Exemplu de problema (aproape) rezolvata

Sa se proiecteze o structura cu 8086 ce contine:

- memorie EPROM de 64k mapata in spatiul 1M-64k →1M (cipuri 8K * 8)
- sistem de intreruperi
- timer cu ceas de 5MHz ; generaza intreruperi la fiecare 20ms, avind adresa de baza 0x20h.

Sistemul este prevazut cu un circuit de urmarire a acceselor de scriere la portul 0x70h. Toate datelel scrise la acest port sint stocate intr-o memorie SRAM de 2k * 8 (un singur cip), mapata in spatiul 512k→514k. Cind memoria este umpluta se genereaza o intrerupere pe IRQ3. Memoria SRAM este accesata de procesor numai pe 8 biti.
Se va folosi un numarator binar (pe 11biti) pentru contorizarea acceselor la portul 0x70h.

Se cer:

1) Schema bloc, cu evidentierea modului de functionare a procesorului (mod maxim/minim).
2) Detalierea memorie EPROM (schema , mapare).
3) Detalierea timerului (schema , initializarea modului de lucru, setarea divizorului)
4) Schema bloc a circuitului de urmarire a acceselor la portul 0x70h

REZOLVARE

1) In alcatuirea schemei intra urmatoarele blocuri:

Procesorul 8086 – trebuie stabilit de la inceput modul de lucru al procesorului. In cazul de fata, diferentele dintre cele doua moduri de lucru sunt importante prin prisma semnalelor pe care le ofera CPU. De asemenea, daca procesorul este configurat in mod de lucru **maxim**, in schema intervine un bloc in plus, si anume **controllerul de magistrala (bus controller).**

Memoria EPROM – in schema bloc nu se cer detalii de implementare pentru aceasta memorie, ci doar modul in care ea este conectata la magistralele sistemului.

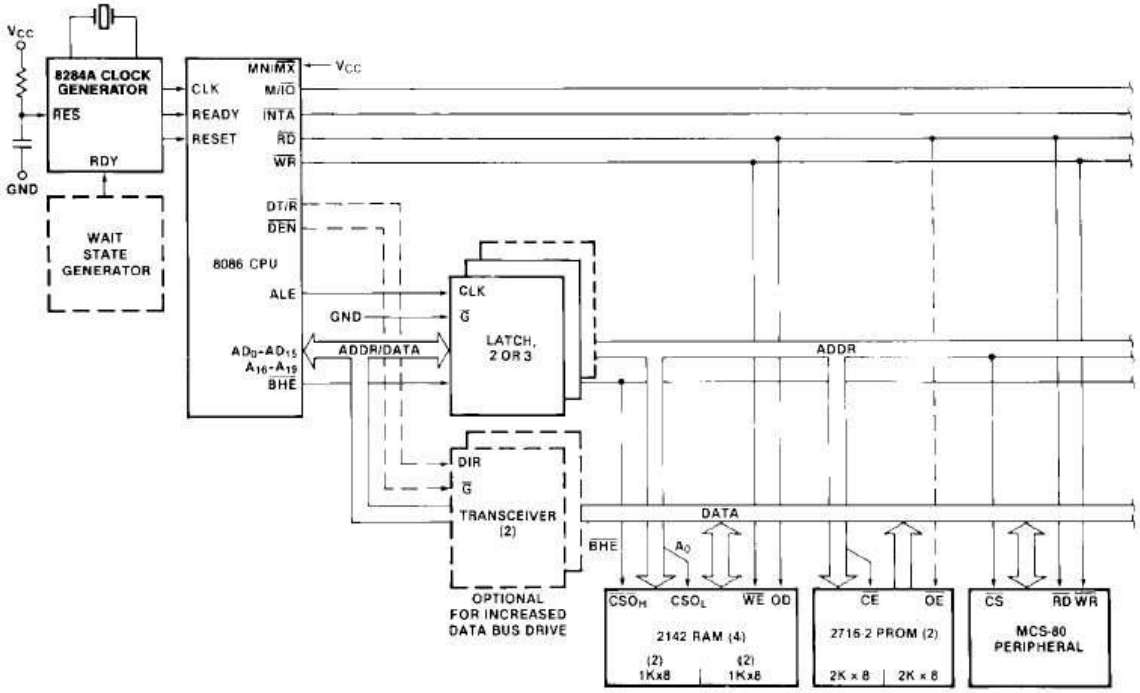
Sistemul de intreruperi – acesta este implementat cu ajutorul circuitului 8259, care trebuie conectat la magistrale.

Timer-ul – pentru implementarea acestuia se foloseste un circuit 8254.

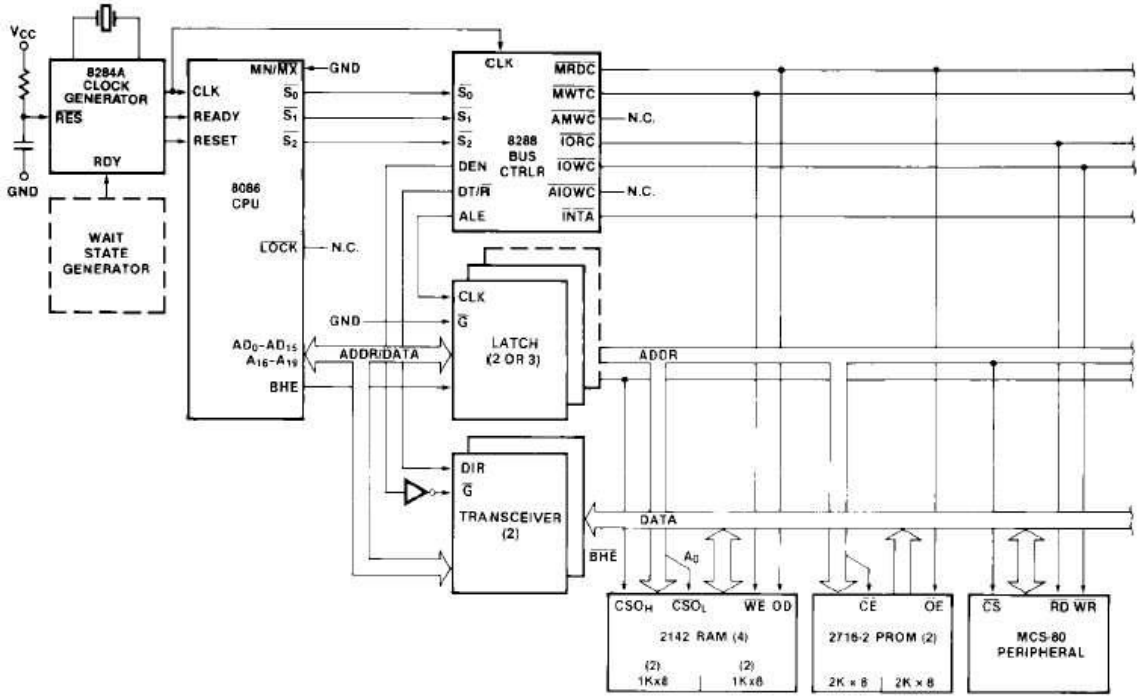
Circuit de urmarire accese la port – pentru conectarea blocului in sistem trebuie identificate intrarile si iesirile acestuia (blocul include si memoria SRAM de inregistrare a acceselor). Astfel, conexiunile vor fi cu :

- magistrala de adrese – blocul **primeste** adrese (conexiune read-only)
- magistrala de date – blocul **citeste/scrie** date (conexiune bidirectionala)
- semnalele de citire/scriere in memorie/spatiul IO (atentie! acestea difera functie de modul de lucru ales al procesorului).
- sistemul de intreruperi – blocul genereaza semnal de intrerupere

Pentru conectarea tuturor acestor blocuri, se poate porni de la schemele din curs sau de la cele oferite de catalogul Intel. Acestea din urma sunt prezentate in continuare, in ambele variante (mod minim si mod maxim).



Schema unui sistem care foloseste procesorul 8086 in modul de lucru **minim**.



Schema unui sistem care foloseste procesorul 8086 in modul de lucru **maxim**.

Tot ceea ce ramane de facut este particularizarea unui astfel de desen pentru cazul problemei cerute. Pana la urma, tine de inspiratie si imaginatie :-) .

2) Detalierea memoriei EPROM (schema, mapare)

Se cere proiectarea si maparea unei memorii EPROM in ultimii 64Ko ai spatiului de adresare al procesorului, folosindu-se cipuri de 8K x 8.

Care sunt pasii necesari pentru aceasta operatie?

a. *Identificarea numarului de circuite necesare pentru implementarea memoriei.*
   Pentu cazul de fata avem nevoie de 8 cipuri de felul celor specificate in problema.

b. *Stabilirea modului de adresare pentru cipuri.*
   Trebuie pornit de la semnificatia datelor pentru cip.
   Retineti : primul numar (in cazul de fata 16K) implica numarul de pini de intrare (adresa), iar cel de-al doilea reprezinta numarul de biti/semnale de iesire (date).
   O memorie de 16k x 8 este o memorie cu 16 * 1024 cuvinte de date de cate 8 biti. Aceasta inseamna ca la fiecare adresa se afla un cuvant de 8 biti (sau, altfel spus, doua adrese consecutive difera printr-un cuvant de date, nu printr-un bit).
   16k adrese => $2^4 * 2^{10}$ adrese => 10 + 4=14 biti de adresa.
   Asadar, din adresa de 20 de biti pe care o furnizeaza procesorul, 14 vor fi folositi pentru a identifica un cuvant in cadrul unui cip EPROM.

   Cateva exemple de celule:

Tip cip	**Numar biti/pini adresa**	**Pentru accesare date**			**Capacitate in octeti**
		pe 8 biti	**pe 16 biti**	**pe 32 biti**	
8K x 8	13 ($2^3 * 2^{10}$ adrese)	1 celula	2 celule in paralel	4 celule in paralel	8K
16K x 8	14 ($2^4 * 2^{10}$ adrese)	1 celula	2 celule in paralel	4 celule in paralel	16K
32K x 8	15 ($2^5 * 2^{10}$ adrese)	1 celula	2 celule in paralel	4 celule in paralel	32K
8K x 16	13 ($2^3 * 2^{10}$ adrese)	-	1 celula	2 celule in paralel	16K

   Atentie! Verificati enuntul problemei pentru a vedea in ce fel trebuie accesate datele din memoria EPROM. In cazul structurii cu 8086, datele trebuie sa fie accesibile si pe 8 si pe 16 biti. Automat, celulele vor fi grupate in perechi.

   Fiecare cip are doua semnale importante:
   - CS (semnal negat) – chip select – numai daca pe pinul respectiv se trimite valoarea 0, circuitul este activ.
   - OE (semnal negat) – output enable – daca pe acest pin se trimte 0, circuitul trimite datele la iesire.

   Avem deci 4 perechi de celule de memorie. La un moment dat, numai un cip sau cel mult o pereche de cipuri poate fi activa (pentru citire pe 8, respectiv 16 biti).

c. Adresarea celulelor

Pentru a citi din memoria ROM, procesorul trimte o adresa de 20 de biti. Datele cerute se afla numai in unul dintre cele 8 cipuri (in cazul accesului pe 8 biti) sau in una dintre perechi (in cazul accesului pe 16 biti). Se pune problema decodificarii respectivei adrese astfel incat sa fie accesata corect locatia de memorie. Retineti ca procesorul NU stie in ce fel este organizata memoria (in cate cipuri si de ce fel sunt acestea) si nici nu este influentat in nici un fel de aceasta asezare.

Care sunt pasii de decodificare ?

c1. EPROMSEL
Se defineste un semnal, EPROMSEL, care este „responsabil" de selectarea spatiului memoriei EPROM. Definirea acestui semnal este echivalenta cu maparea memoriei in spatiul precizat.

In cazul acestei probleme, memoria trebuie mapata in ultimii 64KB ai spatiului de adrese, care are 1MB.

Ultima adresa din EPROM :
1111 1111 1111 1111 1111.
Prima adresa din EPROM :
1111 0000 0000 0000 0000
(64K = $2^6 * 2^{10}$ => 6+10 = 16 biti 0)

Se observa ca toate adresele din EPROM incep cu 4 biti 1. Altfel spus, memoria EPROM este selectata de orice adresa in care bitii $A_{20}A_{19}A_{18}A_{17}$ sunt 1.
Deci, EPROMSEL = $A_{19} \wedge A_{18} \wedge A_{17} \wedge A_{16}$

c2. CS0 – CS3
Aceste semnale sunt semnalele „chip select" pentru fiecare linie (pereche) in parte. Pentru ca se cere ca ambele cipuri de pe o linie sa poata fi accesate simultan, semnalul CS va fi comun.

Pentru alegerea uneia dintre cele 4 linii se vor folosi bitii A_{15}-A_{14}.

A15	**A14**	**Semnal**
0	0	CS0 – selectare linia 0
0	1	CS1 – selectare linia 1
1	0	CS2 – selectare linia 2
1	1	CS3 – selectare linia 3

Pentru obtinerea acestor semnale in acest fel, solutia cea mai simpla este folosirea unui decodificator. Acesta va fi activat (prin semnalul ENABLE de care dispune) chiar de EPROMSEL, astfel incat o linie este activata numai daca adresa este in spatiul EPROM-ului.

c3. Adresa in cip.

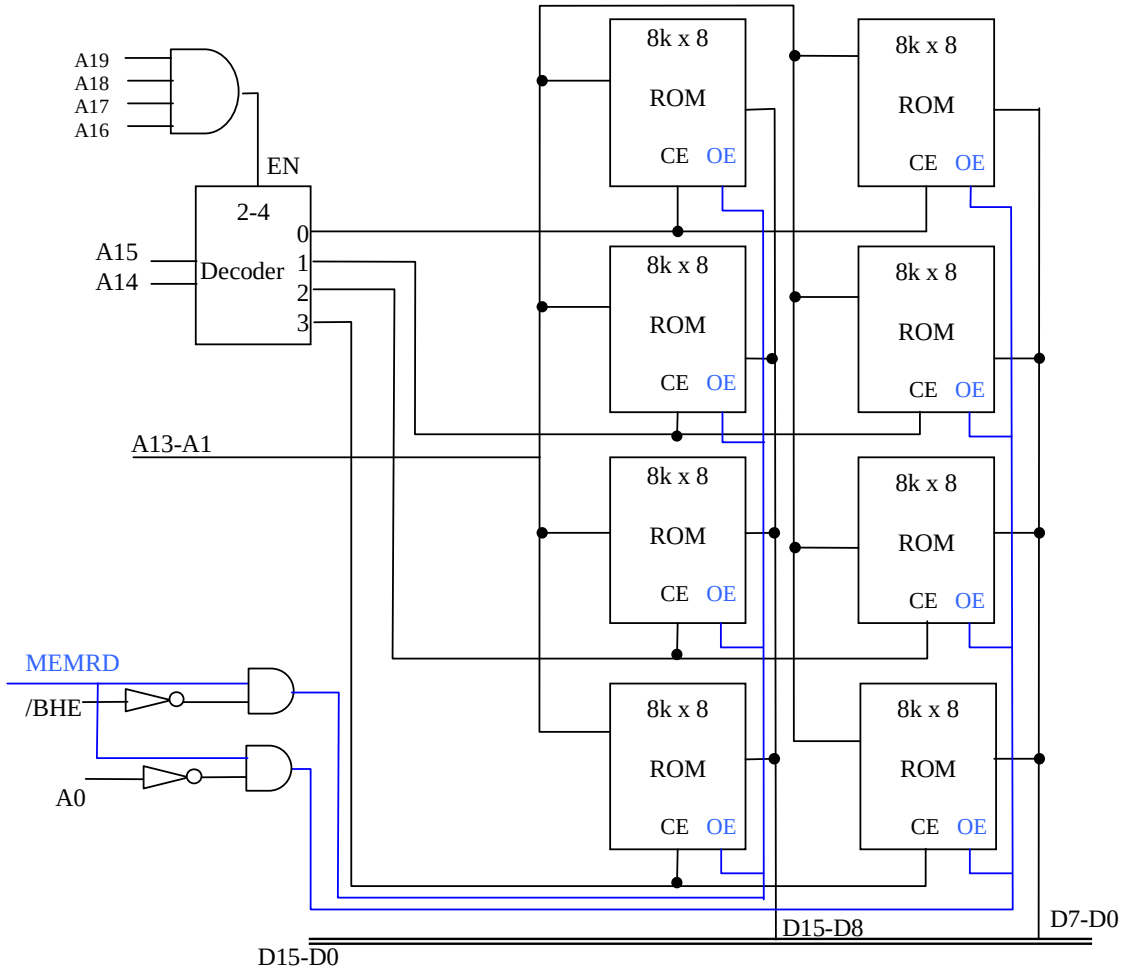
In cadrul fiecarui cip, adresarea se face pe 13 biti. Acestia sunt bitii A_{13}-A_1, care sunt furnizati ca intrari de adresa pentru fiecare cip in parte. Citirea are loc numai pentru cipurile in care CS este activ.

c4. Date disponibile pe 8/16 biti

Datele din EPROM sunt organizate in octeti: adresele pare intr-o coloana (cea din dreapta), cele impare in cealalta coloana (cea din stanga).
Daca se doreste citirea datelor pe 8 biti, in linia selectata cu CS se va alege numai una dintre celule. Pentru a determina ca una singura dintre celule sa ofere date se foloseste semnalul OE, care va fi activ pentru celula adresata, si inactiv pentru perechea acesteia. Aceasta alegere se face in functie de bitii A0 si BHE. Datele citite sunt puse pe magistrala de date pe liniile corespunzatoare: fie D_0-D_7 pentru adresele pare, fie D_8-D_{15} pentru adresele impare.
Modul de lucru al semnalelor respective este prezentat tabelul urmator. Retineti ca aceasta varianta este o restrictie obligatorie pentru procesorul 8086. (O arhitectura „proprie” poate sa foloseasca doar bitul A0 pentru selectarea octetului de citit).

/BHE	A0	Date
0	0	Date citite pe 16 biti, de la adrese pare, pe D_0-D_{15}
0	1	-
1	0	Date citite pe 8 biti, de pe D_0-D_7
1	1	Date citite pe 8 biti., de pe D_8-D_{15}

In concluzie, schema memoriei EPROM este prezentata in continuare:



Semnalele care sunt transmise pe pinii OE ai cipurilor de memorie se numesc EPROMLBE si EPROMHBE (EPROM Low Byte Enable , respectiv EPROM High Byte Enable).

$EPROMLBE = MEMRD \wedge \overline{A0}$

$EPROMHBE = MEMRD \wedge BHE$

O alta varianta pentru rezolvare – in cazul in care nu se folosesc semnalele OE – este introducerea unor buffere 3-state pentru conectarea celor doi octeti la magistrala. In acest fel, semnalele EPROMLBE si EPROMHBE sunt folosite ca semnale de enable pentru cele 2 buffere. Expresiile lor trebuie putin modificate pentru aceasta varianta:

$EPROMLBE = EPROMSEL \wedge \overline{A0}$

$EPROMHBE = EPROMSEL \wedge BHE$

In concluzie, rezolvarea acestui punct presupune obtinerea unei scheme echivalenta cu cea de mai sus si evidentierea modului in care sunt folositi bitii de adresa pentru a obtine rezultate corecte.
Atentie! O parte dintre semnale pot fi obtinute in moduri diferite in functie de modul de lucru al procesorului.

3) Detalierea timer-ului.

Dupa cum se stie, pentru a implementa un astfel de timer se va folosi circuitul 8253.
Rezolvarea are 2 pasi:

- se introduce circuitul in schema, in functie de adresa la care este el mapat
- se configureaza circuitul astfel incat sa se obtina intreruperile la intervalele dorite

Pentru prima parte, rezolvarea este asemanatoare cu aceea din punctul anterior. Mai exact, timer-ul are un semnal CS (semnal negat) care trebuie sa fie activat (deci egal cu 0) in cazul in care adresa de pe magistrala de adrese este 0x20.

Solutia cea mai simpla ar fi maparea „directa”, in sensul ca se poate scrie functia logica prin intermediul careia sa se obtina TIMERCS = 0 in cazul in care bitii de pe magistrala de adresa au configuratia : 0000 0000 0000 0010 0000. (in ordinea A19-A0).
O astfel de logica functioneaza, dar este foarte costisitoare si deloc scalabila. In acest caz, se foloseste de fapt un decodificator pentru spatiul de adrese I/O care sa permita conectarea mai multor astfel de periferice. Avand in vedere ca in general adresele pentru periferice se pun la porturile de forma 0xN0, decodificatorul primeste la intrare bitii A8,A7,A6,A5 si returneaza semnale de tip CS pentru 16 dispozitive, aflate la adresele intre 0x00 si 0xF0. Pentru ca maparea sa fie corecta, decodificatorul primeste ca semnal de ENABLE o functie de genul :

$EN = (IORD + IOWR) \cdot \overline{A_{20}} \cdot \overline{A_{19}} \cdot \ldots \cdot \overline{A_9}$

Se pot genera multe alte variante. Important este ca semnalele CS sa asigure ca un singur dispozitiv I/O este activ la un moment dat.

Pentru conectarea la magistrale, mai trebuie facute doar cateva precizari:
Timer.RD = IORD
Timer.WR = IOWR
Timer.CS = TIMERCS (calculat/decodificat anterior)
Timer.A1 = Address.A1
Timer.A0 = Address.A0

Partea a doua este cea care tine de functionalitatea timerului.
Pentru a putea obtine dintr-o freventa de 5MHz un tact cu frecventa de 20ms, trebuie o divizare cu 100000.
De ce?
Daca timerul numara pe frecventa proprie, el furnizeaza intreruperi la :
$1/ (5 * 10^6) = 2 * 10^{-7}$ secunde.
Se cere o marire a intervalului la $2*10^{-2}$ secunde.
Rezulta deci ca este nevoie de divizarea cu 10^5, adica 100000.

Se va folosi modul 0 („interrupt on terminal count”), ceea ce inseamna ca numara de la valoarea setata pana la 0, si apoi genereaza o intrerupere.
Datorita faptului ca numaratoarele din timer sunt de 16 biti, numarul maxim care poate fi setat in cadrul unui numarator este 2^{16} = 64K < 100000. Vor trebui deci folosite doua dintre numaratoarele lui 8253. Ambele vor fi folosite in modul 0. Primul va numara de la 50000 dupa ceasul propriu (urmand sa dea cate o intrerupere la fiecare 10ms). Cel de-al doilea va primi pe intrarea de ceas iesirea primului, si va numara de la 2, ceea ce inseamna ca va furniza intrerupere la 20ms.
Lista conexiunilor este urmatoarea:

Timer.CLK0 = CLK_5MHz
Timer.CLK1 = Timer.Out0

Secventa de initializare trebuie sa tina cont de :

- adresa portului de control (A1=1, A0=1 => 0x23)
- ambele numaratoare (counter0 si counter1) trebuie programate, fiecare la adresa lui (0x20 si 0x21).

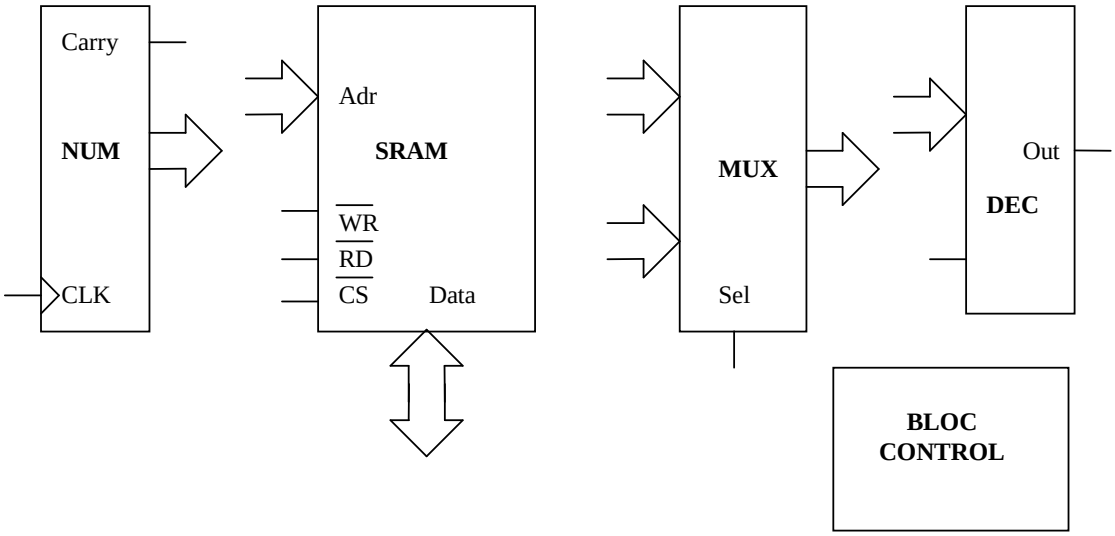
Rezolvarea trebuie sa contina si schema 8253 (mai ales pentru a se evidentia modul in care sunt conectate semnalele) si „programul” de intializare (cuvintele de intializare si porturile la care acestea trebuie scrise).

4) Circuitul de supraveghere a acceselor la portul 0x70

Schema bloc a acestui circuit se bazeaza pe urmatoarele blocuri:

- Decodificator adresa port (DEC). Acesta primeste ca intrare adresa de pe magistrala de adrese supravegheata si activeaza iesirea daca a primit adresa corespunzatoare perifericului care trebuie supravegheat

- Un numarator pe 11 biti (NUM), care va genera urmatoarea adresa din memoria SRAM in care se scriu datele. Are ca intrare semnalul de ceas (si eventual un semnal de reset, pentru a fi adus la 0).
- Blocul de memorie (SRAM). Acesta este accesibil pentru scriere (semnalul WR este activat de o scriere la portul 0x70) sau pentru citire (semnalul RD este activat chiar de procesor).Semnalul CS se obtine din adresa la care este mapat blocul de memorie (respectiv 512K).
- Un multiplexor (MUX) care va selecta – in functie de actiunea asupra memoriei – adresa care trebuie furnizata. Astfel, daca are loc o operatie de citire, va fi data adresa de la procesor (mai exact ultimii 11 biti), iar daca se incearca o scriere adresa este funrnizata de numarator.
- Un bloc de control, care asigura (eventual) interfata cu semnalele pe care le furnizeaza procesorul si cu sistemul de intreruperi.



Lista conexiunilor este prezentata (pe scurt) in continuare.

Decodificarea accesului la portul 0x70h se face :

DEC.IN[0,7] = Adr[0,7] # Adresa portului
DEC.En = IOW # Acces la periferice LA

Umplerea memoriei SRAM este echivalenta cu terminarea numarari:

IRQ3 = **NUM**.Carry

Fiecare acces la portul 0x70h duce la incrementarea numartorului

NUM.Clk = **DEC**.Out

Adresele generate de numarator sint multiplexate cu adresele generate de procesor

MUX.In1[0,10] = **NUM**.Out[0,10]
MUX.In2[0,10] = Adr[0,10] # Accesul se face pe 8 biti

Selectia pentru multiplexor este data tot iesirea decodificatorului

MUX.Sel = **DEC**.out # Daca este activ selecteaza iesirea numaratorului

Memoria este conectata pe magistrala de date a sistemului

SRAM.Data[0,7] = Data[0,7]

Daca procesorul vrea sa scrie la port (iesirea decodificatorului este activa) sau vrea sa citeasca din memorie atunci inseamna ca trebuie activat SRAM-ul:

SRAM.CS = **DEC**.Out + SelectSRAMCitire
unde:
SelectSRAMCitire se obitine cind Adr[0-19] indica spatiul SRAM-ului iar MEMR este activ (0)
OBS: trebuie luata in considerare intirzierea data de numarator

Daca se acceseaza portul atunci facem o scriere in SRAM

SRAM.WR = **DEC**.Out

Dace procesorul vrea sa acceseze SRAM-ul atunci o citire:

SRAM.RD = **PROCESOR**.MEMRD

Rezolvarea trebuie sa includa schema completa a circuitului respectiv.

GATA!!!

Precizare:
Autorul rezolvarii este Ana Varbanescu. De aceea, pentru sugestii, intreabri si reclamatii trimiteti mesaje la adresa de mail cunoscuta.