală

Memoria FLASH poate opera cu cele 2 bank-uri în paralel, unul poate fi scris / şters iar celălalt citit.

Dificultatea operării duale constă în faptul că vectorii de întrerupere sunt situaţi în SA). De aceea este necesară schimbarea sectoarelor SA9 cu SA3 printr-un bit din FSR.

O diagramă de timp arată modul de operare normal, figura 2.4:

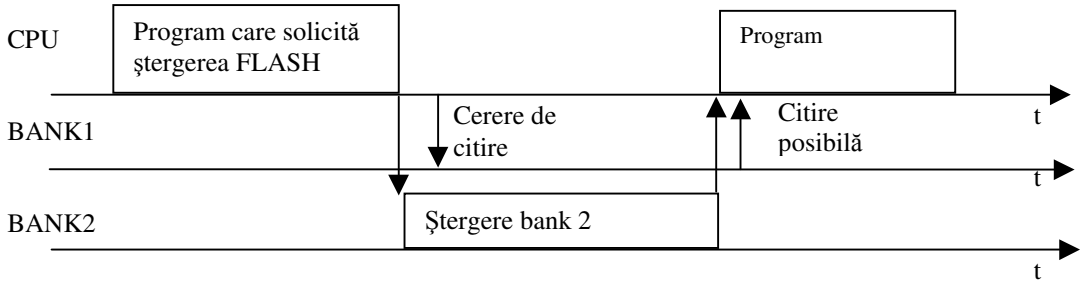


Figura 2.4: operarea normală

În operare normală în timpul ştergerii unui bank nu se poate citi din celălalt bank. O cerere de citire trebuie să aştepte până la terminarea ştergerii. Avantajele operării duale sunt demonstrate în figura 2.5:

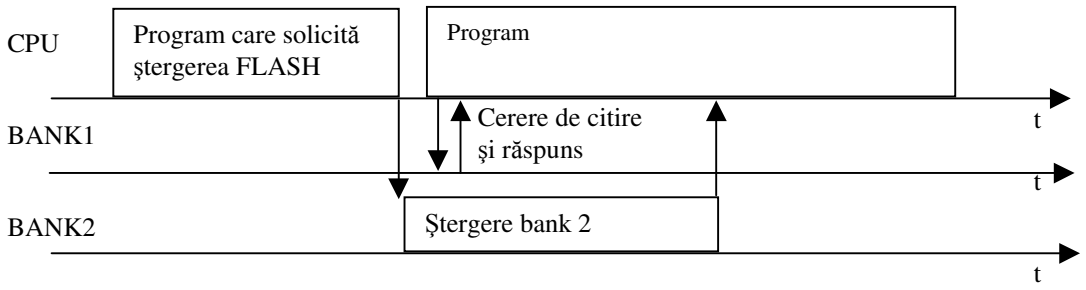


*Figura 2.5: operarea dual*ă

În operare dual*ă* se face în acelaşi timp citirea dintr-un bank şi ştergerea celuilalt.

Programarea extern*ă* serial*ă*

Se realizează prin canalul serial asincron USART. Pinii canalului USART0 sunt folosiţi pentru a stabili modul de programare serial iar transferul de date se ralizează prin USART1, figura 2.6:

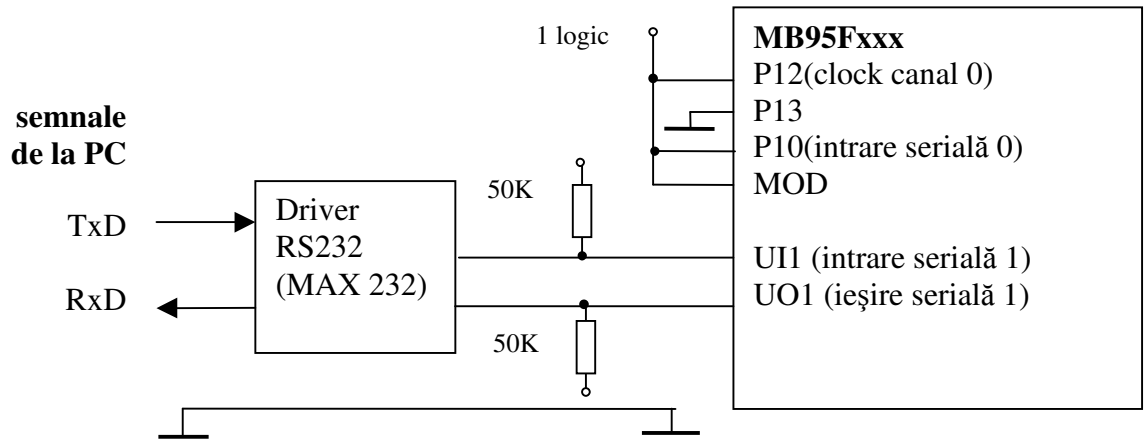


*Figura 2.6: conexiunile pentru programarea serial*ă *extern*ă

Aplicaţia care realizează programarea se numeşte FLASH.EXE . Ecranul iniţial al aplicaţiei este prezentat în figura 2.7. Se alege tipul de MC şi frecvenţa externă (a cristalului). Cu *Set environment* se poate alege portul serial din PC. Se alege fişierul HEX (în format Motorola HEX) în care este stocat programul şi se trimite spre MC. Microcontrollerul trebuie RESET-at. Programarea se face în mai mulţi paşi: *Download, Erase, Blank Check, Program and Verify* sau cu opţiunea rapidă *Full Operation.*

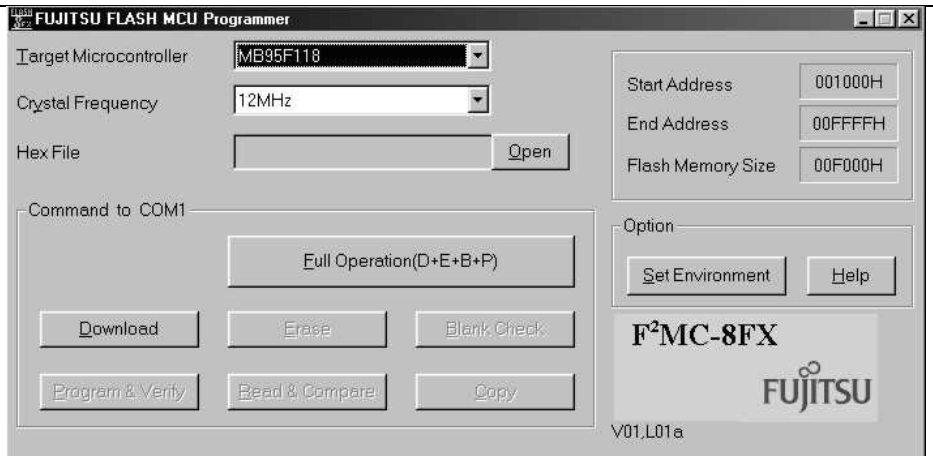


Figura 2.7: ecranul aplicației FLASH.EXE pentru programarea serială externă

2.6.Regiştrii WILD

Aceşti regiştrii se folosesc pentru a modifica programe care au defecte. Regiştrii WILD constau în:

- 3 regiştrii de date de 8 biți
- 3 regiştrii de adresă de 16 biți pentru adresa inferioară şi adresa superioară
- un registru validare a comparării adresei
- un registru de test.

Se pot înlocui 3 locații din ROM, oriunde în spațiul ROM prin datele conținute în regiştrii WILD. Opțiunea aceasta este folositoare după ce s-a programat (prin ardere ROM, *mask ROM*) memoria ROM pe chip.

Regiştrii de comandă WILD

WRD0, WRD1 şi WRD2 sunt regiştrii de date al căror conținut se trimite pe magistrala de date.

WRAR0, WRAR1 şi WRAR2 (16 biți) specifică adresa pentru care, atunci când se citeşte din ROM se substituie datele.

WREN, registru de validare care validează substituirea datelor pentru fiecare registru de date în parte

WROR registru de test, validează citirea pentru fiecare registru de date în parte.

Adresele puse de CPU sunt comparate continuu cu adresele WILD (cu cele validate în WREN) şi dacă apare o coincidență datele nu sunt citite din ROM ci din registrele WILD (cu cele validate în WROR pentru citire).

3.MICROCONTROLLERE PE 16 BIŢI (F^2MC-16LX)

3.1.Regiştrii, maparea memoriei, adresarea, instrucţiuni

Aspecte specifice la tipul datelor şi adresare

Datele vehiculate în acest MC pot fi pe 8 biţi (Byte), pe 16 biţi (Word) şi pe 32 de biţi (Long).

Memoria adresabilă este de 16M, magistrala de adrese fiind de 24 de biţi. Memoria se poate adresa în 2 moduri:

- Adresare liniară, se specifică în instrucţiune o adresă de 24 de biţi.
- Adresare cu regiştrii ajutători în care se stochează cei mai semnificativi 8 biţi de adresă. În instrucţiune se specifică ceilalţi 16 biţi de adresă.

1.Adresarea liniară

- adresarea directă: JMPP 123456H are ca efect salt la adresa 123456H, adică înlocuirea PC (Program Counter) cu valoarea 123456H
- adresarea indirectă: MOV A, @RL1+1 RL1 este un registru de 32 de biţi care conţine adresa cu al cărei conţinut încărcăm acumulatorul. Cei mai semnificativi 8 biţi din RL se ignoră. Dacă RL=12345678H, se ignoră 12H, RL+1=345678H+1=345679H. Conţinutul adresei 345679H se încarcă în acumulator.

2.Adresarea cu regiştrii PCB(Program Bank Register), DTB (Data Bank Register), USB (User Stack Bank Register), SSB System Stack Bank Register), ADB (Additional Bank Register), DPR (Direct Page Register) (regiştrii de 8 biţi).

De exemplu dacă PCB=FFH spaţiul program poate fi între FF0000H şi FFFFFFH. Dacă DTB=50H spaţiul de date poate fi între 500000H şi 50FFFFH.

Exemplu: dacă DTB este 00H

Prin adresare directă MOV A, 1234H, se încarcă în acumulator valoarea stocată la adresa 001234H.

Sau prin adresare indirectă MOV A, @RW1 se încarcă în acumulator valoarea stocată la adresa formată din 00H (DTB) şi conţinutul lui RW1.

Regiştrii speciali:

Acumulatorul A constă din 2 regiştrii de 16 biţi (AH şi AL) şi este folosit la stocarea temporară a rezultatelor.

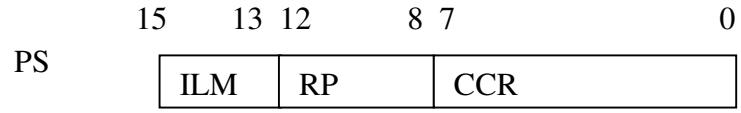
Dacă RW1 este 0100H şi DTB este 00H

MOVL A,@RW1+6 adresa este formată din DTB=00H şi RW1+6=0106H

Adresa	Conţinut	
000109H	BBH	→ AH devine BB22H
000108H	22H	
000107H	AAH	→ AL devine AA33H
000106H	33H	

Indicatoarele de stivă USP (Stiva utilizator, User Stack Pointer) şi SSP (Stiva sistem, System Stack Pointer) sunt registre de 16 biţi care indică adresa de memorie pentru a salva/ restaura date sau pentru intrarea/ ieşirea din subrutine. USP este validată dacă bitul S din registrul de stare a procesorului este 0 iar SSP când bitul este 1. SSP este folosit în cazul întreruperilor iar USP pentru salvări/ restaurări de date. Se poate folosi acelaşi spaţiu folosind doar SSP.

Registrul de stare a procesorului PS

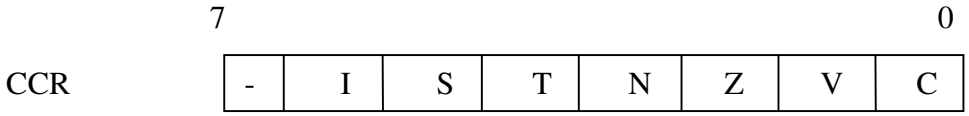


ILM (Nivelul de mascare al întreruperilor, Interrupt Level Mask Register) (de 3 biţi) arată nivelul curent al mascării întreruperii. O cerere de întrerupere este acceptată doar dacă nivelul ei este mai mare decât cel indicat de ILM. Nivelul 0 este cel mai prioritar. Dacă ILM=000B atunci întreruperile sunt invalidate.

RP (Register Bank Pointer) (de 5 biţi) indică relaţia între registrele de uz general şi adresele memoriei RAM. RP indică adresa de memorie a bankului de registre folosit şi poate lua valori între 00H şi 1FH, iar bankurile de registre pot fi situate între 000180H şi 00037FH în memorie. Relaţia de alocare este:

Adresa bank=000180H * RPx10H

CCR (Registrul de condiţii al programului, Condition Code Register)



I=1 întreruperile sunt validate (Interrupt)

S=1 se foloseşte SSP, dacă S=0 USP (Stack)

T=1 dacă în urma unei deplasări prin Carry există un bit de 1 în datele deplasate care trece de Carry (Sticky)

N=1 dacă rezultatul unei operaţii are un bit de 1 pe poziţia celui mai semnificativ bit (număr negativ în complement faţă de 2) (Negativ)

Z=1 dacă rezultatul unei operaţii este 0 (Zero)

V=1 dacă există o depăşire (Overflow)

C=1 dacă apare transport (Carry)

Registrul PC (Indicator de program, Program Counter) este un registru pe 16 biţi care arată cel mai puţin semnificativi 16 biţi ai adresei de program care se execută de CPU. Cei mai semnificativi 8 biţi sunt stocaţi în PCB.

Regiştrii pentru adresare sunt regiştrii de 8 biţi

PCB(Registru de spaţiu program, Program Bank Register), împreună cu PC formează o adresă de 24 de biţi. PCB defineşte spaţiul program.

DTB (Registru de spaţiu date, Data Bank Register). Acest spaţiu este folosit implicit de adresările @RW0,1,4,5, @A, addr 16, dir

USB (Registru de spaţiu stiva utilizator, User Stack Bank Register), spaţiu folosit implicit de PUSHW, POPW, @RW3,7

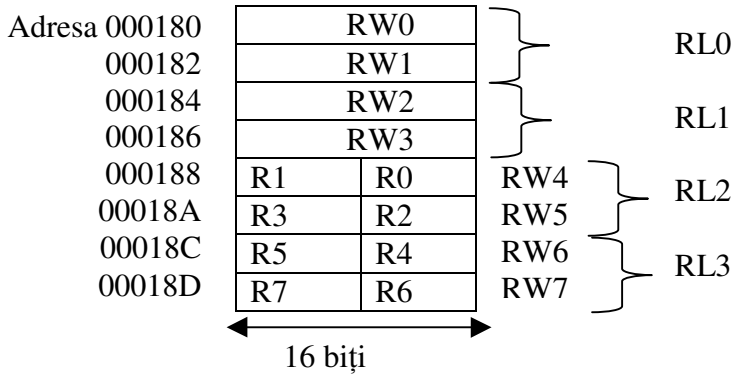
SSB (Registru de spaţiu stiva sistem, System Stack Bank Register), spaţiu folosit implicit de PUSHW, POPW, @RW3,7

ADB (Registru de spaţiu adiţional, Additional Bank Register), spaţiu folosit de @RW2,6

DPR (Registru de pagină, Direct Page Register) specifică 8 biți de adresă , aşa încât dacă DTB specifică cei mai semnificativi 8 biți de adresă, adresarea în interiorul unei pagini să se poată face cu instrucțiuni care specifică 8 biți de adresă (cel mai puțin semnificativi):

Regiştrii de uz general:

Sunt localizați între adresele 000180H şi 00037FH. R0-R7 sunt regiştrii pe 8 biți, RW0-RW7 de 16 biți iar RL0-RL3 de 32 de biți. În RAM se pot defini maximum 32 de bankuri de astfel de regiştrii.



Observații

1.Referitor la registrul Acumulator, datele pe 16 biți (Word) se încarcă în AL. Când se primesc date noi în AL (şi dacă datele noi sunt de tip Byte sau Word), datele vechi se încarcă în AH (funcție de păstrare a datelor, Date preservation function). Se pot realiza operații aritmetice între AH şi AL şi astfel procesarea devine mai eficientă. Această funcție a AH este asemănătoare cu funcția acumulatorului temporar de la MC pe 8 biți.

2.Date multibyte sunt date formate din mai mulți octeți. Acestea se scriu/ citesc din memorie începând de la adresa inferioară.

De exemplu MOV A,808000H Conținutul adresei 807FFFH intră în AL partea inferioară (8 biți) iar conținutul adresei 808000H intră în AL partea superioară. Transferurile de acest tip consideră acelaşi spațiu în timpul execuției instrucțiunii, adică adresa inferioară lui 800000H este 80FFFFH.

Coduri prefix

Funcția unei instrucțiuni se poate modifica adăugând în fața ei un cod prefix. Există următoarele tipuri de coduri prefix:

- Prefix pentru alegerea spațiului (bankului) de memorie, (indiferent de spațiul folosit implicit de acea instrucțiune). Prefixe posibile sunt PCB, DTB, ADB, SPB (System sau User Stack Space, funcție de bitul S).
- Pentru a uşura schimbul de date între task-uri se defineşte un bank de regiştrii de uz general comun, indiferent de valoarea lui RP (Register Bank Pointer). Acest bank este alocat între

000180H şi 00018H. El este selectat când RP este 0 sau când în faţa instrucţiunii există prefixul CMR (Common Register Bank Prefix) indiferent de valoarea lui RP.

- NCC (Inhibarea schimbării flagurilor, Flag Change Inhibit Prefix Code) este folosit pentru a elimina schimbarea flagurilor de către instrucţiunea cu acest prefix.

Observaţie: Unele instrucţiuni nu ţin cont de prefixe, uneori sunt afectate şi instrucţiunile următoare.

Moduri de adresare

Sunt implementate 23 de moduri de adresare grupate în 2 tipuri, adresare directă şi indirectă.

Adresare directă

Imediată (#imm), valoarea imm poate fi pe 4,8,16 sau 32 de biţi.

MOVW A,#1212H încarcă imediat 16 biţi în acumulator (AL). Valoarea care era în AL este mutată în AH (funcţia de acumulator temporar).

Adresare directă cu regiştrii, se pot folosi regiştrii de uz general sau regiştrii speciali.

MOV R0,A mută conţinutul lui A (8 biţi mai puţin semnificativi din AL) în R0

Salturi directe (addr16). Se specifică o adresă pe 16 biţi care împreună cu PCB formează o adresă de 24 de biţi pentru salt.

JMP 3820H dacă PCB=4FH se face salt la 4F3820H.

Salturi la adrese fizice (addr24). Se specifică o adresă pe 24 de biţi.

Adresare directă I/O. Se specifică o adresă de 8 biţi în instrucţiune. Restul adresei este completat automat pentru spaţiul I/O, indiferent de DTB sau DPR. Un cod prefix de specificare a unui spaţiu este invalid.

MOVW A,I:C0H citeşte date în acumulator de la portul C0H şi C1H (16 biţi) care intră în AL (C0 în AL partea inferioară, C1 în AL partea superioară).

Adresare directă prescurtată (dir) specifică 8 biţi de adresă (0-7). Adresele 8-15 sunt specificate de DPR iar 16-23 de DTB.

MOV S:20H,A dacă DTB=77, DPR=66H, conţinutul lui AL intră la adresa 776620H şi 776621H

Adresare directă (addr16) specifică 16 biţi de adresă (0-15). Adresele 16-23 sunt în DTB.

Adresare directă pe bit în spaţiul I/O

SETB I:C1H:0 se setează bitul 0 al octetului de la adresa 0000C1H (zona I/O)

Adresare prescurtată pe bit, ca şi adresarea prescurtată

SETB S:10H:0 dacă DTB=77H, DPR=66H, se setează bitul 0 al octetului de la adresa 776610H

Adresare directă pe bit (addr16:bp), 8 biţi de adresă sunt specificaţi în DTB

SETB 2222H:0 dacă DTB=77H, se setează bitul 0 de la adresa 772222H

Adresare cu vector (#vct) specifică apelului unei subrutine sau întreruperi soft.

CALLV #15 apelează subrutina cu adresa stocată în tabela de vectori.

CALLV #_	**Adresa de salt**	
	Octet inferior	**Octet superior**
CALL #0	FFFEH	FFFFH
CALL #1	FFFCH	FFFDH
----------	----------	----------
CALL #14	FFE2H	FFE3H
CALL #15	FFE0H	FFE1H

Din această corespondenţă, pentru fiecare apel corespund 16 biţi de adresă. Cei 8 biţi mai semnificativi se află în PCB.

Adresare indirectă

Adresare indirectă cu registre (@RWj, j=0,1,2,3)

Adresa operandului este stocată în registrul RWj (16 biți) iar 8 biți în DTB la utilizarea RW0 sau RW1, în SSB (USB) la utilizarea RW3 sau ADB la utilizarea RW2 (utilizarea implicită a registrelor de uz general a fost menționată la descrierea regiştrilor pentru adresare).

MOVW A, @RW1 dacă DTB=78H, RW1=D30FH, conținutul adreselor 78D30FH şi 78D310H se încarcă în AL.

Adresare indirectă cu regiştrii cu post increment (@RWj+)

După execuția instrucțiunii, RWj este incrementat cu 1 pentru Byte, 2 pentru Word şi 4 pentru Long.

Adresare indirectă cu regiştrii şi offset (@RWi+disp8, i=0-7, @RWj+disp16,j=0-3).

Regiştrii speciali de adresare se folosesc conform destinației implicite.

Adresare indirectă cu regiştrii de 32 de biți şi offset (@RLi+disp8, i=0-3)

MOVW A, @RL2 + 25H RL2=F3824B02H, F3H nu contează, adresa este 824B02H+25H= 824B27H şi conținutul acestei adrese intră în AL (partea mai puțin semnificativă) iar conținutul adresei 824B28H intră în AL (partea mai semnificativă).

Adresare indirectă cu PC şi offset (@PC+disp16). Adresa este formată din PC şi PCBla care se adaugă offsetul.

Adresare indirectă cu registru index (@RW0+RW7, @RW1+RW7). Adresa este formată din DTB şi suma registrelor RW0sau RW1 şi RW7.

Adresare relativă la PC

BRA 10H salt necondiționat relativ. Dacă PCB=4FH, PC=3C32H saltul se face la 4F3C32H+10H= 4f3C42H.

Salvare/ restaurare regiştrii din listă (rlst) specifică un registru care se introduce sau se scoate din stivă.

Adresare indirectă cu acumulatorul (@A). Adresa este indicată de AL iar adresele 16-23 sunt date în DTB.

MOVW A, @A AL=2534H, DTB=10H, datele de la adresa 102534H intră în AL (partea mai puțin semnificativă) iar datele de la adresa 102535H intră în AL (partea mai semnificativă).

Salt indirect adresat cu acumulatorul (@A)

JMP @A PCB=4FH, AL=3B20H, se face salt la 4F3B20H

Salt indirect la o adresă specificată (@ear, @eam)

JMP @@RW0 DTB=21H, RW0=7F48H la adresa 217F48H se află 20H, iar la 217F49H se află 3BH, PCB=4FH, saltul se face la 4F3B20H

JMP @RW0 DTB=21H, RW0=7F48H, PCB=4F, saltul se face la 4F7F48H

Moduri de lucru şi comportarea la RESET

Modurile de lucru sunt stabilite de 3 pini externi astfel:

MD2	MD1	MD0	Mod de lucru
0	0	0	Bus extern de 8 biți şi tabelă de vectori externă
0	0	1	Bus extern de 16 biți şi tabelă de vectori externă
0	1	1	Secvența de Reset şi spații controlate de cuvântul de mod (Mode Data). Tabela de vectori este situată în ROM
1	1	0	Programare serială a FLASH-ului
1	1	1	Programare paralelă a FLASH-ului

Dacă este selectat modul de lucru controlat de cuvântul de mod se citeşte cuvântul de mod care este stocat la FFFFDFH şi este folosit pentru a controla CPU. Acest cuvânt poate fi schimbat doar prin RESET. Biții M1 (bit 7) şi M0 (bit 6) specifică modul de operare după RESET.

M1	M0	Mod de lucru comandat de cuvântul de mod
0	0	mod *single chip*
0	1	ROM intern, bus extern
1	0	ROM extern, bus extern

Bitul S0 (bit 3) specifică lărgimea busului extern de 8 biţi (S0=0) sau 16 biţi (S0=1).
Observaţie: alocarea memoriei şi semnificaţia unor pini diferă funcţie de modul de lucru.

Referitor la comportarea MC la RESET, există 4 cauze posibile pentru RESET:

- La punerea sub tensiune
- Depăşire WATCHDOG
- RESET extern prin punerea la 0 a pinului de RESET
- Cerere de RESET software

Diagrama de funcţionare la execuţia unui Reset este dată în figura 3.1:

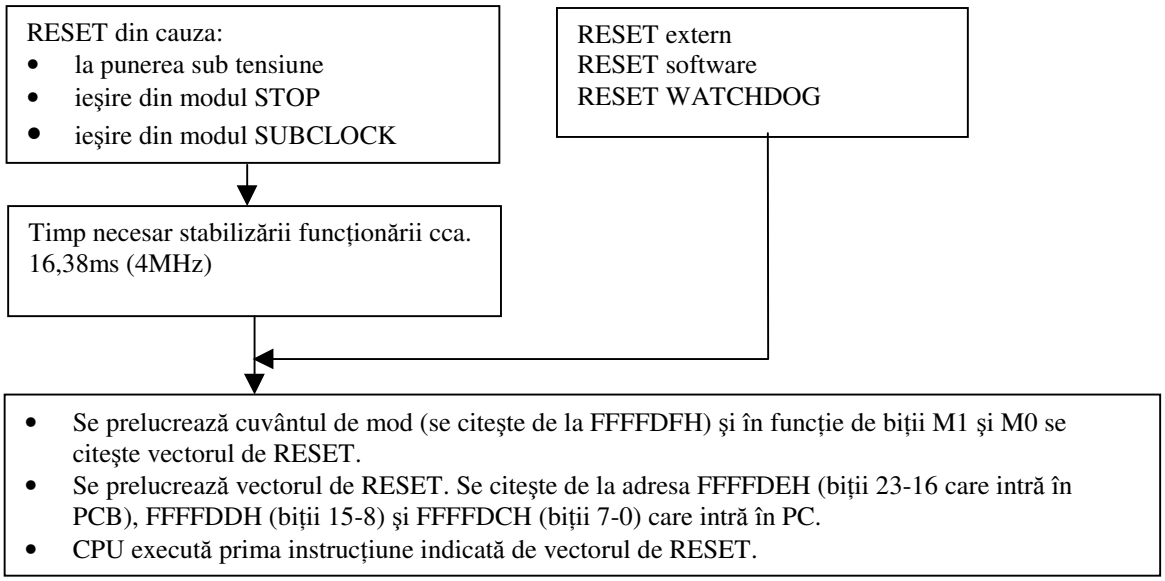


Figura 3.1: diagrama de funcţionare la execuţia unui Reset

Observaţie: fiecare eveniment care poate produce un RESET poate fi identificat prin alocarea unui flag blocului corespunzător.

Maparea memoriei (figura 3.2.)

Observaţii

1.De cele mai multe ori cuvântul adresă se referă la adresa unui operand. Pentru prescurtare s-a omis în text această precizare care reiese din context.

2.Prin convenţie cifrele hexa peste 9 sunt precedate în notare de un 0. Pentru a fi clar în text câţi octeţi are o adresă acest 0 s-a omis.

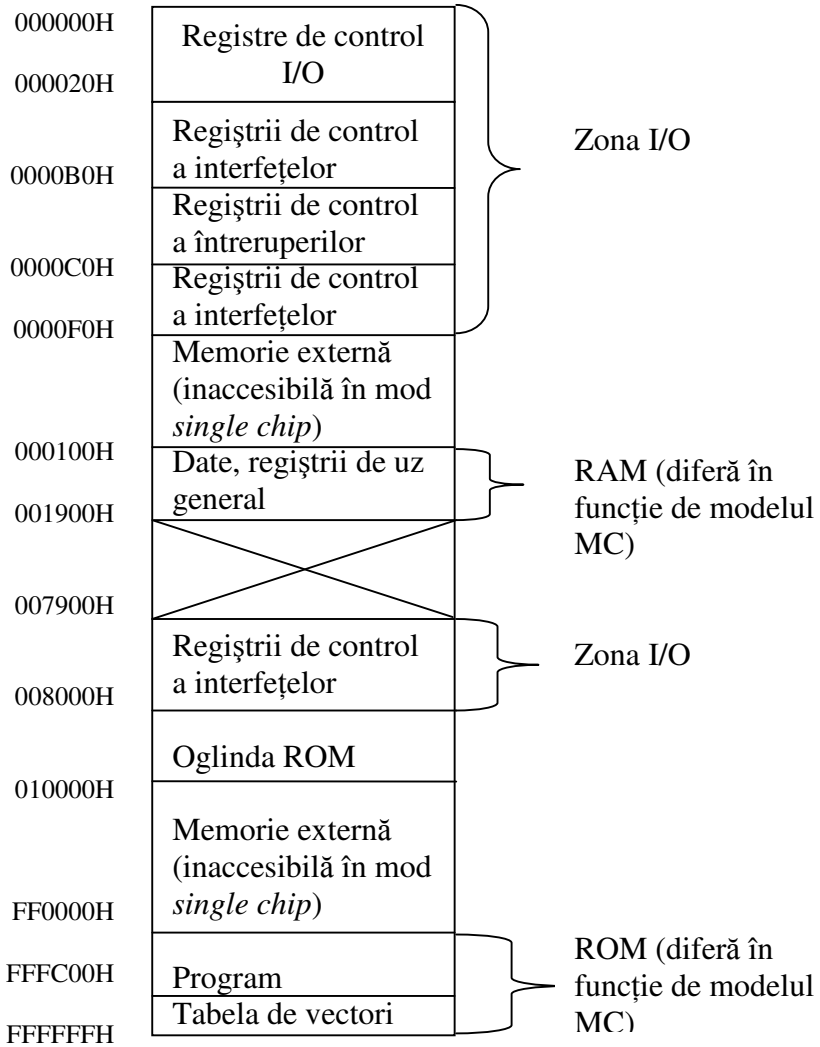


Figura 3.2: maparea memoriei

Oglinda ROM

Zona de memorie din bankul FFH (FLASH sau ROM) se încarcă în zona de RAM în bankul 00H. Principiul este asemănător celui de la calculatoarele PC la care memoria ROM video se încarcă în RAM (video shadow). Această operaţie este necesară pentru că anumite date constante sunt stocate în FLASH de către link editor, dar fiind date este bine şi mai uşor (nu se schimbă spaţiul) să fie accesibile din RAM.

3.2.Funcţionarea în întreruperi şi DMA

Există 5 tipuri de întreruperi:

- Hardware, datorită cererii unui dispozitiv intern sau activarea unei linii de cerere de întrerupere externă
- Software, datorită unui eveniment software (instrucţiune)
- Întreruperi de la serviciul EI^{2}OS (Extended Intelligent I/O Service)
- µDMAC datorită unui eveniment legat de transferul DMA
- excepţii

Întreruperi hardware

Se solicită o întrerupere hardware dacă dispozitivul intern are validat regimul de întreruperi cu un flag specific şi dacă flagul de cerere de întrerupere este activ. Fiecare dispozitiv intern are asociat un registru ICR (Interrupt Control Register) în care se poate programa nivelul de prioritate al cererii, dacă se lucrează în întreruperi obişnuite sau cu EI^{2}OS şi selecţia canalului EI^{2}OS.

ICR	ICS3	ICS2	ICS1	ICS0	ISE	IL2	IL1	IL0

IL0, IL1, IL2 programează nivelul de prioritate al întreruperii de la 000B (cel mai prioritar) la 110B (cel mai puţin prioritar) şi 111B întreruperi nvalidate. Bitul ISE=1 activează serviciul EI^{2}OS. Când EI^{2}OS este activat, biţii ICS3, 2, 1, 0 determină canalul de lucru (de la 0 la 15D) şi descriptorii EI^{2}OS (situaţi între 000100H şi 000178H).

Nivelul de mascare al întreruperii poate fi programat în registrul PS al CPU, biţii ILM (IL0, IL1, IL2). Întreruperea este acceptată dacă nivelul ei de prioritate este mai mare decât nivelul de mascare (IL0 IL1 IL2 din ICR < IL0 IL1 IL2 din ILM). Dacă întreruperea este acceptată se salvează în stivă 12 octeţi: PS, PC, PCB, DTB, ADB, DPR, A în locaţiile de memorie indicate de SSB şi SSP.

Întreruperi software

Se solicită o întrerupere la o instrucţiune INT. Aceste întreruperi nu au nivele de prioritate sau flaguri de validare. Servirea lor se face la fel cu servirea întreruperilor hardware.

Întreruperi EI^{2}OS

Acest serviciu transferă date automat între o resursă internă şi memorie, asemănător transferului DMA deoarece datele nu mai trec prin acumulator. Sursa şi destinaţia transferului sunt determinate prin descriptori. Descriptorii sunt stocaţi în RAM la adrese indicate de biţii ICS3, 2, 1, 0 din ICR. Descriptorii (ISD, EI^{2}OS Service Descriptor) au următoarea configuraţie, figura 3.3:

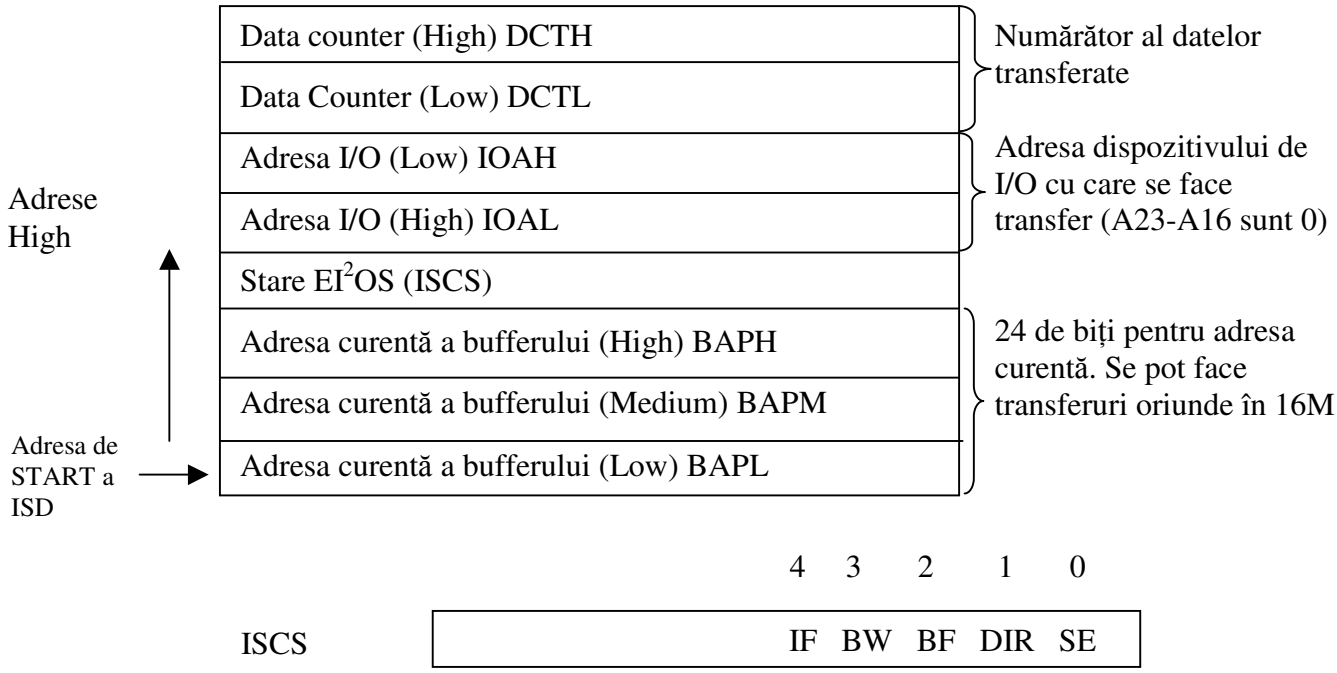


Figura 3.3: descriptori EI^{2}OS

IF specifică dacă adresa I/O este fixă sau poate fi modificată după transfer
BW specifică dacă transferul se face pe Byte sau Word
BF specifică dacă adresa bufferului este fixă sau poate fi modificată după transfer
DIR sensul transferului
SE=1, EI^{2}OS s-a terminat la o cerere a dispozitivului înainte de transferarea tuturor cuvintelor.

Excepții

Întreruperea datorită unei excepții se execută ca orice întrerupere. De exemplu la întâlnirea unei instrucțiuni cu un cod nedefinit (cod inexistent, care poate apare doar în cazul unei erori) se generează o astfel de întrerupere (INT 10). Se recomabdă ca în tratarea acestei întreruperi să se activeze proceduri de recuperare a funcționării, de exemplu RESET.

Vectori de întrerupere

Sunt utilizați atât pentru întreruperile soft cât şi hard. O parte din vectori sunt alocați dispozitivelor interne. De exemplu nivelul de întrerupere care corespunde lui INT 29 este alocat convertorului A/D. Sunt posibile 256 de nivele, de la INT 0 la INT 255, dintre care INT 0-INT 7 sunt comune şi pentru adresarea vectorizată (CALLV #nr), INT 8 este alocat acțiunii RESET, iar nivelele INT 9- INT 42 au semnificații speciale, fiind alocate dispozitivelor interne.

În tabelul cu corespondența nivel de întrerupere -vector este specificată şi adresa registrului ICR pentru fiecare nivel. De exemplu pentru convertorul A/D ICR este la adresa 0000B)H iar rutina de servire are adresa formată astfel:

- La FFFF88H se află 8 biți mai puțin semnificativi
- La FFFF89H se află 8 biți mai semnificativi
- La FFFF8AH se află 8 biți pentru adresa bankului (spațiu program)

Întreruperi multiple

Dacă o întrerupere cu nivel de prioritate mai mare apare când se execută o rutină de întrerupere, după execuția instrucțiunii în curc se acceptă întreruperea cu prioritatea mai mare. Dacă întreruperea este servită de serviciul EI^{2}OS toate celelalte întreruperi sunt invalidate (nu pot fi acceptate întreruperi multiple).

Transfer DMA

Transferul prin DMA numit μDMAC este asemănător serviciului EI^{2}OS, având următoarele caracteristici:

- Se transferă direct date între un dispozitiv de I/O şi memorie
- CPU se opreşte în timpul transferului
- Sunt implementate 16 canale de transfer cu prioritate fixă, canalul 0 fiind cel mai prioritar
- Transferul începe la o cerere de întrerupere a dispozitivului de I/O
- După terminarea transferului o cerere de întrerupere de STOP este generată automat de controllerul DMA.

Regiştrii DMA sunt situați în zona I/O începând de la adresa 00009BH (nu au poziții succesive).

DCSR (Registrul de selecție canal, DMA Descriptor Channel Specification Register) (8 biți) realizează selecția canalului DMA. Fiecare canal DMA are asociat un dispozitiv de I/O.

DSR (Stare DMA, DMA Status Register) (16 biți) arată pentru fiecare canal dacă există o cerere de întrerupere ca urmare a unui transfer DMA complet.

DSSR (Stare de STOP a DMA, DMA Stop Status Register) (16 biți) arată pentru fiecare canal dacă transferul DMA s-a oprit datorită unei cereri de STOP de la un dispozitiv de I/O.

DER (Validare DMA, DMA Enable Register) (16 biți) arată pentru fiecare canal dacă cererea de întrerupere este procesată ca atare sau întreruperea generează un DMA la terminarea căruia se cere întrerupere (dacă se lucrează cu întreruperi sau DMA).

DDWR (Registru de descriptori, Descriptor Window Register). Desriptorii DMA sunt organizați ca 8 octeți x 16 canale, cu semnificații asemănătoare cu EI^{2}OS. Cei 8 octeți sunt:

Data counter (High) DCTH (8 biți)
Data Counter (Low) DCTL (8 biți)
Adresa I/O (Low) IOAH (8 biți)
Adresa I/O (High) IOAL (8 biți)
Registru de control DMA DMACS
Adresa curentă a bufferului (High) BAPH
Adresa curentă a bufferului (Medium) BAPM
Adresa curentă a bufferului (Low) BAPL

Descrierea funcționării DMA

- Un dispozitiv I/O cere o întrerupere pentru acces DMA
- Dacă bitul din DER validează cererea DMA se citesc din descriptor: adresa sursei transferului, adresa destinației transferului şi numărul de cuvinte de transferat.
- Se transferă datele de la I/O la memorie sau invers.
- După fiecare element transferat (Byte sau Word) se verifică dacă dispozitivul I/O a cerut STOP şi dacă da transferul se termină.
- Un flag este poziționat în DSR (stare DMA) şi se cere o întrerupere. Citirea regiştrilor dă informații despre modul de rezolvare al transferului DMA.

Observație: denumirea de µDMAC arată că transferul nu are toate proprietățile unui transfer DMA pentru că nu se pot transfera date din memorie în memorie.

Întreruperi externe

Interfața de transfer DTP (Data Transfer Peripheral) face legătura între un periferic extern şi CPU. Perifericul extern solicită o întrerupere sau face o cerere de DMA iar DTP transferă cererile către controllerul de întreruperi, sistemul EI^{2}OS sau DMA.Cererile de întrerupere pot fi : la nivel H, la nivel L, pe front crescător sau pe front căzător.

Regiştrii DTP:

ENIR (Validare întreruperi externe, External Interrupt Request Enable Register) pe 16 biți validează fiecare linie de întrerupere externă.

EIRR (Flaguri de întrerupere, External Interrupt Request Register), pe 16 biți. Bitul corespunzător unei linii de cerere de întrerupere devine 1 la apariția unei cereri, iar dacă bitul din ENIR validează cererea, aceasta este transmisă către CPU.

ELVR (Registrul de nivele, External Interrupt Level Register) ELVR0 (16 biți) şi ELVR1 (16 biți). Fiecare linie are alocați 2 biți pentru a stabili nivelul logic la acre se cere întrerupere (0,1, front crescător sau descrescător).

EISSR (Sursa de întrerupere, External Interrupt Source Register) este un registru de 8 biți din care 3 activi. Trei linii externe de întrerupere pot scoate MC din modul adormit. Registrul poate selecta unul din două seturi de trei linii. Primul set de trei linii este: de la CAN (P42), USART3 (P12) sau P32. Al doilea set este: P01, P02, P03. Portul 0 este folosit la magistrala externă, deci în acest mod pinii P01,

P02, P03 nu pot fi folosiți ca cereri de întrerupere, de aceea se comută pe al doilea set de trei linii. În modul de lucru cu magistrală externă pentru ieşirea din *stanby* se folosesc pinii de la CAN, USART sau P32.

Funcționare:

Dacă pe o linie externă se cere o întrerupere, dacă este validată de ENIR ea se transmite controllerului de întreruperi. Acesta selectează cea mai prioritară cerere (dacă sunt mai multe cereri simultan) şi o transmite CPU. CPU compară biții ILM din CCR cu cei ai cererii de întrerupere din ICR. Dacă nivelul cererii este mai mare decât cel indicat de ILM, CPU activează procesul de întrerupere după terminarea instrucțiunii în curs, figura 3.4:

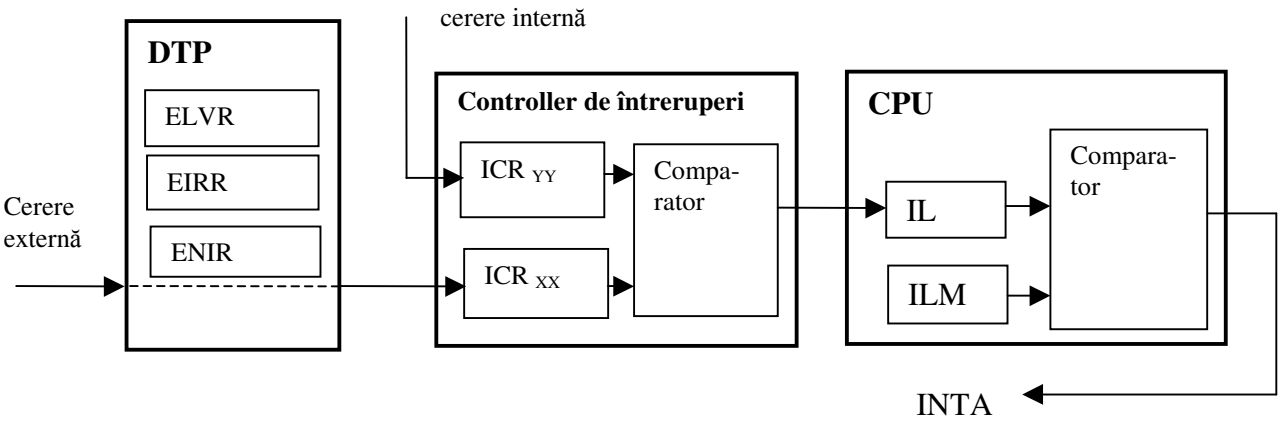


Figura 3.4: procesul de acceptare a unei întreruperi externe

Observații:

1.Perifericul care foloseşte DTP trebuie să inactiveze cererea de întrerupere în timpul a 3 ciclii maşină după încheierea transferului, altfel se va considera că a apărut o nouă întrerupere.

2.În registrul ICR se poate programa ca cererea de întrerupere să fie servită de sistemul EI^{2}OS.

3.3.Porturi de I/O

Structura simplificată a unui pin extern este dată în figura 3.5.

Nu sunt figurate blocurile care asigură semnificația dublă (de exemplu blocul de intrare analogică). Se observă posibilitatea de selectare pentru anumite porturi a nivelelor de intrare pentru mărirea siguranței în funcționare. În plus față de MC pe 8 biți apare posibilitatea de a opta pentru mai multe variante de nivele logice de intrare, cum ar fi nivelele logice auto sau nivelele TTL.

Regiştrii asociați porturilor de I/O sunt:

PDR (Registrul de date). Prin citirea PDR se pot citi datele de la intrare sau datele care au fost scrise în PDR, funcție de nivelul logic din DDR. Portul I/O are proprietatea de *Read Back*, se poate citi ceea ce s-a înscris în port. Scrierea datelor în PDR se face în latch, deci datele se mențin până la o nouă scriere în PDR.

DDR (Registrul de sens) (0-intrare, 1-ieşire)

DDRA (Registru de configurare a nivelelor de intrare, Input Level Setting Register pentru interfața serială LIN-UART). Doi biți stabilesc nivelul de intrare astfel: nivel auto şi TTL sau CMOS în legătură şi cu unii dintre biții din registrul ILSR.

PUCR0-PUCR3 (regiştrii de comandă a rezistențelor de pull up). La modelul 90350 4 porturi pot fi programate să aibă rezistențe de *pull up*.

ADER (registre de validare a intrărilor analogice). Există 2 regiştrii ADER5 şi ADER6. Inițial (după RESET) pinii sunt programați ca intrări analogice.

ILSR (Registru de selecție a nivelelor de intrare, Input Level Setting Register). Există 2 regiştrii ILSR0 şi ILSR1. În ILSR0 biții IL6-IL0 stabilesc nivelul de intrare pentru porturile 0-6 cu 0 nivel auto,

cu 1 nivel CMOS. În ILSR1 biții ILT0-ILT3 stabilesc nivelul de intrare pentru porturile 0-3, cu 0 nivel CMOS, cu 1 nivel TTL.

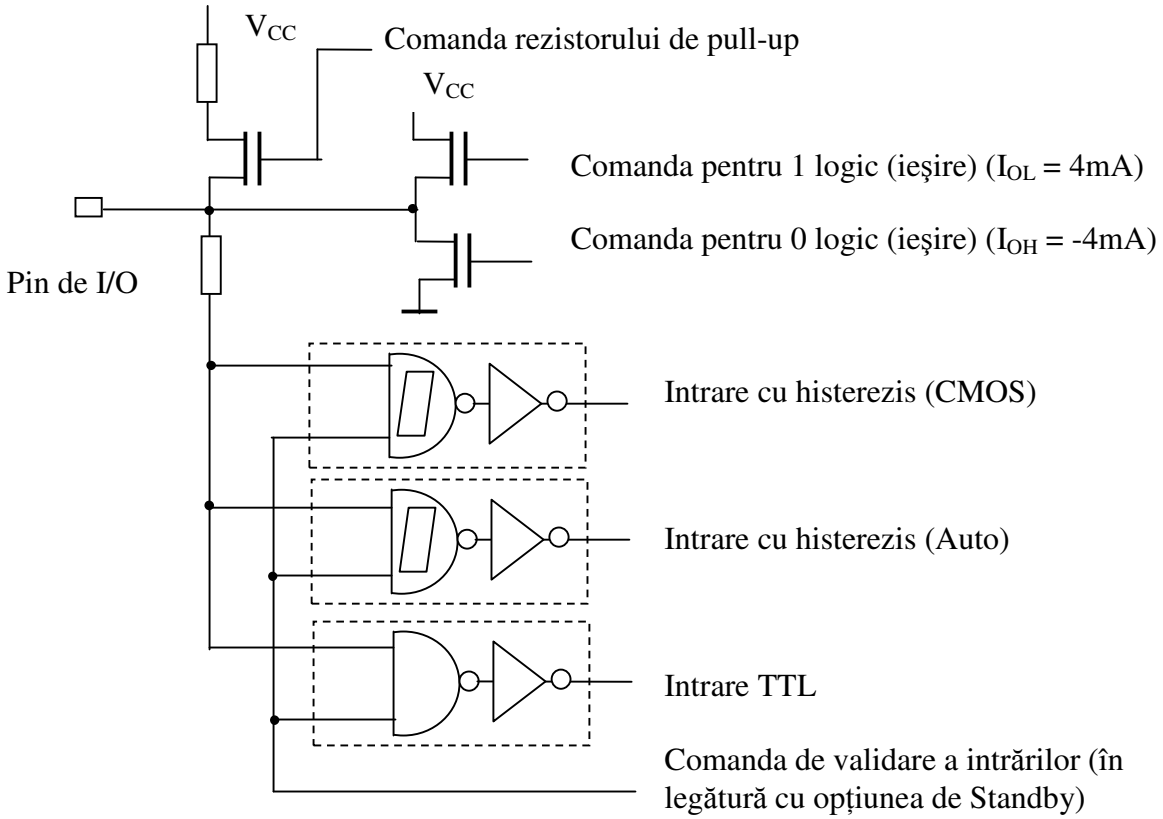


Figura 3.5: structura simplificată a unui pin extern

Observații:

1. Numărul de porturi de I/O diferă în funcție de model. Modelul 90350 are 7 porturi de8 biți.
2. Regiştrii de I/O sunt situați începând de la adresa 000000H.
3. Dacă la port este conectat un dispozitiv intern (la pinii cu semnificație dual) şi bitul de validare al dispozitivului este activ atunci portul este alocat acelui dispozitiv.
4. Față de MC pe 8 biți la care ILSR programa doar o linie în anumite porturi, la MC pe 16 biți se pot programa toate liniile din anumite porturi. De asemenea a crescut şi numărul de nivele de intrare posibile.
5. În modurile cu economie de energie (Standby) pinii intră în înaltă impedanță indiferent de DDR. Unele linii de intrare sunt invalidate prin legătura evidențiată în schema simplificată a unui pin de I/O. Dacă s-au programat rezistențe de *pull up*, liniile rămân la potențial ridicat.

3.4.Generarea tactului, moduri de lucru cu economie de energie

Terminologie:

- un tact intern care controlează operarea CPU şi a interfețelor interne se numeşte tact maşină (Machine Clock).
- un tact generat de oscilatorul exterior se numeşte tact (Oscillation Clock).
- un tact generat de PLL se numeşte tact PLL (PLL Clock).

Ca şi la MC pe 8 biți există un tact principal (de regulă 4MHz) şi un subtact (32KHz).

Schema bloc de generare a tactului:

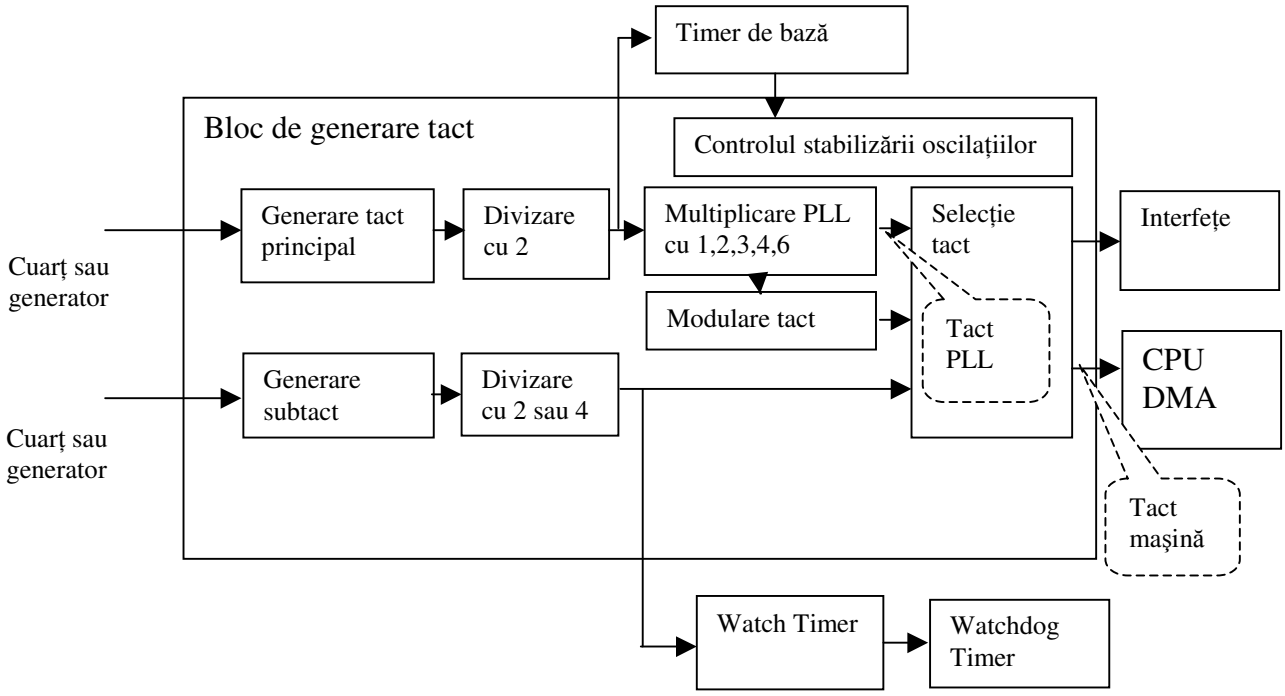


Figura 3.6: schema bloc de generare a tactului

Tactul maşină maxim la care pot funcționa interfețele şi CPU este de 24MHz. La un tact extern de 16MHz se poate defini un factor de multiplicare prin PLL de maximum 1.

Tactul înainte de multiplicare este folosit pentru Timerul de bază (tact principal) şi pentru Watchdog Timer (subtactul).

Regiştrii de comandă ai tactului şi moduri de tact

CKSCR (Registru de selecție a tactului) selectează factorul de multiplicare PLL 1,2,3,4 sau 6 împreună cu un bit din PSCCR, selectează tact principal sau subtact, timpul de stabilizare al oscilațiilor, selectarea tactului prin multiplicare PLL sau direct.

PSCCR (Registru de control PLL şi subtact, PLL/Subclock Control Register) selectează factorul de divizare al subtactului 2 sau 4 şi selectează factorul de multiplicare PLL înpreună cu 2 biți din CKSCR.

Moduri de tact posibile:

- Tact principal, se foloseşte tactul principal divizat cu 2, bucla PLL este invalidată.
- Mod PLL, se utilizează bucla PLL pentru multiplicare.
- Mod subtact, se utilizează subtactul divizat cu 2 sau 4.

Tranzițiile între modurile de tact se fac cu durate de timp bine stabilite şi programabile pentru ca oscilațiile să se stabilizeze.

Moduri de lucru cu economie de energie

Modurile de lucru ale MC pe 16 biți sunt:

Mod de operare normal (cu tact principal, subtact sau tact PLL)

Mod de operare intermitent (tact intermitent, subtact intermitent sau tact PLL intermitent)

Mod Standby (mod adormit, mod timer de bază, mod Watch şi mod STOP)

O diagramă a consumului de curent în aceste moduri este dată în figura 3.7:

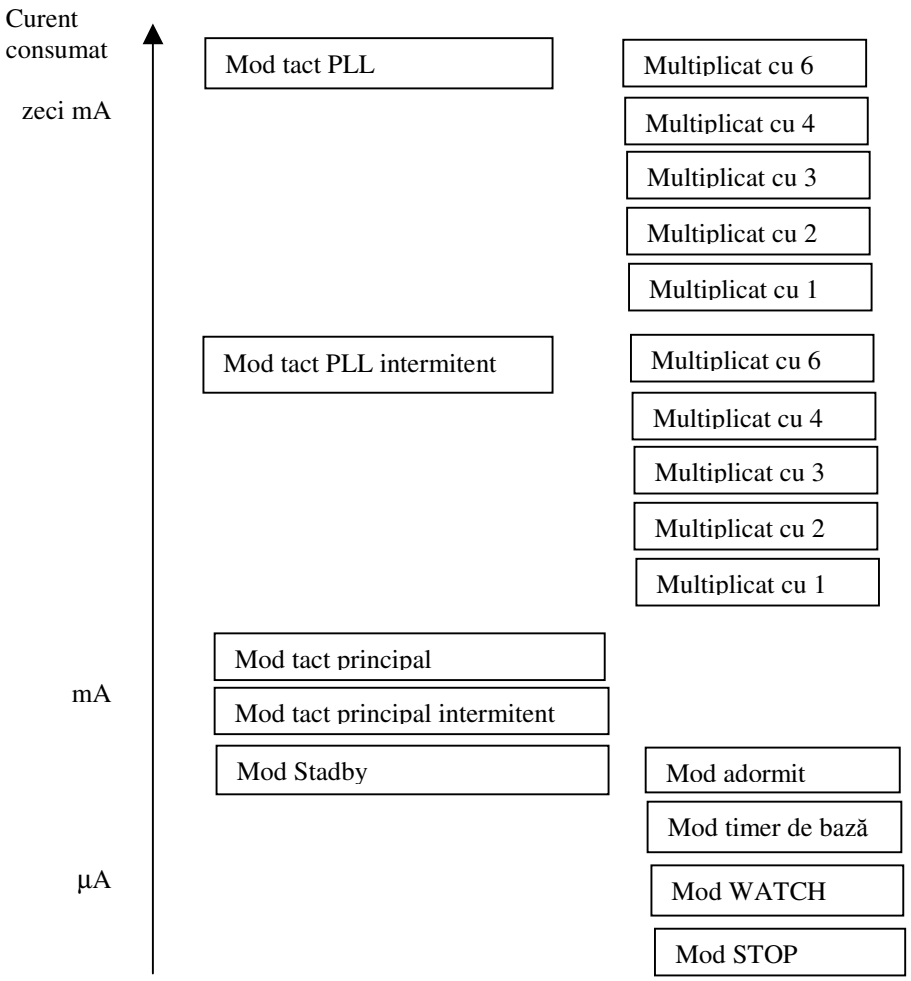


Figura 3.7: curentul consumat în diferite moduri de funcţionare

Modul intermitent înseamnă lipsa unor impulsuri de tact la CPU după execuţia unei instrucţiuni, interfeţele având tactul întreg.

În modul Standby se opreşte tactul către CPU (mod adormit) sau către CPU şi unele interfeţe (mod Timer de bază) sau se opreşte tactul (mod STOP). Modurile Standby sunt:

- Mod adormit PLL, se opreşte tactul PLL către CPU
- Mod adormit principal, se opreşte tactul principal către CPU
- Mod subadormit, se opreşte subtactul către CPU
- Mod Timer de bază, se opreşte tactul tactul către CPU şi interfeţe. Funcţionează doar Timerul de bază (cu tact principal) şi Watch Timerul cu subtact.
- Mod STOP, toate funcţiile sunt inactive.

Regiştrii de comandă ai modului economic

LPMCR (Registru de control al modului, Low Power Consumption Mode Control Register) selectează: numărul de tacte în care se opreşte CPU după efectuarea unei instrucţiuni (0, 8 16 sau 32), comanda intrării în mod STOP şi în mod adormit, se comandă starea liniilor externe în modurile WATCH, Timer de bază şi STOP (reţinerea ultimei valori logice sau trecerea în înaltă impedanţă). Un bit se poate poziţiona pentru iniţierea unui RESET.

Observaţie: ieşirea din modurile cu economie de energie se poate realiza cu un RESET extern, cu un RESET comandat din registrul LPMCR, iar în modurile în care funcţionează anumite interfeţe cu

cereri de întrerupere de la aceste interfețe sau cu cereri de întrerupere externe. O diagramă de timp pentru ieşirea din mod STOP inițiată de un RESET extern este dată în figura 3.8:

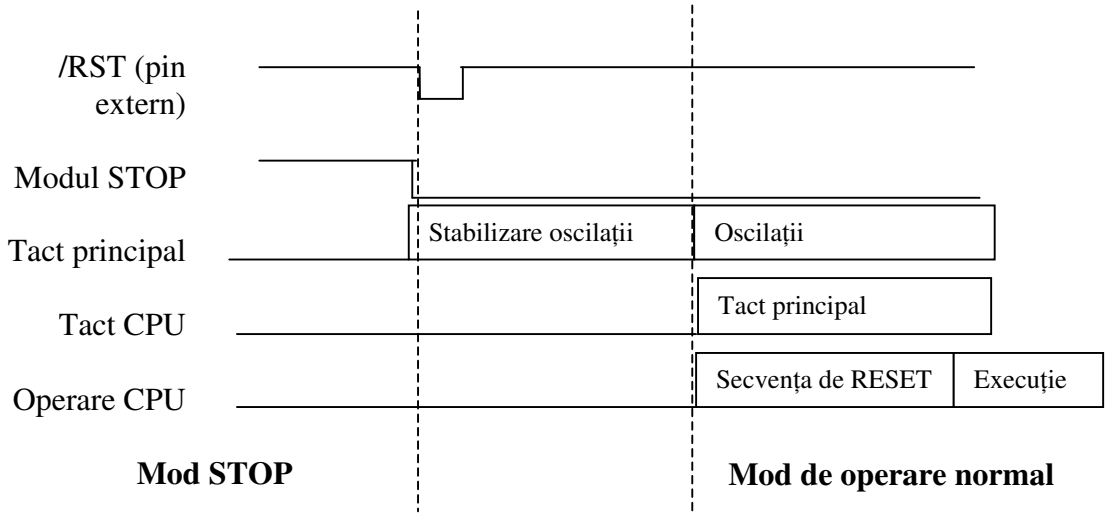


Figura 3.8: diagrama de timp pentru ieşirea din modul STOP

Modulatorul de tact

Are rolul de a reduce interferențele electromagnetice (EMI) prin împrăştierea spectrului semnalului de tact. Frecvența dată de tactul PLL este F_0 iar spectrul are un vârf corespunzător acestei frecvențe. Modularea tactului înseamnă variația frecvenței /fazei între 2 limite, ceea ce micşorează vârful spectrului.

Tactul de la bucla PLL poate fi modulat în 2 feluri:

- modulație în fază, semnalul modulator fiind un semnal triunghiular
- modulație în frecvență, semnalul modulator fiind un semnal pseudo aleator.

La modelul MB90350 tactul de la bucla PLL este modulat cu un semnal triunghiular. Comanda modulatorului se realizează cu registrul CMCR (Clock Modulator Control Register) care validează modularea.

Lățimea impulsului de tact modulat variază cu +/- 0,8ns, ceea ce înseamnă la frecvența maximă a tactului PLL de 24MHz o frecvență maximă a tactului maşină modulat de 25,45MHz. Interfețele care nu au tact PLL (Timerul de bază, Watchdog Timer) nu pot avea ca tact tactul modulat. De asemenea când se utilizează interfața CAN (mod de funcționare direct) nu se poate folosi tactul modulat.

Modelul MB90390 admite ambele variante de modulație. La modulația în frecvență variația între F_{max} şi F_{min} este definită în 7 grupe, 1 fiind varianta cu variația cea mai mică şi 7 cu variația cea mai mare.

Regiştrii de comandă sunt:

CMCR (Registru de control/stare) (8 biți) în care se validează/ invalidează modulația, se selectează modulația în fază sau frecvență. 3 biți arată starea modulatorului. După o comandă de modulare generatorul de tact se autocalibrează, ceea ce durează 64ms (4MHz). Starea modulatorului înseamnă: modulație în fază sau frecvență, generatorul este în autocalibrare sau activ sau modulația este inactivă.

CMPR (Registru de parametrii) (16 biți) programează limitele variației de frecvență sau de fază pentru generatorul de tact. Aceste limite sunt recomandate pentru fiecare model în funcție de frecvența de tact.

Observaţii

1.Familia MB90340 are implementat un supervizor de tact care înlocuieşte tactul sau subtactul cu un tact RC cu frecvenţa de 100KHz sau 2MHz (în principiu ca la MC pe 8 biţi).

2.Anumite modele (versiunea T) au implementat modulul de detecţie a tensiunii de alimentarea prea mici şi generarea unui RESET (în principiu ca la MC pe 8 biţi).

3.5.Funcţionarea cu magistrală externă

Modurile de acces la memorie sunt:

- Mod RUN (funcţionare normală)
  -fără magistrală externă (single chip)
  -cu magistrală externă pe 8 sau 16 biţi şi ROM intern
  -cu magistrală externă pe 8 sau 16 biţi şi ROM extern
- Mod de programare a FLASH-ului

Modul de acces este comandat de pinii externi MDx şi biţii Mx din cuvântul de mod.

Magistrala externă este formată din următoarele semnale:

AD07-AD00 pe portul P07-P00, 8 biţi de date multiplexate cu adrese

AD15-AD08 (16 biţi) sau **A15-A08** pe portul P17-P10, 8 biţi de date multiplexate cu adrese sau doar 8 biţi de adresă

A21-A16 pe portul P25-P20, 6 biţi de adresă. Nu este adresabilă spre exterior toată zona de adresare pe 24 de biţi.

ALE pe linia P30, Address Latch Enable

/RD pe linia P31, Read

/WR (8 biţi) sau /WRL (16 biţi) pe linia P32, Write

Port I/O (8 biţi) sau **/WRH** (16 biţi) pe linia P33, activ doar la magistrala pe 16 biţi, Write.

HRQ pe linia P34 cerere către MC de intrare în HOLD.

/HAK pe linia P35 răspuns de confirmare la cererea de intrare în HOLD.

RDY pe linia P36

CLK pe linia P37.

Zona de memorie alocată lucrului cu magistrala externă în cele 3 moduri de funcţionare (zona cenuşie este alocată adresării externe), figura 3.9:

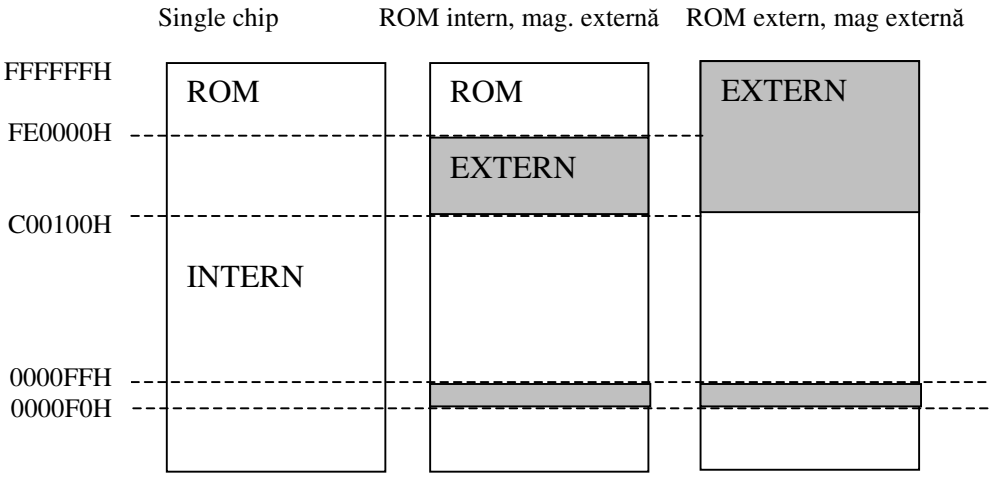


Figura 3.9: alocarea memoriei în cele 3 moduri de funcţionare

Regiştrii de comandă pentru lucrul cu magistrală externă

ARSR (Registru de selecţie a timpului de WAIT, Automatic Readz Function Selection Register) selectează inserarea a 1, 2 sau 3 stări de WAIT la accesul extern pentru zona 0000F0H-0000FFH (zona de I/O), selectează inserarea a 1, 2 sau 3 stări de WAIT la accesul extern pentru zona 800000H-FFFFFFH (zona superioară) şi selectează inserarea a 1, 2 sau 3 stări de WAIT la accesul extern pentru zona 008000H-7FFFFFH (zona inferioară).

HACR (Registru de control al adreselor externe, External Address Output Control Register) controlează liniile A21-A16. Liniile pot fi definite ca linii de adresă sau linii de I/O.

ECSR (Registru de control al magistralei, Bus Control Signal Selection Register) stabileşte dacă liniile au semnificaţia de: CLK sau I/O, RDY sau I/O, HRQ sau I/O, WR sau I/O şi lărgimea magistralei de date pe 8 sau 16 biţi.

Observaţie: în mod single chip toate liniile care se pot programa cu funcţii duale în aceste registre pot avea doar funcţia de linii de I/O. Registrele de programare a magistralei externe au printre altele şi rolul de a defini ca linii de I/O pinii nefolosiţi în transferul pe magistrala externă.

Diagrama de timp pentru un acces pe 8 biţi, un ciclu de citire urmat de un ciclu de scriere este dată în figura 3.10:

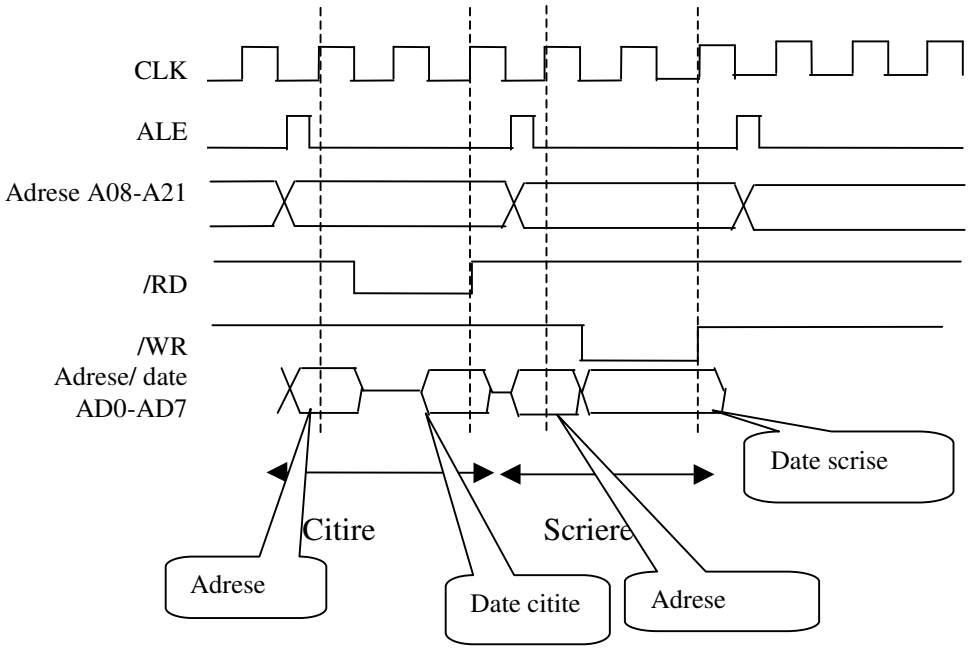


Figura 3.10: diagrama de timp pentru un acces pe 8 biţi

Se remarcă uşor că semnalele au o variaţie în timp ca la orice interfaţă paralelă standard (cu multiplexare).

3.6.Programarea memoriei FLASH

Memoria FLASH are capacitatea de 1Mbit (între 24K octeţi şi 512K octeţi funcţie de model) şi este mapată în segmentele FE şi FF. Srierea /ştergerea memoriei FLASH se poate face în următoarele moduri:

- paralel, cu un inscriptor extern specializat
- serial printr-o conexiune externă la PC sau cu un inscriptor specializat
- prin program

Câteva caracteristici ale memoriei FLASH:

- MC au implementat un algoritm automat de scriere (Embedded Algorithm)

- se poate semnaliza terminarea scrierii / ştergerii prin întreruperi sau prin flag-uri
- se pot realiza minimum 10 mii de scrieri /ştergeri. Se pot şterge sectoare individuale.
- ciclul de citire este de minimum 2 ciclii maşină (citirea unui cuvânt de 2 octeți, fără activarea algoritmului automat)
- datele sunt reținute cel puțin 10 ani.
- memoria FLASH nu necesită o sursă de tensiune suplimentară.

Memoria FLASH este împărțită în sectoare astfel:
SA4 (16K octeți) adresa superioară FFFFFFH
SA3 (8K octeți) adresa superioară FFBFFFH
SA2 (8K octeți) adresa superioară FF9FFFH
SA1 (32K octeți) adresa superioară FF7FFFH
SA0 (64K octeți) între FEFFFFH şi FE0000H

Operarea cu memoria FLASH prin program

Pentru a se putea scrie prin program în memoria FLASH, conținutul ei (unde este stocat programul care rulează) trebuie copiat în memoria RAM.

Registrul de comandă /stare este FMCS (FLASH Memory Control Status Register) care validează ca o întrerupere să fie cerută la terminarea scrierii / citirii / ştergerii, un flag arată starea memoriei (operare în curs sau operare terminată), un flag validează scrierea.

Sunt posibile 4 comenzi: Read/RESET, Write, Chip Erase şi Sector Erase. Fiecare dintre aceste comenzi înseamnă trimiterea unei secvențe de date care activează algoritmul automat (în afară de una dintre comenzile de citire). Pentru a vedea starea executării comenzii algoritmul automat oferă posibilitatea de citire a unui registru de satre format din 5 flaguri cu aceeaşi semnificație ca la MC pe 8 biți.

Citirea din FLASH se poate face normal (şi atunci durează 2 ciclii maşină) sau prin algoritmul integrat care se activează cu o anumită secvență de program.

Securitatea memoriei FLASH se poate realiza scriind 01H la adresa FE0001H, astfel se invalidează citirea memoriei din exterior. Singura operație admisă este ştergerea memoriei.

Programarea serială asincronă

Pinii externi MD2, MD1, MD0 se stabilesc la 1,1,0. Pinii portului 0, P01 şi P02 se stabilesc la 0,0 pentru transfer asincron şi 1,0 pentru sincron. Datele se transferă prin portul serial 3 (SOT3 ieşire, SIN3 intrare, SCK3 tact).

Programarea serială asincronă se realizează tipic cu un calculator PC AT, figura 3.11:

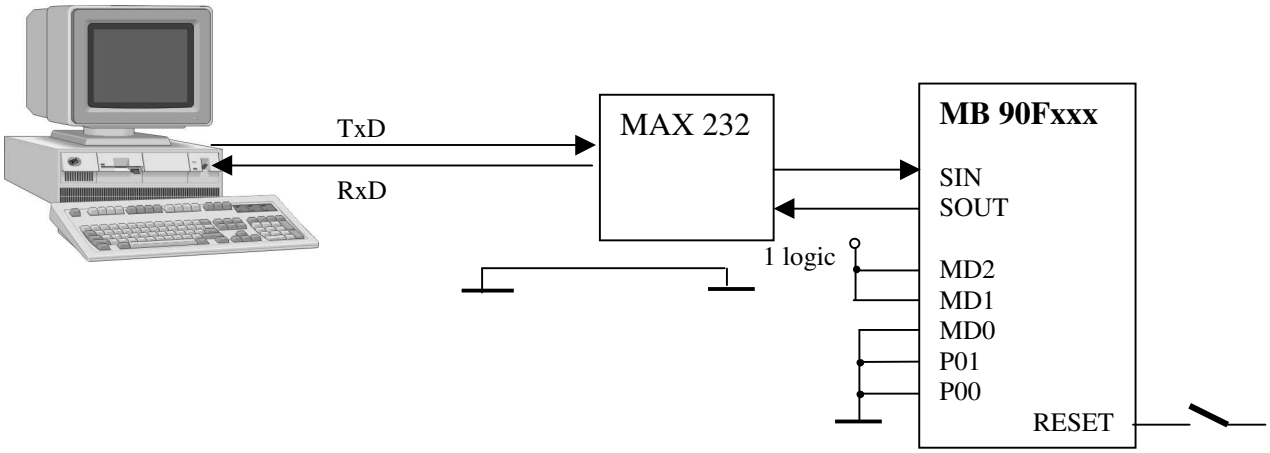


Figura 3.11: programarea serială externă

Pentru programare se foloseşte pe PC programul FLASH.EXE, asemănător cu cel de la 8 biţi, figura 3.12:

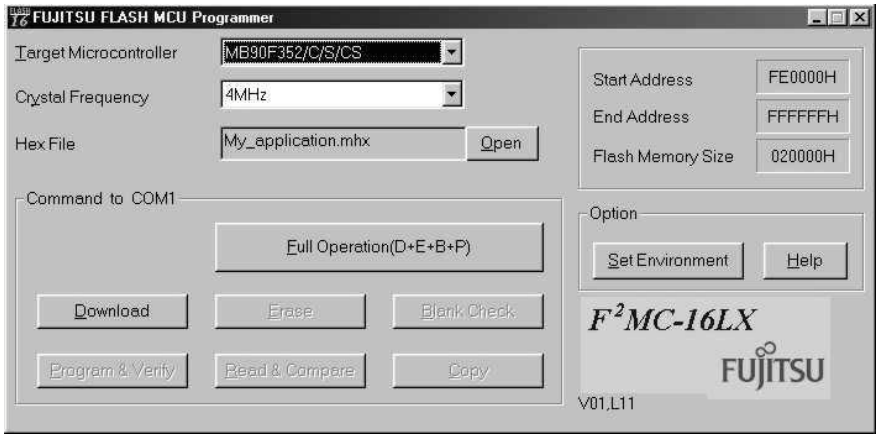


Figura 3.12: ecranul iniţial al programului FLASH.EXE

Mecanismul de programare serială implică următorii paşi:

- pinii externi se conectează la valorile logice necesare programării seriale asincrone
- se aplică un RESET necesar selecţiei modului de lucru programare
- în ROM-ul MC există o secvenţă de program numită BOOT LOADER care este activată după RESET în modul de lucru programare serială şi care încarcă în RAM prin legătura serială nucleul de programare pentru memoria FLASH (Programming Kernel) care conţine comenzile de programare: scriere, ştergere sector, ştergere memorie, verificare.
- se execută operaţiile cerute de utilizator: ştergere, scriere program utilizator, verificare.

Programarea serială sincronă

Se poate realiza cu un inscriptor dedicat, asa cum este cel recomandat de Fujitsu de la firma Yokogawa Digital Computer Corporation, figura 3.13:

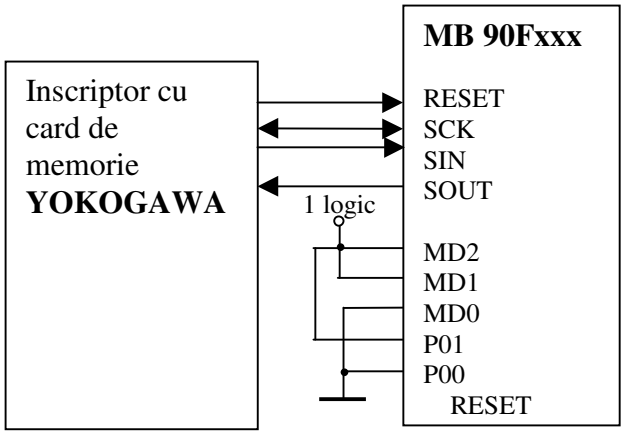


Figura 3.13: programarea serială sincronă

Inscriptorul Yokogawa înscrie datele serial sincron şi comandă RESET-ul la momentul potrivit. Datele se pot stoca în inscriptor într-un card de memorie.

Programarea paralelă

Programarea paralelă se foloseşte la cantităţi mai mari de MC, oferind o productivitate şi o viteză de programare mai bune. Programatorul recomandat de Fujitsu este GALEP 4, a patra generaţie de programatoare realizate de firma Conitec Datensysteme GmbH. Programatorul este universal (poate programa memorii de tip EPROM, EEPROM, FLASH, EEPROM serial şi MC) şi suportă lucrul cu tensiuni mici la pini (1,3V).

- Programarea paralelă se face prin magistrala externă cu câteva particularităţi:
- transferul de date se face pe 8 biţi iar adresarea pe 19 biţi
- pe portul 0 datele nu sunt multiplexate cu adresele
- la portul 3 semnalele au semnificaţii speciale

În acest mod de programare comandat de pinii MD2, MD1, MD0 stabiliţi la 1,1,1 memoria FLASH din MC se poate comanda ca orice memorie FLASH obişnuită printr-o magistrală paralelă. Accesul paralel dublu la memoria FLASH, atât din exterior (pentru programare paralelă) cât şi din interior (pentru programarea cu algoritmul integrat) se poate figura astfel, figura 3.14:

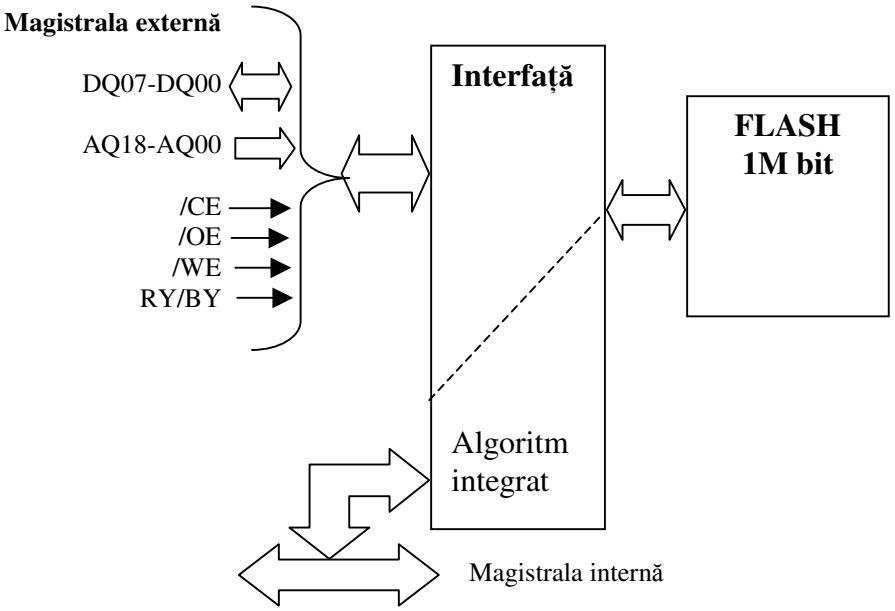


Figura 3.14: programarea paralelă

Alocarea acestor semnale la porturi este:
Port 0 DQ07-DQ00 date scrise sau citite pe 8 biţi
Port 1 AQ15-AQ08 adrese (octetul superior)
Port 2 AQ07-AQ00 adrese (octetul inferior)
Port 3

P30 AQ16 linia de adresă 16
P31 /CE validare circuit
P32 /OE validare ieşire
P33 /WE validare scriere (un impuls la fiecare octet scris)
P34 AQ17 linia de adresă 17
P35 AQ18 linia de adresă 18
P36 /BYTE=0, transferul se face pe octet
P37 RY/BY validare scriere (un impuls la sfârşitul scrierii ştergerii)

Operarea dual

Ca şi la MC pe 8 biţi memoria FLASH poate opera în mod dual, adică se poate şterge /scrie un sector în timpul rulării unui alt program, folosind sistemul de întreruperi. La operarea normală pentru a scrie date în memoria FLASH aceasta trebuie ştearsă şi apoi rescrisă. Înainte de ştergere trebuie ca BOOT LOADER-ul să fie mutat în RAM. În operarea dual memoria FLASH este împărţită în 2 bank-uri. Programul utilizator poate scrie date imediat în FLASH sau unul dintre bank-uri poate fi şters în timp ce programul utilizator rulează. Schema următoare arată diferenţa dintre operarea normală şi cea dual, figura 3.15:

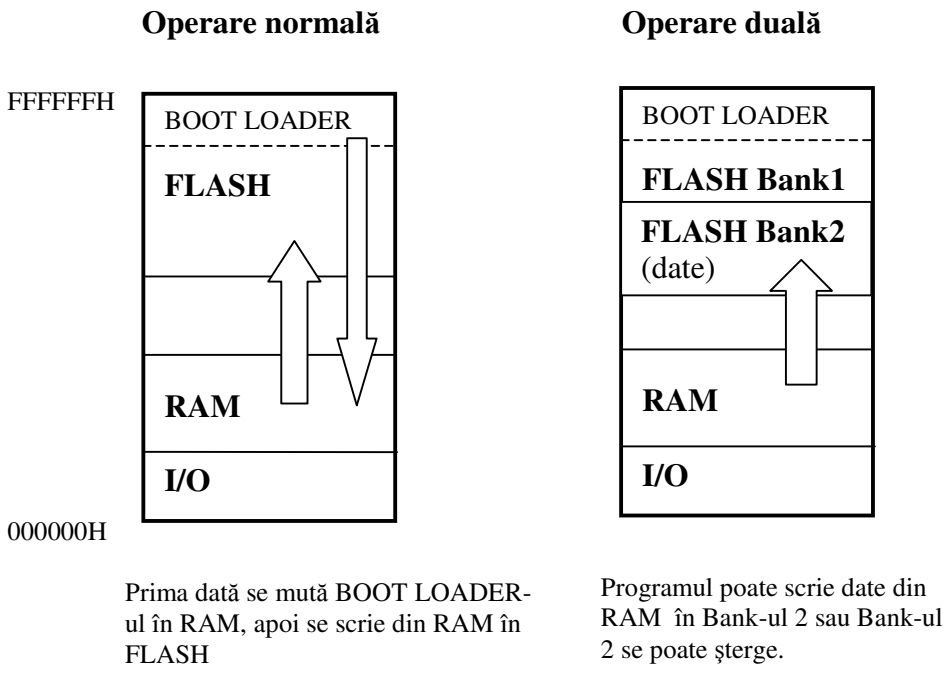


Figura 3.15: alocarea memoriei în programarea normală şi programarea dual

4.MICROCONTROLLERE PE 32 BIȚI (Familia RISC FR)

4.1.Structura CPU. Regiştrii. Maparea memoriei. Instrucțiuni. Comportarea la RESET. Moduri de operare.

Familia FR se bazează pe o arhitectură RISC cu structură Harvard, cu magistrale separate pentru instrucțiuni şi date. CPU conține un convertor Harvard-magistrală unică pentru că magistrala externă şi interfețele interne sunt conectate pe o magistrală unică de date şi instrucțiuni. Un alt convertor de magistrală transformă magistrala pe 32 de biți în magistrală pe 16 biți (formând magistrala R-Bus) necesară pentru transferul de date cu interfețele interne. Schema bloc este dată în figura 4.1:

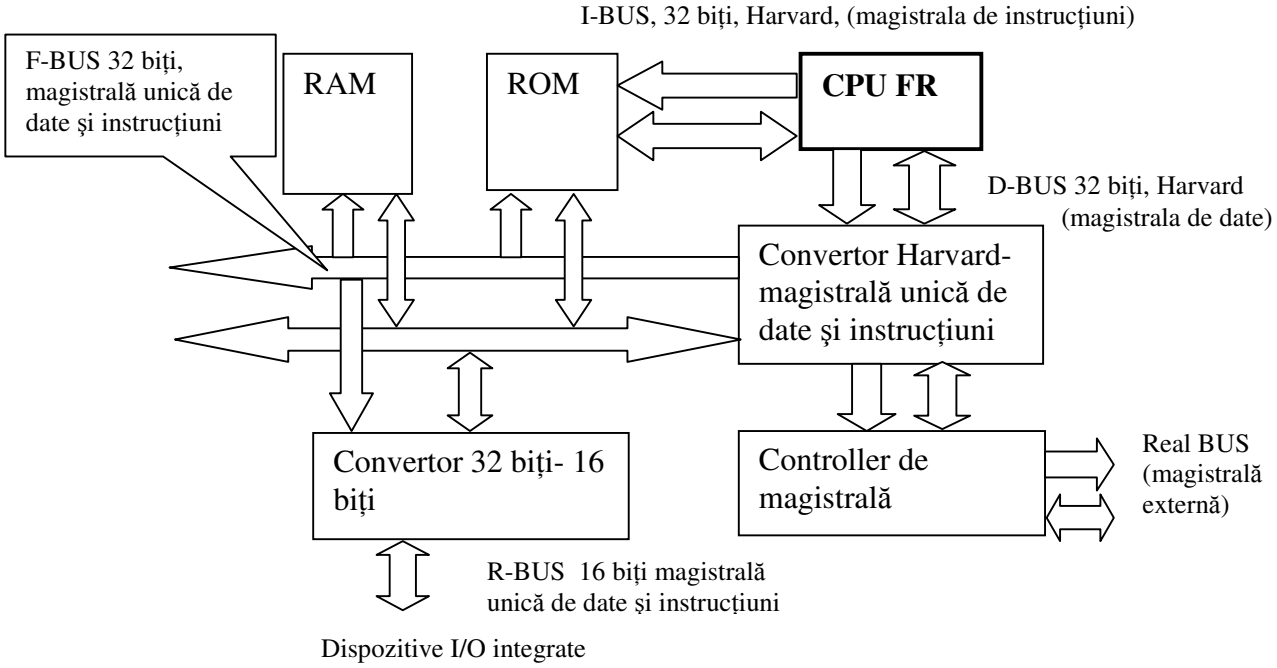


Figura 4.1: schema bloc a UC de tip FR

Memoria este conectată ori pe magistrala Harvard ori pe cea unică, funcție de model.

CPU conține o bandă (conductă) cu 5 etaje de execuție a instrucțiunilor (*pipeline*) pentru a putea executa o instrucțiune într-un ciclu.

Diagrama din figura 4.2 arată structura *pipeline*:

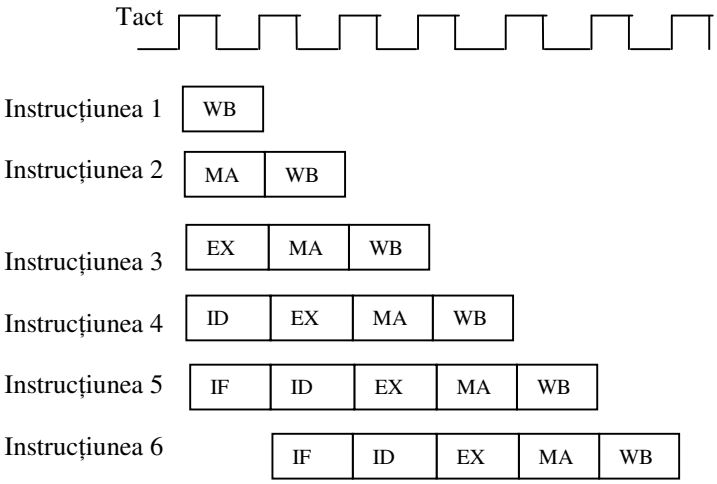


Figura 4.2: structura pipeline

IF- aducerea instrucțiunii (Instruction Fetch)
ID- decodificarea instrucțiunii
EX- execuția instrucțiunii
MA- acces la memorie
WB- scrierea rezultatului într-un registru (Write Back).

Instrucțiunile se execută în ordine, adică dacă instrucțiunea 1 intră în *pipe* înainte de instrucțiunea 2 atunci ajunge la scrierea rezultatului înainte de instrucțiunea 2.

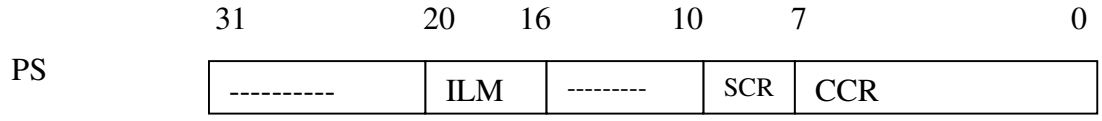
Un bloc important implementat în CPU este blocul de deplasare într-un tact (*Barrel Shifter*). În acest bloc datele se pot deplasa sau roti cu oricâte poziții într-un singur tact. Barrel shifter-ul este implementat ca un multiplexor, fiecare ieşire (fiecare poziție a bitului în registru) poate fi conectată la fiecare intrare (fiecare poziție a bitului în registru) depinzând de distanța de rotire. Această componentă a fost introdusă de Intel în procesoarele 386.

Regiştrii CPU

Regiştrii CPU sunt regiştrii dedicați şi regiştrii de uz general.

1.Regiştrii dedicați

PS (Starea CPU, Program Status) este un registru de 32 de biți cu configurația:



CCR (Condition Code Register) conține: bitul 5 (S) (Stack Flag) indică dacă SSP sau USP sunt folosite ca registre de uz general (R15), bitul 4 (I) validează întreruperile, bitul 3 (N) arată semnul rezultatului, bitul 2 (Z) indică un rezultat nul, bitul 1 (V) arată depăşirea, bitul 0 (C) arată un transport sau împrumut.

SCR (System Condition Register) conține 2 biți folosiți la împărțire iar unul este folosit în rularea pas cu pas supervizată de un emulator.

ILM (Interrupt Level Mask Register) are 5 biți care setează nivelul de acceptare a unei întreruperi.

PC (Program Counter) este un registru de 32 de biți care indică adresa instrucțiunii care se execută. Codul instrucțiunii este pe 16 biți şi se află la adrese pare. Un 1 pe poziția celui mai puțin semnificativ bit din PC (poziționat de o instrucțiune de salt) este invalid.

TBR (Baza tabelei, Table Base Register) este un registru de 32 de biți care arată prima adresă a tabelei de vectori în procesul EIT (Exception, Interrupt and Trap).

RP (Indicator de întoarcere din subrutină, Return Pointer) este un registru de 32 de biți care conține adresa de întoarcere din subrutină. La o instrucțiune CALL valoarea din PC se transferă în RP, iar la un RET valoarea din RP se transferă în PC.

SSP (System Stack Pointer) este un registru de 32 de biți şi conține adresa stivei sistem sau R15 (indicat de flagul S din CCR).

USP (User Stack Pointer) este un registru de 32 de biți care conține adresa stivei utilizator sau R15.

MDH/MDL (regiştrii de înmulțire şi împărțire, Multiply/Divide Register High/Low) sunt de 32 de biți. Pentru o înmulțire de 32x32 biți rezultatul este de 64 de biți şi se stochează în MDH/MDL. Pentru o înmulțire de 16x16 biți rezultatul se stochează în MDL. La împărțire deîmpărțitul se stochează în MDL şi se obțin câtul în MDL şi restul în MDH.

2.Regiştrii de uz general sunt 16 regiştrii de 32 de biți, R0-R15. Instrucțiunile extinse folosesc semnificații speciale pentru R13 (acumulator virtual), R14 (indicator de cadre) şi R15 (indicator de stivă).

Maparea memoriei este prezentată pentru varianta de emulare (V) şi o variantă cu FLASH la modelele MB 91V230 respectiv MB91F233, figura 4.3:

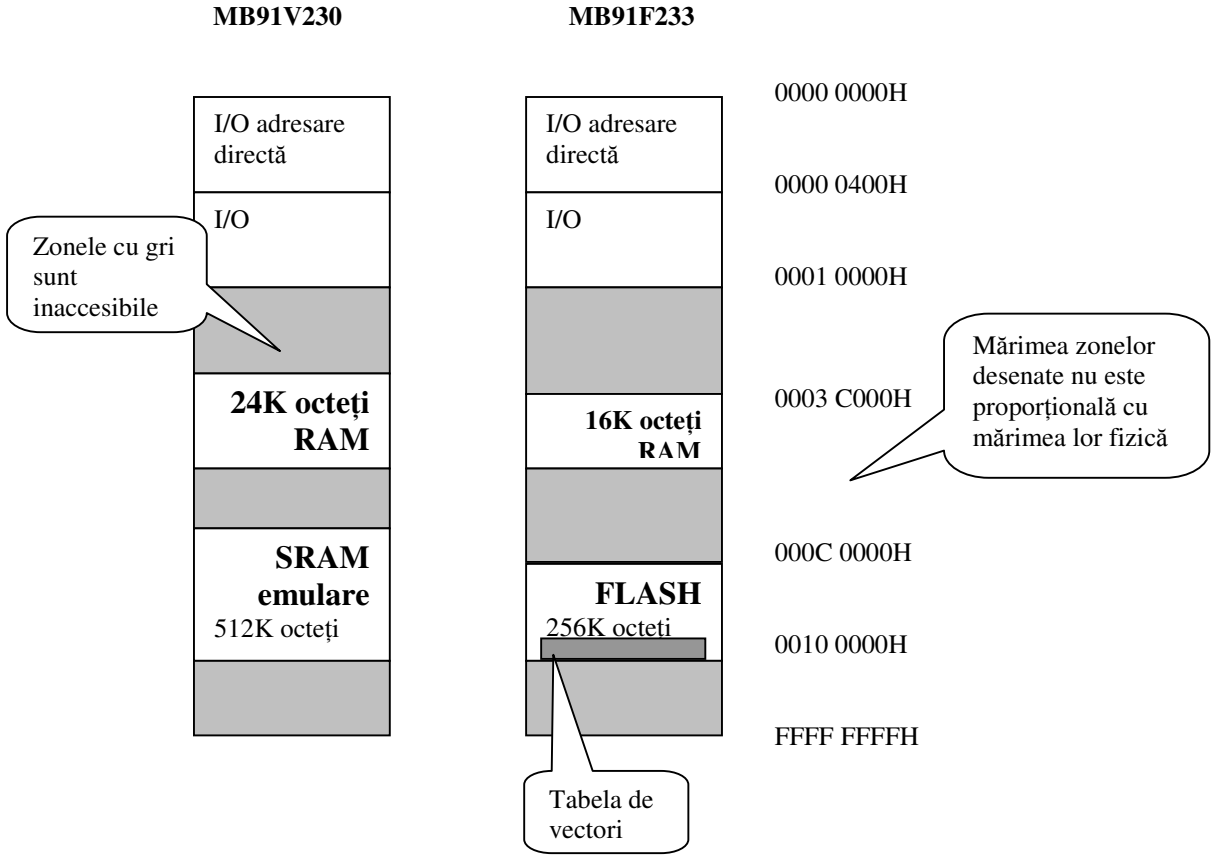


Figura 4.3: maparea memoriei

MB91360 are alocate adrese diferite pentru memoria RAM de instrucțiuni (conectată pe I-BUS), pentru cea de date (conectată pe D-BUS) şi pentru cea conectată pe magistrala Von Neumann (F-BUS).

În zona de adresare directă, spațiul I/O poate fi adresat direct, adică adresa poate fi specificată direct în instrucțiune.

Ordinea în cuvintele de date

Ordinea biților în octet este " little endian", adică cel mai semnificativ bit este pe poziția 31 iar cel mai puțin semnificativ pe poziția 0. Ordinea octeților în cuvântul de date este "big endian", adică cei mai semnificativi 8 biți sunt stocați la adresa n, următorii la n+1, următorii la n+2 şi cei mai puțin semnificativi la n+3.

Instrucțiunile fiind pe 16 biți ocupă adrese pare, adică bitul cel mai puțin semnificativ al PC este 0.

Datele pot fi:

- cuvinte pe 32 de biți (Word), adresa trebuie să fie multiplu de 4
- cuvinte pe 16 biți (Halfword), adresa trebuie să fie multiplu de 2
- octeți (Byte)

Instrucțiuni

Aritmetice sunt instrucțiuni standard de adunare, scădere, comparare, înmulțire şi împărțire. Adunarea şi scăderea pot fi realizate cu transport (Carry) pentru a ajuta operațiile pe mai mulți octeți şi pot fi fără poziționarea flagurilor pentru calcule de adrese. Înmulțirea şi împărțirea 32x32, 16x16 şi 32/32 se realizează cu regiştrii speciali ai CPU.

Exemple:

ADDN Rj,Ri are ca efect Ri+Rj cu rezultatul în Ri (cu regiştrii de uz general)
MUL Rj,Ri are ca efect RjxRi cu rezulatul în MDH, MDL (înmulțire 32x32)

Încărcare şi memorare pot fi pe 8, 16 sau 32 de biți[2]. Modurile de adresare pot fi directe sau indirecte. Adresările indirecte pot fi şi cu regiştrii care se auto incrementează sau decrementează.

Exemple:

MOV Rj,Ri[3] are ca efect Rj se încarcă în Ri
ST Ri,@Rj are ca efect Ri se stochează la adresa indicată de Rj (4 octeți)
STB Ri,@Rj are ca efect Ri se stochează la adresa indicată de Rj (1 octet)
LD @Rj,Ri are ca efect conținutul adresei indicată de Rj se încarcă în Ri (4 octeți)
LDI:32 #132,Ri data de 32 de biți (i32) se încarcă în Ri
LD @R15+, Ri conținutul adresei indicate de R15 se încarcă în Ri şi R15 se incrementează cu 4

Operații logice şi manipulări de biți . Se pot face operații logice ŞI, SAU, SAU exclusiv între regiştrii sau între regiştrii şi locații de memorie (sau locații de I/O) şi deplasări.

Exemple:

AND Rj,Ri ŞI LOGIC între Rj şi Ri, rezultatul fiind stocat în Ri
AND Rj,@Ri ŞI LOGIC între Rj şi conținutul adresei din Ri (4 octeți)
BANDL #u4,@Ri ŞI LOGIC între data pe 4 biți (u4) şi cei mai puțin semnificativi 4 biți din octetul care se găseşte la adresa indicată de Ri. Rezultatul se obține la adresa indicată de Ri
LSL Rj,Ri se realizează o deplasare la stânga a datei din Ri cu Rj biți şi se salvează rezultatul în Ri. Numărul de biți pentru deplasare este reprezentat pe maximum 5 biți în Rj

Adresare directă, este utilizată pentru transferuri între zona de I/O şi regiştrii de uz general sau memorie.

Exemplu:

---

[2] Instrucțiunile au următoarele sufixe: B cele care operează cu un octet, H cele care operează cu 16 biți (Halfword) şi fără sufix cele pe 32 de biți (Word)

[3] Sensul operării este invers față de MC pe 8, 16 biți sau procesoarele INTEL. De exemplu instrucțiunea ***MOV Sursa Destinație*** înseamnă că rezultatul operației se salvează în registrul destinație.

DMOV @dir10, R13 un cuvânt de 32 de biți de la adresa directă (dir10) se încarcă în R13

Instrucțiuni de salt, pot fi de 2 tipuri:

- Salturi cu întârziere
- Salturi fără întârziere

În salturile cu întârziere marcate cu sufixul :D instrucțiunea următoare celei de salt se execută înaintea saltului.

Exemplu:

ADD R1,R2 se adună conținutul lui R1 cu cel al lui R2

BRA:D eticheta instrucțiune de salt

MOV R2,R3 R2 se mută în R3. Instrucțiunea se execută înainte de execuția saltului

---

eticheta: ST R3,@R4 se salvează conținutul lui R3 (care conține rezultatul adunării) la adresa de memorie indicată de R4.

Dacă saltul este condiționat, următoarea instrucțiune se execută indiferent de condiția de salt. Dacă nu se doreşte execuția unei instrucțiuni trebuie pus un NOP.

Dacă de exemplu:

JMP:D @R0 salt la adresa conținută în R0

LDI:8 #0,R0 încărcare imediată a lui R0 cu 0 (instrucțiunea modifică adresa de salt)

Saltul se execută totuşi la vechea valoare a lui R0. Instrucțiunile care pot fi executate înaintea saltului trebuie să fie de un ciclu, să nu fie de salt şi să nu afecteze rezultatul în cazul execuției inversate.

Salturile fără întârziere se execută în ordinea firească: prima dată saltul şi ulterior următoarea instrucțiune.

Exemplu:

ADD R1,R2

BRA eticheta

MOV R2,R3 nu se execută înaintea saltului.

Comportarea la RESET

La un RESET microcontrollerul întrerupe orice program şi activitate hardware şi se inițializează. După revenirea semnalului de RESET în stare inactivă programul şi activitatea hardware încep de la o stare inițială.

Există 2 nivele de RESET extern, un RESET intern şi un RESET declanşat de intrarea în mod *standby*:

1 **INIT** (RESET-ul extern cel mai puternic) este numit RESET de inițializare, este declanşat de un nivel L la pinul extern INITX şi are ca efect:

- inițializează modul de lucru (cu magistrală externă sau *single chip*)
- inițializează toate setările tactelor
- inițializează toate setările magistralei externe
- inițializează setările pinilor
- inițializează setările care sunt inițializate şi de RST.

2.**RST** este numit RESET de operare, este declanşat de un nivel L la pinul extern RSTX şi inițializează:

- programul
- CPU şi magistrala internă
- valorile regiştrilor
- setările I/O
- setările semnalelor de selecție pentru zonele de adresare externe.

La un INIT se setează flagul INIT din registrul RSRR (registrul de surse de RESET), iar la un RST se setează flagul ERST.

3.**RESET soft** (Internal Power Down Reset) este echivalent cu un RST şi se obţine setând bitul SRST în registrul de control al modului *standby* (STCR). Se poziţionează bitul ERST ca şi la un RST hardware. Un RESET de la Watchdog este echivalent cu un INIT. După un astfel de RESET se poziţionează bitul WDOG în RSRR.

4.La intrarea în mod ***standby*****,** comandată de pinul extern HSTX se comandă un RESET de tip INIT. Se setează bitul HSTB din RSRR. Starea de *standby* este abandonată dacă apare un INIT.

Secvenţa de operaţii la RESET-ul de tip INIT:

- după nivelul L pe pinul INITX se aşteaptă un timp de stabilizare a oscilaţiilor tactului, stabilit cu 2 biţi în STCR
- reţine datele din STCR, opreşte tactul (se generează un RST)
- se citeşte vectorul de mod de lucru de la adresa 000FFFF8H şi se scrie în registrul de mod (MODR)
- se citeşte vectorul de RESET (adresa primei instrucţiuni) de la adresa 000FFFFCH şi se scrie în PC

Adresa de start a programului este prima adresă a Boot-ROM.

La un RST se citeşte doar vectorul de RESET.

Modurile de operare

- Mod single chip- nu se foloseşte memorie externă şi magistrală externă. Interfaţa CAN este localizată în memoria externă, deci nu se poate folosi în acest mod.
- Mod magistrală externă care poate fi:
  -ROM/FLASH intern
  -ROM/FLASH extern

Modurile de operare se selectează prin pinii externi MD2, MD1, MD0 şi registrul de mod MODR.

MODR (Mode Register) cu bitul 2 stabileşte modul de lucru cu ROM extern sau ROM intern. Biţii 0 şi 1 stabilesc modul de lucru *single chip* sau cu magistrală externă pe 8, 16 sau 32 de biţi.

Pinii externi MD2, MD1, MD0 stabilesc:

- 0,0,0 MODR şi vectorul de RESET sunt citite intern
- 0,0,1 MODR şi vectorul de RESET sunt citite de pe magistrala externă
- 1,1,1 mod de programare paralel al FLASH

În modul 0,0,0 se citesc valorile implicite pentru MODR şi vectorul de RESET care sunt: ROM/FLASH intern, CAN validat (acces extern).

Porturi de I/O

Regiştrii porturilor de I/O sunt de 3 tipuri:

Registru de date PDR

Registru de sens DDR

Registru de funcţii PFR

Registrul de funcţii PFR selectează pentru fiecare linie cu semnificaţie dublă una dintre funcţiile posibile.

De exemplu modelul MB91360 are 18 porturi de I/O de 8 biţi (dar nu toate porturile au 8 linii de I/O) notate astfel: port 7, port 8, port 9, port B, port G, port H, port I, port J, port K, port L, port M, port N, port O, port P, port Q, port R, port S, port T.

4.2. Funcționarea în întreruperi. Procesarea EIT. Modulul EDSU (Embedded Debug Suport)

EIT este un termen generic care înseamnă suspendarea execuției unui program în cazul apariției unui eveniment, care poate fi o excepție, o întrerupere sau o capcană (trap). Aceste evenimente sunt operații similare aplicate în condiții diferite şi implică terminarea execuției instrucțiunii în curs, salvarea informației în stivă şi saltul la o rutină de servire a cererii.

Excepția este un eveniment legat de contextul execuției unui program. Execuția se lansează la instrucțiunea care a cauzat excepția (şi nu la instrucțiunea următoare). Cauza unei excepții este instrucțiunea nedefinită.

Întreruperea apare independent de execuția programului, cauza fiind hardware. Execuția se lansează după terminarea instrucțiunii în curs.

Capcana este legată de contextul execuției programului, cauza fiind o instrucțiune. Execuția se lansează după instrucțiunea care a declanşat capcana. Instrucțiunile care lansează capcane sunt INT şi INTE.

Întoarcerea dintr-o rutină EIT se face cu instrucțiunea RETI.

Întreruperile pot fi interne/externe sau nemascabile (NMI).

Întreruperile interne/externe apar ca urmare a cererilor de la interfețele interne integrate sau de la pinii externi. Fiecărei cereri i se asignează un nivel de întrerupere cu registrul ICR (Interrupt Control Register). Registrul conține 4 biți prin care se poate aloca un nivel de prioritate al cererii de întrerupere fiecărei cereri de întrerupere. Valorile ICR sunt între 16H şi 31H.

O întrerupere este acceptată dacă:

- interfața internă sau pinul extern cer o întrerupere (lucru semnalizat prin poziționarea unui flag de cerere de întrerupere)
- bitul de validare din registrul de control al interfeței este activat
- valoarea priorității din registrul ICR este mai mică decât cea din ILM
- flagul I din CCR validează întreruperile.

După acceptarea unei întreruperi CPU realizează operațiile:

- conținutul PS este salvat în stivă
- nivelul întreruperii acceptate este stocat în ILM
- adresa (vector) întreruperii acceptate este stocat în PC.

Corespondența între sursa de întrerupere, registrul ICR corespunzător şi vectorul de întrerupere este:

Sursa de întrerupere	**ICR**	**Adresa ICR**	**Adresa vector**
IRQ00	ICR00	00000440H	TBR+3BCH
IRQ01	ICR01	00000441H	TBR+3B8H
------	-------	---------	---------
IRQ47	ICR47	0000046FH	TBR+300H

Registrul TBR (Table Base Register) de 32 de biți constituie adresa de început a unei zone de 1Koctet de vectori pentru EIT. Fiecare vector este de 4 octeți. Zona EIT implicită după un RESET este 000FFC00H până la 000FFFFFH.

Întreruperi nemascabile NMI au ca sursă cereri de la un pin extern (NMI). O întrerupere NMI este acceptată dacă :

- pinul extern NMI cere o întrerupere
- valoarea ILM este mai mare decât 15. Valoarea lui I nu contează.

După acceptare procesul EIT se desfăşoară la fel.

Excepții

Diferența față de întreruperi constă în faptul că saltul se face la execuția instrucțiunii care a creat excepția (instrucțiune nedefinită). Următoarea instrucțiune care se execută va fi instrucțiunea de după cea care a creat excepția. Rutina de servire poate executa şi o altă secvență, de exemplu un RESET.

Capcane

Cauzele unei capcane pot fi:

- instrucțiunea INT, de forma INT #u8 (#u8 adresare imediată pe 8 biți fără semn) este folosită pentru o capcană soft cu numărul întreruperii ca operator (u8).
- instrucțiunea INTE este folosită pentru o capcană soft pentru debugger, permițând astfel folosirea emulatoarelor.
- capcană pentru detecția urmei unui pas (step trace) este folosită de debugger. Pentru această capcană trebuie setat bistabilul T din SCR. Toate întreruperile sunt invalidate. O astfel de capcană apare după fiecare instrucțiune, fiind astfel posibilă rularea pas cu pas şi efectele efectele fiecărei instrucțiuni pot fi analizate.
- capcană care anunță că nu s-a găsit coprocesorul, apare când se folosesc instrucțiuni de coprocesor şi nu există coprocesor.
- capcană de eroare de coprocesor, indică o eroare într-o operație cu coprocesorul.

Întreruperile, excepțiile şi capcanele se execută la fel. Ca şi indicator de stivă se foloseşte SSP (prin înscrierea bitului S din CCR). Tabela de vectori indicată de TBR conține vectorii pentru întreruperi interne/ externe, NMI, excepții şi capcane.

Controllerul de întreruperi

Are următoarele funcții:

- Detectarea unei cereri de întrerupere sau NMI
- Evaluarea priorității
- Transmiterea către CPU a nivelului priorității
- Generarea unei cereri de abandonare a stării de HOLD.
- Asigură revenirea din modul de standby la generarea unei întreruperi.

Controllerul conține regiştrii **ICR** (Interrupt Control Register). Fiecare linie de cerere de întrerupere are alocat un registru ICR de 8 biți (ICR0-ICR47). În fiecare registru sunt activi 5 biți care codifică nivelul de prioritate al întreruperii. Valoarea inițială este 11111B (prioritatea cea mai mică sau altfel spus întrerupere invalidată). Dacă valoarea din ICR este mai mare decât cea stocată în ILM, întreruperea este mascată.. Întreruperea cu prioritatea cea mai mare este NMI (01111B în ICR), urmează întreruperea 10000B, 10001B .... Dezactivarea întreruperii înseamnă 11111B.

HRCL (Hold Request Cancel Level Register) defineşte nivelul la care este generată cererea de abandonare a stării de HOLD. Dacă se generează o întrerupere cu un nivel de prioritate mai mare (număr mai mic în ICR) decât nivelul stocat în HRCL se transmite o cerere de întrerupere care să genereze abandonarea stării de HOLD. În cazul unui transfer DMA, CPU este în stare de HOLD. Se poate ca o cerere de întrerupere să genereze abandonarea stării de HOLD, dacă are o prioritate mai mare decât cea stocată în HRCL.

Modulul de revenire din modul de *standby* asigură intrarea MC în modul de lucru normal la primirea unei cereri de întrerupere (inclusiv NMI)(cu altă prioritate decât 11111B). MC poate fi readus în modul de lucru normal de orice interfață internă prin întreruperi, dacă se programează în ICR un nivel mai mic decât 11111B.

Controllerul de întreruperi externe

Controlează pinii externi de cerere de întrerupere INT0-INT7 şi NMI. Se poate realiza detecţia nivelului H, L, front crescător sau front descrescător (mai puţin pentru NMI). Cererile de întrerupere pot fi folosite şi ca cereri de DMA.

ENIR (Enable Interrupt Request Register, de 8 biţi) controlează mascarea (validarea) întreruperilor, câte un bit pentru fiecare linie.

EIRR (External Interrupt Request Register, de 8 biţi) arată care linie a cerut întreruperea. Scrierea în acest registru (a valorii logice pentru întrerupere inactivă) poate anula cererea de întrerupere (eventual după achitare).

ELVR (External Level Register, 16 biţi) stabileşte pentru fiecare linie cu 2 biţi nivelul logic la care se face cererea de întrerupere (H, L, front crescător sau front descrescător).

Întreruperea NMI este detectată pe front descrescător iar în modul STOP pe nivel L.

Funcţionarea controllerului de întreruperi externe este foarte asemănătoare cu cea de la MC pe 16 biţi.

Modulul EDSU (Embedded Debug Suport)

EDSU este un modul implementat cu scopul de a oferi posibilitatea de depanare a programelor (debugging) în modul "*single chip*"şi este destinat utilizatorilor care nu doresc soluţia mai scumpă de ICE (In Circuit Emulation). EDSU se mai foloseşte în cazul în care dimensiunile fizice ale modulului cu microcontroller fac imposibilă utilizarea ICE (care este un dispozitiv auxiliar şi care se conectează cu modulul depanat).

Suportul hardware pentru EDSU este de fapt apariţia şi tratarea unei capcane software nemascabile. Partea software care interpretează capcana poate fi folosită pentru depanare, datele fiind transmise către calculatorul gazdă prin UART.

Sistemul EDSU pentru MB91360 conţine:

- 4 puncte de întrerupere la întâlnirea unei adrese de instrucţiune (2 puncte pot fi mascate)
- 2 puncte de întrerupere la întâlnirea unei adrese de operand (1 punct poate fi mascat)
- 2 puncte de întrerupere la întâlnirea unei valori de operand (1 punct poate fi mascat)
- puncte de întrerupere la întreruperi generate de resursele interne.

4.3.Descrierea DMA

Controllerul DMA şi funcţionarea transferului DMA este în mare parte diferită la aceste microcontrollere faţă de cele pe 16 biţi. Structura şi funcţiunile controllerului DMA sunt:

- 5 canale DMA independente
- 2 regiştrii pentru adrese pe 32 de biţi pentru fiecare canal
- registru de 16 biţi numărător al cuvintelor care se transferă, câte unul pentru fiecare canal
- numărător pe 4 biţi de blocuri, un registru pentru fiecare canal
- pini externi de cerere de DMA, DREQ0, 1, 2 şi pini externi de confirmare DACK0, 1 şi 2 (doar canalele 0, 1 şi 2) şi pini externi care indică terminarea transferului DEOP0, 1 şi 2.
- transferul unui cuvânt se face în 2 tacte
- se pot realiza transferuri independente prin mai multe canale
- prioritatea cea mai mare o are canalul 0 apoi 1, 2, 3, 4. Se poate schimba prioritatea între canalul 0 şi 1.
- cauzele unui transfer DMA poate fi o cerere din exterior (se poate programa să fie declanşată pe front sau pe nivel), o cerere de la o interfaţă internă sau o cerere soft
- modurile de transfer pot fi la cerere, în salvă, cuvânt sau bloc/ pas
- modurile de transfer pot fi pe 8 biţi, pe 16 sau pe 32

- adresa sursei şi destinației pot fi exprimate pe 32 de biți

Regiştrii DMA

DMACA (DMA Channel 0,1,2,3,4 control/ status Register A, registru de comandă/ stare a DMA A)(un registru de 32 de biți pentru fiecare canal) validează canalul, un bit comandă oprirea temporară a trnsferului DMA, un bit este folosit ca cerere de DMA (cerere soft), cu 5 biți se codifică sursa cererii de DMA, 4 biți arată mărimea blocului (număr de cuvinte/ bloc, 1-16 cuvinte) într-un transfer pe blocuri. DMACA conține registrul DTC pe 16 biți (Data Terminal Count Register) care programează numărul de blocuri care se transferă. Dacă se validează opțiunea de transfer cu reîncărcare, după terminarea transferului contorul DTC este reîncărcat cu valoare inițială.

DMACB (DMA Channel 0,1,2,3,4 control/ status Register B, registru de comandă/ stare a DMA B)(un registru de 32 de biți pentru fiecare canal), în care 2 biți stabilesc tipul transferului:

1.Modul de transfer în 2 cicluri în care se stabilesc adresele sursei în registrul DMASA, a destinației în registrul DMADA şi transferul se face cu numărul de blocuri specificate în contorul DTC.

2.Modul *fly-by* în care transferul are loc între o zonă de memorie externă şi o zonă I/O într-un ciclu.

2 biți stabilesc tipul transferului din alt punct de vedere:

1.Cuvânt sau bloc/ pas (se transferă un cuvânt sau un bloc la un ciclu de transfer DMA)

2.În salvă.

3.La cerere.

2 biți stabilesc lungimea cuvântului (8, 16 sau 32 de biți), un bit arată dacă adresa sursei din DMASA se incrementează sau decrementează după transferul unui cuvânt, un bit arată acelaşi lucru pentru adresa din DMADA, un bit arată dacă se lucrează cu reîncărcare pentru DTC sau nu, un bit arată dacă se lucrează cu reîncărcare pentru DMASA sau nu, un bit arată dacă se lucrează cu reîncărcare pentru DMADA sau nu, un bit validează întreruperea la întâlnirea unei erori, un bit validează întreruperea la terminarea transferului. Sunt codificate cauzele opririi transferului: terminarea normală, terminarea cu eroare, oprirea temporară, eroare de adresă). 8 biți formează registrul SASZ care specifică cu cât se incrementează /decrementează DMASA la un pas (0-255), 8 biți formează registrul DASZ care specifică cu cât se incrementează /decrementează DMADA la un pas (0-255).

DMASA0-DMASA4, DMADA0-DMADA4 regiştrii pentru adresa sursei şi adresa destinației. La începutul unui transfer DMA datele din aceşti regiştrii sunt stocate în numărătoare proprii ale controllerului DMA, apoi valorile se incrementează/ decrementează automat cu numărul specificat în DMACB şi se adresează astfel memoria. Sunt posibile şi transferurile între memorie şi memorie. După terminarea transferului cu numărul de cuvinte specificat în DMACA (DTC) valorile rămase în numărătoare se reînscriu în DMASA şi DMADA. Dacă se foloseşte opțiunea de reîncărcare valorile regiştrilor sunt refăcute cu valorile inițiale.

DMAC (Registru de control DMA global) are efect asupra tuturor canalelor. Un bit validează transferul DMA, un bit schimbă prioritatea şi face canalul 1 mai prioritar decât 0, un bit comandă oprirea temporară a transferului DMA.

Observație: Pentru specificarea adresei în mod *fly-by* se foloseşte adresa din DMADA (DMASA este ignorat). dresa indicată de DMADA trebuie să fie o zonă externă. Transferul propriu zis se desfăşoară la fel, pe blocuri cu lungime stabilită în DMACA şi numărul de blocuri în DTC (parte a lui DMACA) şi adresa se incrementează/ decrementează cu numărul stabilit în DMACB. Acest mod de lucru se poate folosi de exemplu la achiziția de date cu convertorul intern şi stocarea datelor într-o memorie externă.

Desfăşurarea unui transfer DMA

1.Inițierea transferului

a.Dacă transferul este inițiat de o cerere externă se pot folosi doar canalele 0, 1 şi 2. Se poate selecta nivelul şi frontul cererii de DMA (nivel H, L, front crescător sau descrescător) în modurile pe bloc sau continuu. Pentru transfer DMA la cerere se poate programa doar nivel H sau L.

b.De la o interfață internă cererea de întrerupere este interpretată ca cerere DMA dacă întreruperile sunt invalidate în ICR şi în DMACA este validată sursa care cere DMA (validarea surselor se face cu 5 biți).

c.O cerere soft se poate face prin scrierea unui bit în DMACA, separat pentru fiecare canal. Ca urmare poate începe un transfer DMA.

2.Transferul de date

a.Transferul în salvă în 2 cicluri înseamnă transferarea datelor continuu, adresa sursei fiind salvată pe 32 de biți în DMASA iar adresa destinației în DMADA. Transferul durează până la golirea DTC şi se face cu un număr de cuvinte = DTC x mărimea unui bloc. Dacă s-a setat reîncărcarea, regiştrii se reîncarcă cu valorile inițiale şi la următoarea cerere un nou transfer DMA poate începe.

b.Transfer *fly-by* în care zona de memorie de transfer este externă şi transferul poate fi doar între memorie şi I/O, în ambele sensuri. Se specifică doar adresa sursei sau destinației.

c.Transferul la cerere începe când un pin extern de cerere DMA are un nivel activ. Datele se transferă continuu până când cererea externă dispare (devine inactivă) sau până la golirea DTC.

d.*Fly-by* la cerere funcționează ca un transfer la cerere dar cu restricțiile de la *fly-by*.

e.Transfer pe un cuvânt / pas[4], se transferă un cuvânt (mărimea blocului programată în DMACA este 1)

f.Transfer de bloc / pas, se transferă un număr de cuvinte care formează un bloc (număr programat în DMACA, > 1).

g.Transfer pe un cuvânt sau bloc /pas *fly-by*, funcționează ca un transfer de un cuvânt sau bloc /pas cu restricțiile de la *fly-by*.

3.Terminarea transferului

a.Dacă DTC devine 0

b.Dacă se invalidează operarea DMA în DMAC (registrul de control global)

c.oprirea datorită unei erori.

d.Oprirea temporară.

Operarea în mod adormit

Dacă registrul DMAC este scris şi permite o operare DMA, se poate face transfer DMA în modul adormit. Revenirea din modul adormit se poate face de către întreruperea de terminare a transferului DMA.

4.4.Funcționarea cu magistrală externă

Caracteristici:

- Spațiul extern este adresabil cu 32 de biți fiind de 4GB.
- Se pot defini până la 8 bank-uri independente cu semnale de CS generate de MC. Bank-urile pot fi de minimum 64K şi pot ocupa orice poziție logică în aria externă.
- Magistrala de date poate fi pe 32/16/8 biți, lu lărgime care poate fi diferită pentru bank-uri diferite.
- Liniile nefolosite de magistrala externă pot fi folosite ca linii de I/O.

[4]Un pas înseamnă în acest context o operație DMA de la acceptarea cererii până la terminarea transferului.

- Se pot transfera date prin DMA cu operare de tip *fly-by*.

O schemă bloc a interfeței cu magistrala externă este dată în figura 4.4:

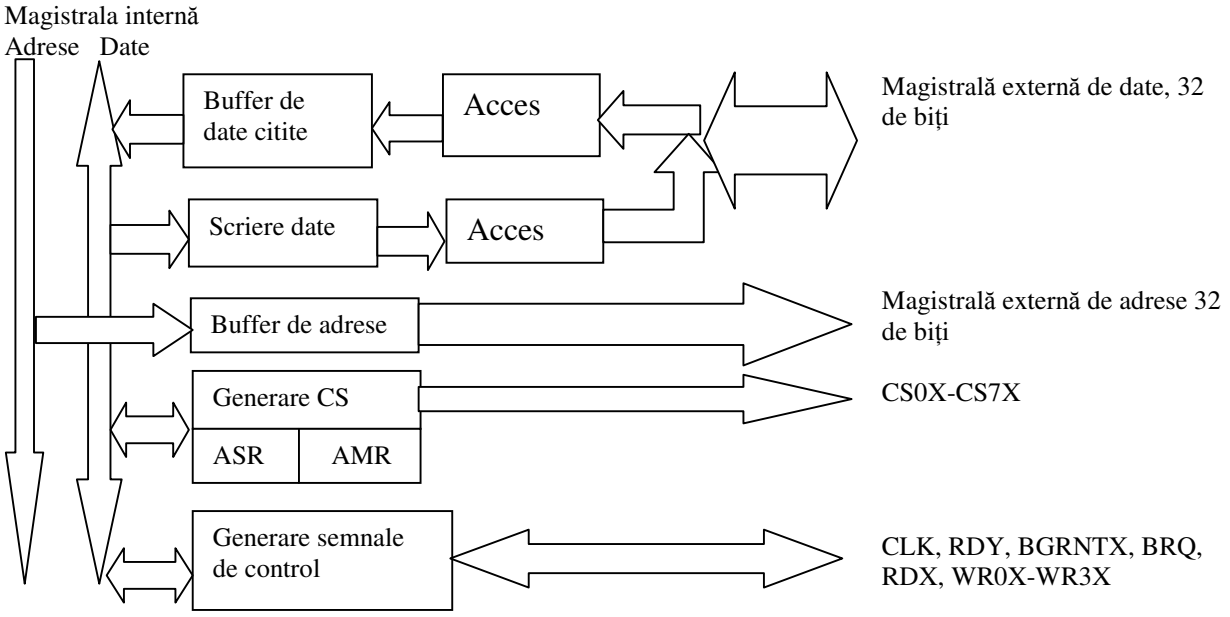


Figura 4.4: interfața cu magistrala externă

Regiştrii

ASR0-ASR7 (Selecția adresei bank-ului, Area Select Register)(16 biți)

AMR0-AMR7 (Registru de mascare, Area Mask Register) (16 biți). Aceşti regiştrii stabilesc zona alocată fiecărui bank (gama de adrese pentru care se activează pinul extern de selecție CSX0-CSX7). Regiştrii ASR specifică adresa zonei (parte H, cei mai semnificativi 16 biți).

Exemplu: ASR1=00000000 00000011B înseamnă că începutul zonei este 0003 0000H şi sfârşitul zonei este 0003 FFFFH (zonă de 64K).

Prin AMR se pot masca biții din ASR ca să se obțină o altă adresă de început, astfel: dacă biții sunt nemascați (valoare 0 în AMR) zona de început şi sfârşit se consideră dată de ASR. Dacă biții sunt mascați (valoare 1) începutul zonei se schimbă, biții mascați de AMR devenind din 1 în 0 în ASR.

Exemplu: ASR1=00000000 00000011B
AMR=00000000 00000010B — Se maschează bitul corespunzător şi se modifică adresa de început

Începutul zonei devine 0001 0000H iar sfârşitul zonei nu se modifică şi rămâne 0003 FFFFH.

Observații:

1.Zonele trebuie definite astfel încât să nu aibă suprapuneri.

2.După RESET se defineşte un singur bank (0) care ocupă toată aria externă.

AMD0-AMD7 (Registru de mod, Area Mode Register) (8 biți) specifică modul de acces pentru fiecare bank. Un bit validează intrarea RDY (funcționarea cu sincronizare), 2 biți stabilesc lărgimea magistralei de date (8/16/32), 3 biți programează inserarea automată de cicluri de WAIT (0-7, 7 este valoarea implicită).

CHE (Cache Enable) (8 biți) validează funcținarea cu cache pentru fiecare bank (fiecare bank are alocat un bit de validare în CHE).

CSE (Validare CS, Chip Select Enable Register) (8 biți) validează semnalele de CS. După RESET doar CS0 este activ.

Semnale de control

WR0X, WR1X, WR2X, WR3X sunt semnale de strob de scriere, cu legătură cu lărgimea magistralei de date. Astfel la transferul pe octet se activează WR0X, la un transfer pe 16 biţi WR0X şi WR1X, iar la transferul pe 32 de biţi se activează toate 4. Ordinea în care se transferă datele din regiştrii interni este "Big Endian", figura 4.5:

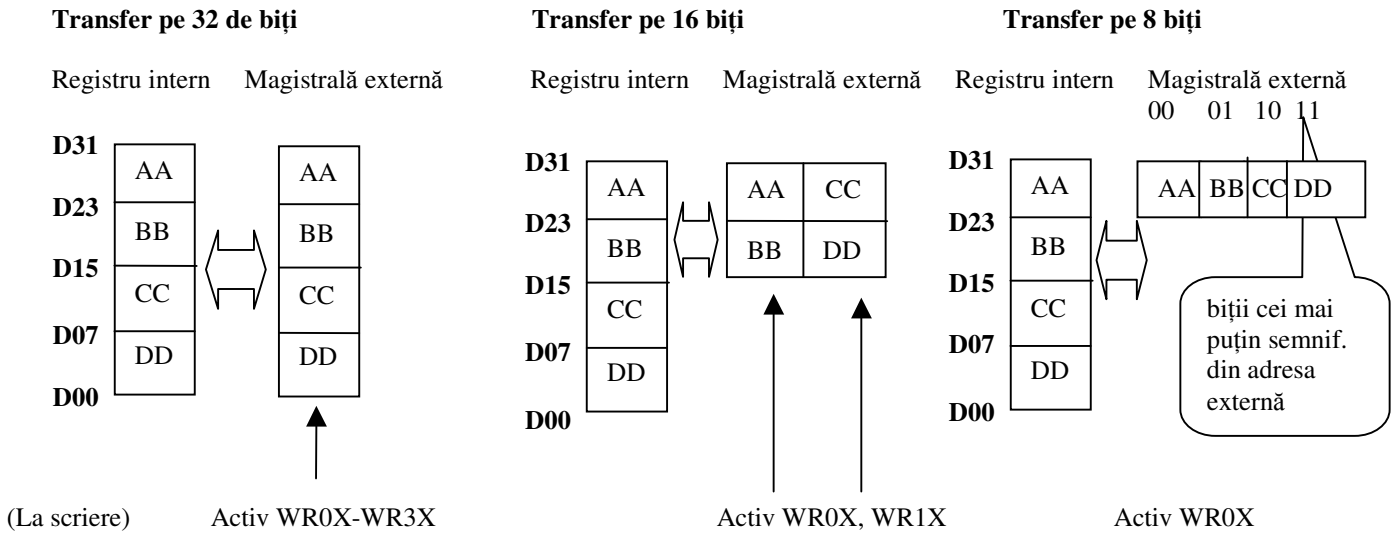


Figura 4.5: transferul de date prin magistrala externă

AS- strob de adrese
BRQ- cerere de magistrală
BGRNTX- acceptarea cererii de magistrală
RDX- strob de citire
RDY- semnal de sincronizare (Ready)
ALE- strob de memorare a adresei într-un latch extern (Address Latch Enable).

Diagrama de timp pentru un acces de 32 de biţi în 2 bank-uri diferite, un ciclu de citire şi un ciclu de scriere este dată în figura 4.6.

Între ciclurile de scriere / citire se introduc cicluri de sincronizare (a tactelor interne) numite *Idle* al căror număr depinde de tact şi de tipul de acces.

Pentru sincronizarea externă se pot introduce cicluri de WAIT programând corespunzător registrul WTC sau se poate utiliza pinul extern RDY.

Observaţie: modul de lucru cu interfaţa externă şi lărgimea magistralei se programează în registrul MODR, citit la pornirea microcontrollerului (discutat la "Comportarea la RESET").

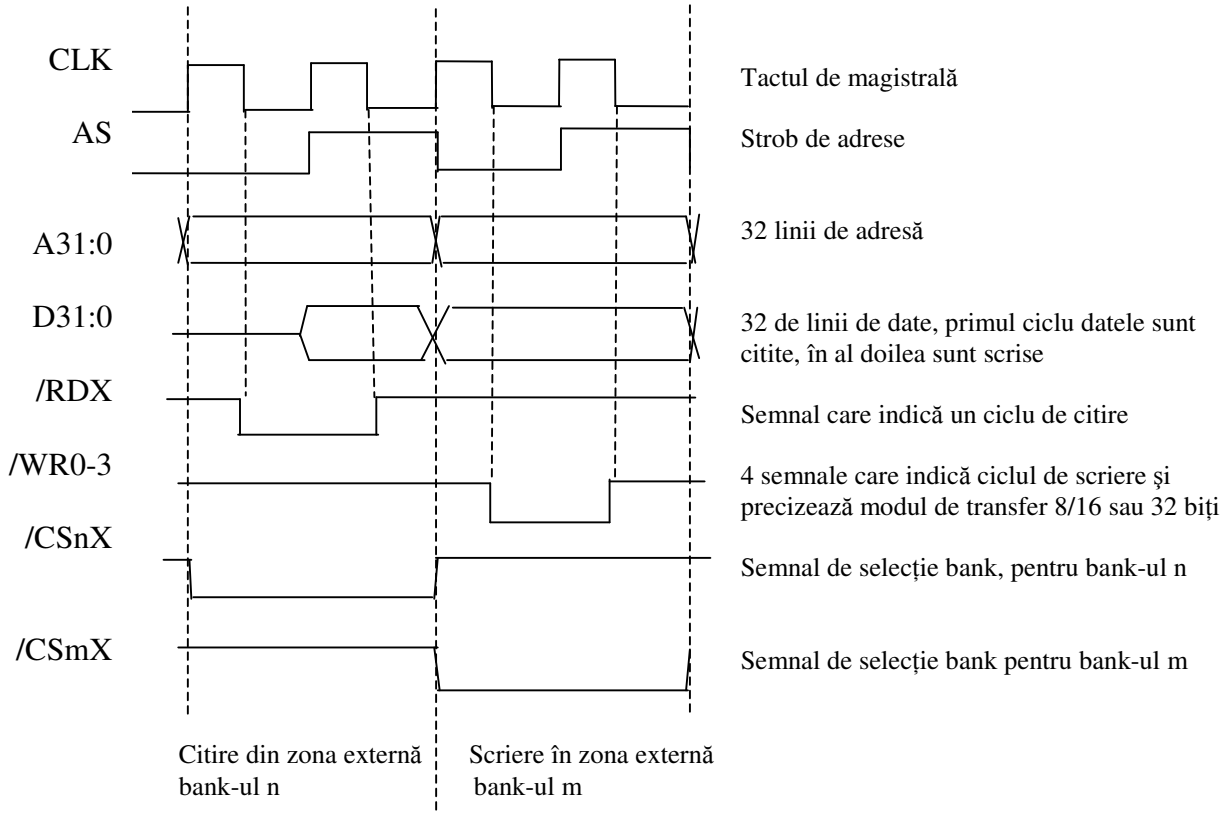


Figura 4.6: diagrama de timp pentru un transfer pe 32 de biţi

4.5.Modulul de căutare bit

Acest modul este folosit de sistemele de operare.

Regiştrii

BSD0 (Registrul de căutare 0, Data Register for Detecting Zeros) (32 de biţi) este accesibil doar la scriere.

BSD1 (Registrul de căutare 1, Data Register for Detecting Ones) (32 de biţi) este accesibil la scriere ca registru de căutare iar la citire ca registru de salvare /restaurare a stării modulului.

BSDC (Registrul de căutare modificări, Data Register for Detecting Change Points) (32 de biţi) este accesibil doar la scriere.

BSRR (Registrul rezultat, Detection Result Register) (32 de biţi), accesibil doar la citire. În acest registru se poate citi rezultatul căutării.

Căutare de 0

În datele stocate în BSD0 se caută primul 0 şi poziţia lui este scrisă în BSRR.

Exemplu: 000...111 în BSD0 duce la BSRR=0 (zecimal) (MSB este în stânga)
100...011 în BSD0 duce la BSRR=1 (zecimal)
11111111111111111111111111111111 în BSD0 duce la BSRR=32 (zecimal)

Căutare de 1

În datele stocate în BSD1 se caută primul 1 şi poziţia lui este scrisă în BSRR.

Exemplu: 010...111 în BSD1 duce la BSRR =1 (zecimal)
00000000000000000000000000000000 în BSD1 duce la BSRR=32 (zecimal)

Căutare modificări

În datele stocate în BSDC se caută primul bit diferit de MSB şi poziția lui este scrisă în BSRR.

Exemplu: 00100...000 în BSDC duce la BSRR=2 (zecimal)
11100...110 în BSDC duce la BSRR=3 (zecimal)

Salvarea / restaurarea stării modulului, necesar de exemplu dacă se lucrează cu întreruperi se face astfel:

- Se citeşte registrul BSD1 şi se salvează
- Se utilizează modulul de căutare bit
- Se restaurează registrul BSD1 din datele salvate

Astfel se restaurează starea modulului, inclusiv conținutul regiştrilor BSD0, BSD1 şi BSDC.

4.6.Generarea tactului, moduri de lucru cu economie de energie, subtactul, modulatorul de tact şi protecția la subtensiune

Schema bloc a generatorului de tact este dată în figura 4.7.

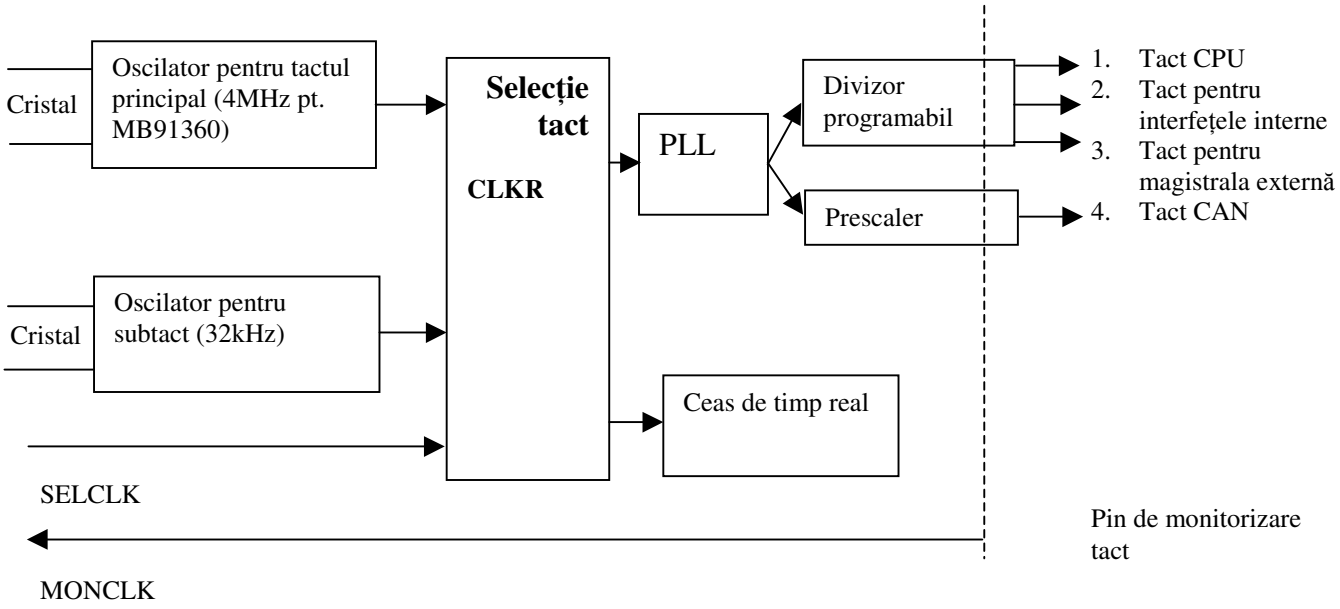


Figura 4.7: schema bloc a generatorului de tact

Se poate selecta folosirea tactului astfel (selecția se face cu registrul CLKR):

- tact principal divizat cu 2
- tact principal multiplicat cu bucla PLL

După stabilirea tactului şi al factorului de multiplicare pentru bucla PLL trebuie să se aştepte un timp de stabilizare al oscilațiilor.

Tactul se distribuie de la PLL formându-se mai multe tacte:

- tact CPU folosit de CPU, memoria internă, magistralele interne (I, D, X şi F), memoria cache de instrucțiuni, controllerul DMA, modulul de căutare bit.
- tact pentru interfețele interne folosit de magistrala pentru interfețe, controllerul de întreruperi (interne şi externe), porturile de I/O. Frecvența maximă este de 32MHz.
- tact pentru magistrala externă. Frecvența maximă este de 32MHz.
- tact pentru CAN, controlat de registrul CMCR.

- tact pentru ceasul de timp real. Pinul extern SELCLK selectează tactul principal sau subtactul pentru ceasul de timp real.

Pinul extern MONCLK oferă posibilitatea de a monitoriza în exterior ceasul intern Se poate selecta cu un registru de comandă care tact să fie monitorizat în exterior.

Tactul pentru CAN nu este comun cu tactul pentru celelalte interfețe deoarece de multe ori este supus unor cerințe speciale.

Regiştrii

CLKR (Sursa tactului, Clock Source Control Register) selectează factorul de multiplicare PLL (4,6,8,10,12,16), validează sau nu bucla PLL, selectează sursa tactului (tact principal divizat cu 2 sau multiplicat prin PLL).

DIVR0 şi DIVR1 stabilesc factorii de divizare pentru fiecare tact în parte astfel:

- tactul CPU se poate diviza cu 2,3,4,...,16 (selecția se face cu 4 biți în DIVR0)
- tactul interfețelor se poate diviza cu 2,3,4,...,16 (selecția se face cu 4 biți în DIVR0)
- tactul magistralei externe se poate diviza cu 2,3,4,...,16 (selecția se face cu 4 biți în DIVR1)

Moduri de lucru cu economie de energie

1.Mod de operare normală (RUN) în care sunt alimentate toate blocurile MC şi toate tactele sunt active.

2.Mod adormit (SLEEP) în care se opresc: CPU, memoria cache de instrucțiuni şi cea de date, modulul de căutare bit, magistralele interne şi externe. Funcționează generatorul de tact, controllerul de întreruperi, interfețele interne, controllerul DMA. Intrarea în mod adormit se face setând bitul SLEEP în registrul STCR (Standby Control Register). Ieşirea din modul adormit se face la apariția unei cereri de întrerupere validă, un INIT, un RST sau o cerere externă de Standby care comandă intrarea în modul Standby.

3.Mod STOP (Mod RTC) în care se opresc: circuitele de tact selectate pentru a se opri şi toate celelalte circuite. Pinii externi pot fi programați să treacă în înaltă impedanță sau să-şi păstreze valoarea logică. Ieşirea din modul STOP se face la apariția unei întreruperi externe (cu nivel H) validă, un INIT, un RST sau o cerere externă de Standby.

4.Mod Standby hardware în care se intră dacă un nivel L este aplicat pinului extern HSTX. Se opresc toate circuitele şi tactele iar pinii de ieşire trec în înaltă impedanță. Ieşirea din modul Standby se face la aplicarea unui nivel H pe pinul HSTX sau un INIT.

STCR (Standby Control Register) controlează modurile de lucru cu economie de energie şi are structura:

- bitul 7 (STOP) comandă intrarea în mod STOP (are prioritate mai mare decât bitul 6 (SLEEP), aşa încât dacă se cere simultan un STOP şi un SLEEP, MC intră în stare STOP.
- bitul 6 (SLEEP) comandă intrarea în mod adormit
- bitul 5 (HIZ) comandă păstrarea valorii logice a pinilor de ieşire la valorile dinainte de STOP sau intrarea în înaltă impedanță
- bitul 4 (SRST) lansează un RESET software
- biții 3 şi 2 stabilesc timpii de stabilizare după un INIT sau la întoarcerea din modul STOP
- bitul 1 stabileşte dacă subtactul se opreşte sau nu în mod STOP
- bitul 0 stabileşte dacă tactul se opreşte sau nu în mod STOP.

Subtactul

Ideea modului de lucru cu subtactul este de a asigura tactul ceasului de timp real (RTC) folosind subtactul de 32kHz, în timp ce restul MC are tactul de 4MHz. Acest lucru permite obținerea unor

moduri de lucru cu economie de energie în care se obține o putere consumată foarte mică. Stabilirea folosirii tactului sau subtactului pentru RTC se realizează cu pinul extern SELCLK.

Un tabel arată dependența între puterea consumată şi diferitele moduri de lucru în cazul în care se lucrează cu subtact la RTC:

Mod	**Putere disipată**	**Componente care operează**				
		Osc 4M	**Osc 32k**	**RTC**	**CPU şi interfețe**	**PLL**
RUN	mare	X	X	X	X	X(-)
RTC 4M32k	medie	X	X	X	-	-
RTC 32k	scăzut	-	X	X	-	-
STOP	cea mai scăzută	-	-	-	-	-

Din tabel se remarcă modul de lucru RTC 32k în care puterea consumată este mai mică decât în mod RTC 4M din cauza tactuluimai mic.

În modul RTC 4M32k operează ambele tacte (tactul pentru RTC este subtactul), fiind selectabile soft.. În acest fel se îmbunătățeşte comportarea CAN (scade timpul de stabilizare al oscilațiilor) pentru că o comutare la tactul de 4M este mai rapidă.

Modul SLEEP este acelați cu modul RUN doar CPU este oprită (acest mod nu s-a trecut în tabel).

Stabilirea acestor moduri de lucru se face cu regiştrii STCR şi CLKR.

Calibrarea subtactului

Subtactul de 32kHz se poate calibra cu ajutorul tactului de 4MHz pentru a mări precizia subtactului. Modulul de calibrare conține 2 timere, unul funcționând cu subtact iar celălalt cu tact. Timerele pornesc deodată şi timerul de 32kHz opreşte timerul de 4MHz, valoarea fiind stocată într-un registru. Această valoare este folosită pentru a calcula setările din RTC.

Regiştrii folosiți la calibrare:

CUCR (Calibration Unit Control Register) (8 biți) porneşte/opreşte calibrarea, validează/invalidează întreruperile, indică sfârşitul calibrării.

CUTD (32kHz Timer Data Register) (16 biți) stochează o valoare care indică durata calibrării

CUTR (4MHz Timer Data Register) (24 biți) stochează rezultatul calibrării.

Dacă perioada de calibrare este de 1s, se numără 4.10^6 tacte de 4MHz şi acuratețea obținută este de 0,25ppm.

Modulatorul de tact (MB91360)

Scopul modulatorului de tact este de a reduce interferențele electromagnetice (EMI) prin împrăştierea spectrului semnalului de tact. Semnalul de tact generat de PLL este modulat cu un semnal pseudoaleator. La tactul modulat se poate modifica gradul de modulare şi rezoluția de variație a frecvenței între Fmin şi Fmax. Se pot programa 3 grade de rezoluție. Modulatorul poate lucra cu tact între 16MHz şi 48MHz, de aceea circuitul de modulare trebuie calibrat. Calibrarea poate fi lansată hardware sau software. Semnalul de tact poate fi observat în exterior la pinul MONCLK.

Modulul CAN nu poate lucra cu tact modulat de aceea tactul este furnizat direct printr-un circuit de prescalare.

Regiştrii de comandă:

CMCR (Control Register). În CMCRL un bit validează modulatorul, un bit validează linia MONCLK, un bit porneşte calibrarea, un bit inițializează generatorul de numere aleatoare, un bit validează prescalarea pentru CAN. În CMCRH se stochează factorul de divizare pentru prescalarea CAN.

CMPR (Modulation Parameter Register) determină gradul de modulare (limitele de variaţie a frecvenţei Fmin şi Fmax).

CMSL0-3 (Frequency Resolution Setting Register) determină rezoluţia variaţiei frecvenţei. De regulă rezultatele cele mai bune se obţin cu maximul gradului de modulare şi maximul rezoluţiei. O rezoluţie mai mare înseamnă mai multe componente spectrale (şi de valori mai mici) în spectrul semnalului modulat.

CMLT0-3 configurează generatorul de numere pseudoaleatoare. CMLT1 conţine registrul de observare în care se poate programa care dintre liniile de tact să fie observabilă la linia MONCLK: tactul modulat, tactul PLL nemodulat, tactul CAN oscilatorul de 4MHz etc.

CMAC (Automatic Calibration Reload Timer Value) se încarcă cu valoarea la care atunci când se ajunge se iniţiază o calibrare. Este de fapt un timer pentru calibrare.

De exemplu, la gradul de modulare 1 şi o rezoluţie mică (rezoluţia poate fi mică, medie sau mare) tactul variază între 15 şi 17MHz, iar la o rezoluţie mare între 16 şi 42MHz. Cea mai mare variaţie este între 10 şi 42MHz. Corespondenţa între gradul de modulare şi rezoluţia admise pentru fiecare tip de MC este dată în tabele.

Protecţia la subtensiune

Modulul de protecţie la scăderea tensiunii de alimentare generează un RESET dacă tensiunea de alimentare scade sub un anumit prag. Validarea acestui RESET se poate face în registrul de control PDRC. Generarea semnalului de RESET se face după 2,27μs de la scăderea tensiunii sub 4V. Tensiunea de referinţă este creată în acest modul. Partea analogică a circuitului poate fi deconectată în modurile de lucru cu economie de energie şi în acest caz consumul este sub 0,26mA. Deoarece semnalul de subtensiune de la partea analogică poate surveni în orice moment şi pentru a evita stările instabile, RESET-ul devine activ după câteva impulsuri de tact.

Registrul de comandă:

PDRCR (Power Down Reset Control Register) în care un bit validează funcţionarea modulului de supraveghere a subtensiunii, un bit validează tactul pentru partea digitală, un bit validează modul de economie de energie pentru partea analogică.

4.7.Ceasul de timp real RTC (Watchtimer)

Ceasul de timp real constă într-un registru de control, un registru pentru sub-secunde, pentru secunde, minute şi ore, un divizor al tactului cu 2, un registru de prescalare de 21 de biţi şi numărătoare de secunde, minute şi ore.

Schema bloc a ceasului de timp real este dată în figura 4.8.

O întrerupere poate fi generată de către bitul de depăşire al numărătoarelor de sub-secunde/ secunde/ minute/ ore (INT3-0) dacă fiecare întrerupere este validată (INTE3-0). Numărătoarele pot fi încărcate paralel pentru a schimba ora (Update) prin regiştrii de sub-secunde, secunde, minute, ore.

Regiştrii:

WTCR (Timer Control Register)(16 biţi) conţine flagurile de cerere de întrerupere (INT3-0) şi biţii de validare (INTE3-0). Un bit arată dacă RTC este activ, un bit comandă modul de modificare al timpului (Update). După o modificare bitul revine automat la starea inactivă. Un bit de start porneşte numărarea după o operaţie de modificare.

WTBR (Sub-second Register)(21 de biţi) stochează valoarea de reluare a numărării pentru circuitul de prescalare. Acest registru trebuie încărcat când RTC este oprit. Prin divizarea tactului se urmăreşte obţinerea unui tact cât mai apropiat de 1 s. La 4MHz factorul de divizare tipic este 1E847FH iar pentru 32KHz este 4000H.

WTSR, WTMR, WTHR (Second- Minute- Hour Registers, 6 biţi/ 6 biţi/ 5 biţi). Citirea acestor regiştrii trebuie făcută la primirea unei întreruperi pentru ca datele să fie stabile.

WTBL (Clock Disable Register), în care un bit dezactivează tactul RTC.

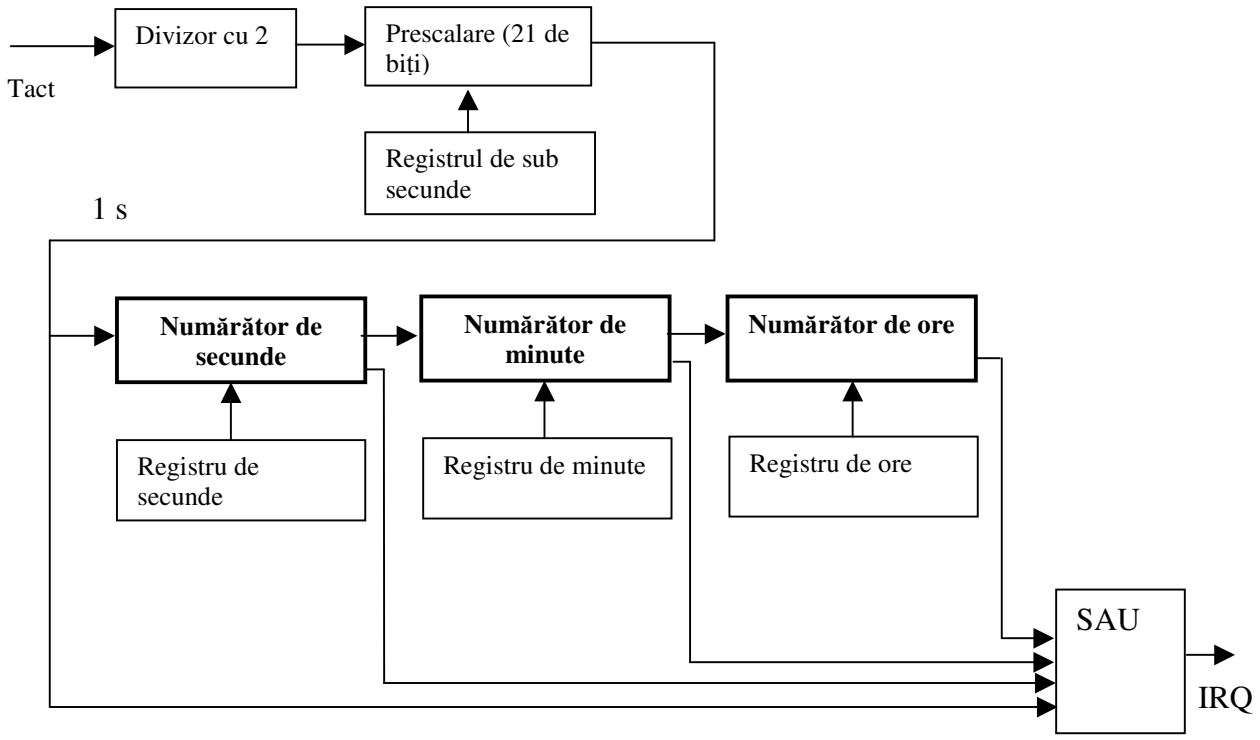


Figura 4.8: schema bloc a ceasului de timp real

O întrerupere poate fi generată de către bitul de depăşire al numărătoarelor de sub-secunde/ secunde/ minute/ ore (INT3-0) dacă fiecare întrerupere este validată (INTE3-0). Numărătoarele pot fi încărcate paralel pentru a schimba ora (Update) prin regiştrii de sub-secunde, secunde, minute, ore.

Regiştrii:

WTCR (Timer Control Register)(16 biţi) conţine flagurile de cerere de întrerupere (INT3-0) şi biţii de validare (INTE3-0). Un bit arată dacă RTC este activ, un bit comandă modul de modificare al timpului (Update). După o modificare bitul revine automat la starea inactivă. Un bit de start porneşte numărarea după o operaţie de modificare.

WTBR (Sub-second Register)(21 de biţi) stochează valoarea de reluare a numărării pentru circuitul de prescalare. Acest registru trebuie încărcat când RTC este oprit. Prin divizarea tactului se urmăreşte obţinerea unui tact cât mai apropiat de 1 s. La 4MHz factorul de divizare tipic este 1E847FH iar pentru 32KHz este 4000H.

WTSR, WTMR, WTHR (Second- Minute- Hour Registers, 6 biţi/ 6 biţi/ 5 biţi). Citirea acestor regiştrii trebuie făcută la primirea unei întreruperi pentru ca datele să fie stabile.

WTBL (Clock Disable Register), în care un bit dezactivează tactul RTC.

4.8.Programarea memoriei FLASH. Rutina Boot ROM.

Memoria FLASH poate fi citită / scrisă / ştearsă din exterior cu controlul unor pini externi (mod FLASH) sau prin program (mod CPU). Ca şi la celelalte MC se foloseşte un algoritm automat integrat (Embedded Algorithm). Dimensiunea memoriei este funcţie de model şi poate fi de 256Kocteţi, 512Kocteţi sau 768Kocteţi. Memoria este împărţită în sectoare cu dimensiuni diferite funcţie de model şi dimensiunea totală a memoriei. De exemplu pentru o memorie de 512Kocteţi sectoarele sunt:

Sector 0 64K (adresa 80000H-9FFFCH în mod CPU şi 00000H-0FFFFH în mod FLASH)
Sector 1 64K (adresa A0000H-BFFFCH în mod CPU şi 10000H-1FFFFH în mod FLASH)
Sector 2 64K, Sector 3 32K, Sector 4 8K, Sector 5 8K, Sector 6 16K, Sector 7, 8 şi 9 64K
Sector 10 32K, Sector 11 şi 12 8K, Sector 13 16K.

Accesul la scriere / ştergere

a.La accesul exterior CPU se opreşte dacă pinii de MOD sunt 111B la un RESET de tip INIT. Ştergerea / scrierea se realizează prin intermediul magistralei externe a MC, astfel MC apare în exterior ca o memorie FLASH şi se poate folosi un inscriptor paralel. Magistrala de date poate fi 8 sau 16 biţi.

b.La accesul interior memoria poate fi scrisă sau ştearsă prin program, prin invocarea algoritmului automat. Memoria FLASH este controlată de registrul FMCS:

FMCS (FLASH Control Status Register) conţine un bit care stabileşte programarea cu acces sincron (timing prestabilit) sau asincron, un bit arată starea algoritmului (scriere /ştergere terminată), un bit validează ştergerea în mod CPU iar un bit programează modul Low Power, posibil doar la tact sub 5MHz.

Accesul la citire / scriere

a.Acces exterior controlat de un programator (mod FLASH).

b.Acces interior (mod CPU), accesul este controlat de registrul FMWT:

FMWT (FLASH Wait Control Register) care controlează cu 2 biţi timpul în care impulsul de citire este în stare High şi cu 3 biţi comandă inserarea de stări de WAIT. Valorile acestui registru sunt recomandate în foile de catalog şi diferă dacă se lucrează cu tact modulat sau nemodulat.

Algoritmul automat este pornit de o secvenţă de comandă. Secvenţele posibile sunt: citire/ RESET, scriere, ştergere memorie şi ştergere sector. Aceste secvenţe pot fi executate în mod CPU (prin program) şi în mod FLASH de către inscriptorul extern.

Pentru citirea datelor nu este nevoie de pornirea algoritmului automat pentru că datele pot fi citite din zona FLASH printr-o instrucţiune de acces la memorie obişnuită. Totuşi se poate face şi o citire prin algoritmul automat trimiţând secvenţa de date F0H la o adresă din zona FLASH. Starea implicită în care este memoria FLASH după un RESET permite citirea obişnuită a datelor, deci secvenţa de citire prin algoritm automat nu este folosită în mod curent.

Pentru scrierea datelor în FLASH secvenţa de program necesită 4 cicluri în care se trimit 4 cuvinte de date la 4 adrese specificate în foile de catalog.

Ca şi la MC pe 8 şi 16 biţi memoria FLASH este comandată de un algoritm automat, de aceea este nevoie de citirea unor flaguri pentru ca CPU să fie informată de starea de operare în curs sau terminarea operării.

Rutina Boot ROM

La adresa FF000H (vector de RESET) se află un salt la o rutină numită Boot ROM care se execută la fiecare RST sau INIT. Rolul acestei rutine este să configureze circuitul şi să asigure o comunicaţie serială simplă pentru programarea memoriei FLASH. Rutina este executată în modul "*single chip*".

Rutina iniţializează semnalele CS pentru zonele externe astfel:

- CS0 pentru a selecta o zonă de acces pe 32 de biţi şi o stare de WAIT între 200000-2FFFFFH
- CS7 pentru a selecta o zonă de acces pe 16 de biţi şi o stare de WAIT între 100000-10FFFFH (pentru CAN).

După iniţializare se verifică un vector de securitate (#66) pentru a se putea proteja memoria FLASH la citiri neautorizate. Timp de 200ms se aşteaptă prin interfaţa UART0 (9600Bd, 8 biţi, fără paritate, un bit de STOP) caracterul V. MC răspunde cu caracterul F, apoi sunt posibile 4 tipuri de comenzi:

- recepţionează date şi le scrie în sectorul specificat.
- citeşte date din sectorul specificat
- iniţiază un CALL la o adesă
- citeşte suma de control pentru verificare.