

MICROSISTEM CU MICROPROCESORUL 8086

STUDENT: TÎRSÎNA NICOLETA

An universitar: 2025-2026

TEMA PROIECTULUI

Să se proiecteze un microsistem cu următoarea structură:

- unitate centrală cu microprocesorul 8086;
- 128 KB memorie EPROM, utilizând circuite 27C2048;
- 64 KB memorie SRAM, utilizând circuite 62512;
- interfață serială, cu circuitul 8251, plasată în zona 0AF0H – 0AF2H sau 0BF0H – 0BF2H, în funcție de poziția microcomutatorului S1;
- interfață paralelă, cu circuitul 8255, plasată în zona 0D70H – 0D76H sau 0C70H – 0C76H, în funcție de poziția microcomutatorului S2;
- minitastatură cu 12 contacte;
- 6 led-uri;
- un modul de afișare cu 7 segmente, cu 6 ranguri;

Toate programele în limbaj de asamblare vor fi concepute sub formă de subrutine. Programele necesare sunt:

- rutinele de programare ale circuitelor 8251 și 8255;
- rutinele de emisie/ recepție caracter pe interfața serială;
- rutina de emisie caracter pe interfață paralelă;
- rutina de scanare a minitastaturii;
- rutina de aprindere/ stingere a unui led;
- rutina de afișare a unui caracter hexa pe un rang cu segmente.

Structura rutinelor (intrări, secvențe, ieșiri) va fi stabilită de fiecare student.

UNITATEA CENTRALĂ CU PROCESOR 8086

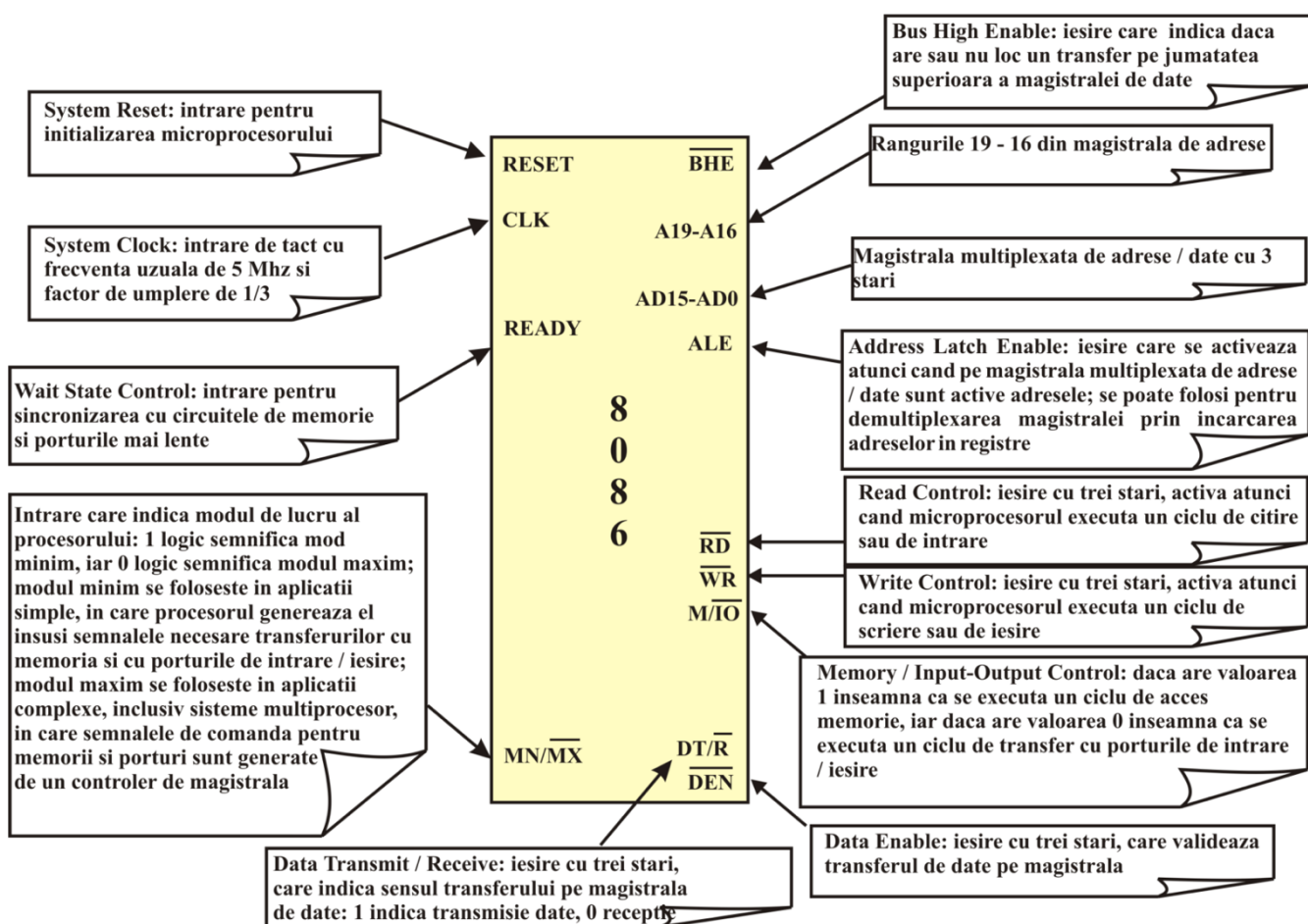
Unitatea centrală a microsistemului este compusă din:

- microprocesorul **Intel 8086**
- generatorul de tact **8284A**
- circuitele pentru amplificarea și separarea magistrelor de adrese și date (circuitul amplificator/separator bidirecțional **74LS245**, circuitul registru **74LS373**)

Microprocesorul Intel 8086

Microprocesorul 8086 este unitatea centrală de calcul a microsistemului, având o arhitectură pe 16 biți și o magistrală de adrese pe 20 de biți, ceea ce permite accesarea a 1 MB de memorie. Acesta generează semnalele necesare pentru accesul la memorie și I/O și comunică prin magistrale multiplexate care trebuie separate extern. 8086 coordonează întregul sistem, executând instrucțiunile și gestionând ciclurile de citire/scriere.

Principalele Terminale ale 8086:



Generatorul de tact 8284A

8284A generează semnalul de ceas necesar funcționării procesorului 8086, asigurând forma corectă și stabilitatea semnalului CLK. Circuitul mai produce semnalele PCLK (pentru periferice), READY (pentru dispozitive lente) și impulsul de RESET. În proiect se folosește **un singur circuit 8284A**, acesta fiind responsabil pentru sincronizarea întregului microsistem. Fără el, procesorul nu ar putea executa instrucțiuni în ritm corect.

Latch-uri 74LS373

Circuitele 74LS373 sunt utilizate pentru **demultiplexarea magistralei AD0–AD15** a procesorului 8086. În proiect sunt folosite **trei astfel de latch-uri**: două pentru memorarea adresei inferioare A0–A15, iar unul pentru A16–A19 și semnalul BHE. La impulsul ALE, acestea rețin partea de adresă emisă de procesor, asigurând o adresare stabilă către memoria EPROM și SRAM. Practic, ele separă adresele de datele multiplexate ale procesorului.

Funcționarea

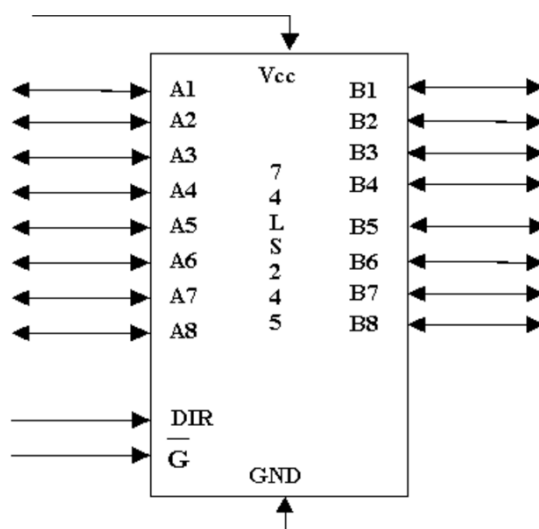
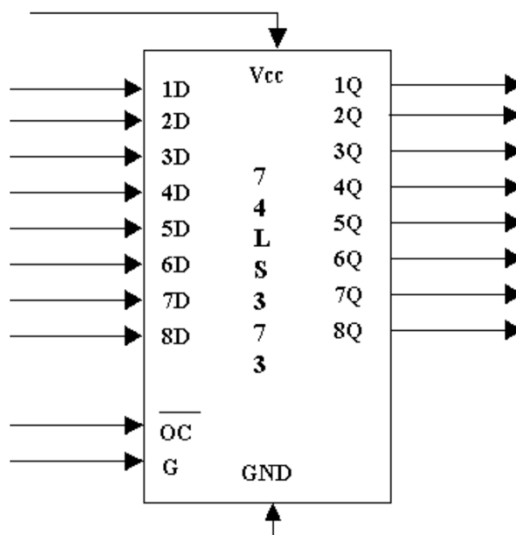
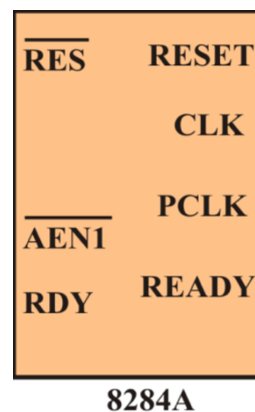
/OC	G	8Q – 1Q
0	0	Vechiul conținut
0	1	8D – 1D
1	X	A 3 – a stare

Buffere bidirecționale 74LS245

Circuitele **74LS245** sunt utilizate pentru izolarea și amplificarea magistralei de date pe 16 biți. Sunt folosite **două astfel de circuite**, unul pentru octetul inferior (D0–D7) și unul pentru octetul superior (D8–D15). Direcția transferului de date este controlată de semnalul DT/ R al procesorului, iar activarea bufferelor se face prin DEN. Aceste circuite previn conflictele pe magistrala de date și permit comunicarea sigură între procesor și memorie

Funcționarea.

/G	DIR	A8 – A1	B8 – B1
0	0	B8 – B1	Intrări
0	1	Intrări	A8 – A1
1	X	A 3 – a stare	A 3 – a stare



MEMORIA MICROSISTEMULUI

Memoria microsystemului este alcătuită din:

- **128 KB EPROM**, realizată cu două circuite **27C2048 (128K × 16)**
- **64 KB SRAM**, realizată cu un circuit **62512 (64K × 8) × 2** pentru date pe 16 biți
- Logica de selecție a memoriei, realizată cu un decodificator **74LS138**
- Liniile de adresă A0–A19 provenite din unitatea centrală

Circuitul de memorii

Circuitul memoriei microsystemului este proiectat pentru a permite accesul procesorului 8086 la două tipuri de memorii: SRAM pentru date volatile și EPROM pentru programul utilizat la pornirea sistemului. Ambele tipuri de memorii sunt conectate la magistrala de adrese A0–A19 și la magistrala de date bidirecțională pe 16 biți, prin buffer-ele 74LS245 și latch-urile 74LS373 din unitatea centrală.

Pentru a asigura accesul corect la fiecare zonă de memorie, circuitul include o logică de decodificare realizată cu un 74LS138, care generează semnalele de selectare (CS) pentru SRAM și EPROM în funcție de liniile superioare ale adresei. Astfel, în funcție de intervalul de adresă emis de procesor, este activat automat fie circuitul SRAM, fie circuitul EPROM

Circuitul EPROM 27C2048 (128K × 16 biți)

În proiect este utilizat **un singur circuit EPROM 27C2048**, deoarece acesta oferă direct o lățime de 16 biți (128 KB × 16), compatibilă cu magistrala de date a procesorului 8086. Această memorie este nevolatilă și păstrează conținutul chiar și în absența alimentării. Programarea memoriei se face aplicând tensiuni ridicate pe pinul VPP, iar ștergerea se poate face doar cu lumină ultravioletă.

Caracteristicile circuitului:

- **17 intrări de adresă A0–A16** → permit accesarea celor **128K locații** → conectate direct la liniile de adresă ale procesorului (adrează cele 128K locații).
- **16 linii de date Q0–Q15** → asigură transferul unui cuvânt de 16 biți la fiecare acces → conectate la magistrala de date D0–D15.
- **!CE – Chip Enable** (activ LOW) – selecția circuitului → activat de semnalul Y7 al decodificatorului 74LS138.
- **!OE – Output Enable** (activ LOW) – activarea ieșirilor în citire → conectat la semnalul **RD** al procesorului pentru operațiile de citire.

Circuitul SRAM 62512 (64K × 8)

În proiect sunt utilizate **două circuite SRAM 62512**, fiecare având capacitatea de **64 KB × 8 biți**. Prin utilizarea a două circuite identice (LOW și HIGH), se obține o magistrală de date de **16 biți**, compatibilă cu arhitectura procesorului 8086. SRAM-ul este volatil, dar are timpi foarte buni de acces și nu necesită reîmprospătare, fiind ideal pentru stivă, variabile și zone temporare de lucru.

Caracteristicile circuitului:

- **16 intrări de adresă A0–A15** → conectate la liniile de adresă ale procesorului pentru selectarea locațiilor interne.
- **8 linii de date I/O0–I/O7** → conectate la D0–D7 (SRAM_LOW) și D8–D15 (SRAM_HIGH).
- **!CS – Chip Select** (activ LOW) → generat din Y0 al decodicatorului 74LS138.
- **!OE – Output Enable** pentru operații de citire → conectat la semnalul **RD**.
- **!WE – Write Enable** pentru operații de scriere → conectat direct la WR al procesorului și permite scrierea doar în cipul selectat prin logica bazată pe A0 și BHE. **VCC și GND** → alimentarea circuitelor.

Decodificarea memoriei

Pentru selectarea corectă a memoriilor se folosește un decodicator **74LS138**, care primește pe intrările A, B și C semnalele: **A17 (LSB), A18, A19 (MSB)**. Acesta generează 8 ieșiri active LOW: Y0–Y7.

SRAM

$$64 \text{ KB} = 2^6 * 2^{10} \text{ B} = 2^{16} \text{ B}$$

Adresa de început: 00000H

Adresa de Final: 0FFFFH

	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Adresa de început	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Adresa de final	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Ecuția de selecție:

$$!SEL1 = A19 + A18 + A17 + A16 = !Y0$$

Eprom:

$$128 \text{ KB} = 2^7 * 2^{10} \text{ B} = 2^{17} \text{ B}$$

Adresa de început: E0000H

Adresa de Final: FFFFFH

	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Adresa de început	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Adresa de final	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Ecuția de selecție:

$$!SEL2 = !A19 + !A18 + !A17 = !Y7$$

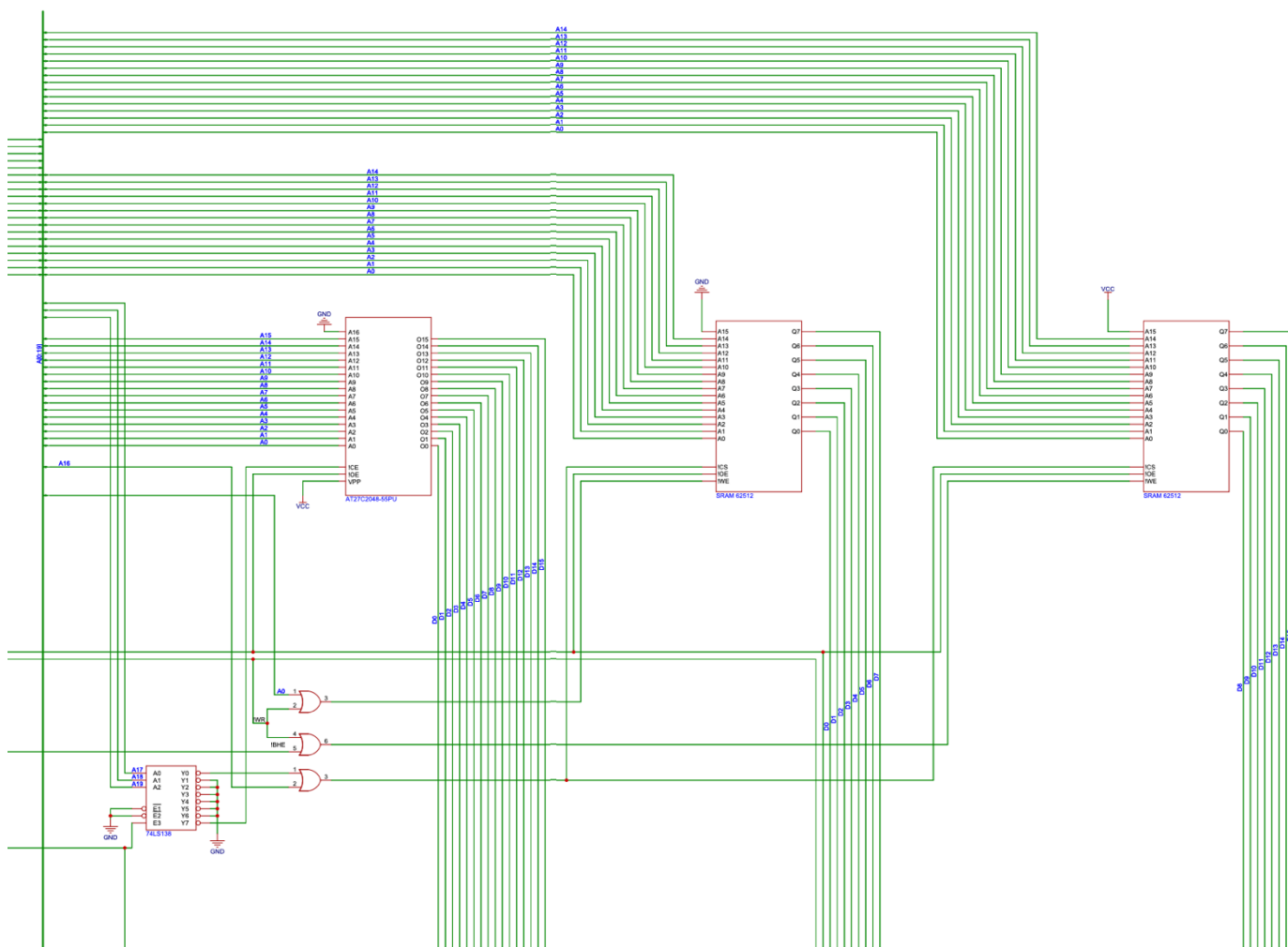


Figura 2. Memoriile in EasyEda

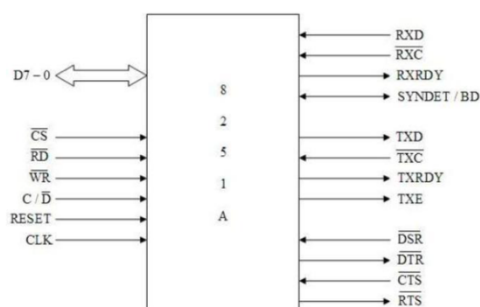
INTERFATĂ SERIALĂ ȘI INTERFATĂ PARALELĂ

Interfața serială

Interfața serială reprezintă ansamblul de circuite și rutine software care permit comunicarea bit cu bit între unitatea centrală și dispozitivele externe. Acest tip de comunicare este utilizat în special la distanțe mai mari de câțiva metri, datorită costului redus al cablării și a robusteții crescute la interferențe electromagnetice. Avantajul major îl constituie necesarul minim de conductoare pentru conectare, ceea ce reduce investiția în cabluri și conectică.

În cadrul microsistemului proiectat, circuitul specializat **8251** îndeplinește rolul principal în transferul serial de date. Acesta primește date paralele de la CPU prin intermediul magistralei de date și le transmite serial, conform parametrilor de funcționare configurați prin registrele de control. În mod similar, datele primite serial din exterior sunt convertite în paralel și puse la dispoziția CPU-ului.

Circuitul 8251 poate funcționa atât în mod asincron, cât și în mod sincron, permițând o flexibilitate ridicată în adaptarea la diverse tipuri de protocoale de comunicare. Configurarea funcționării acestuia (viteza de baud, numărul de biți de date, paritatea, stopul etc.) se realizează prin intermediul secvențelor de inițializare definite de programul de control.

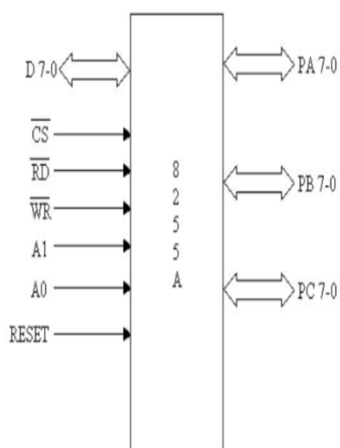


/CS	/RD	/WR	C/D	OPERAȚIE
1	X	X	X	Magistrala de date în a 3-a stare
0	1	1	X	Magistrala de date în a 3-a stare
0	0	1	1	Citire a octetului de stare
0	0	1	0	Citire a datei
0	1	0	1	Scriere a cuvintelor de comandă
0	1	0	0	Scriere a datei

Interfața paralelă

Interfața paralelă permite transferul simultan a 8 biți de date între unitatea centrală și periferice, oferind astfel o viteză superioară în comparație cu transferul serial. Spre deosebire de transmisia bit-cu-bit, specifică comunicării seriale, aici întregul octet este transferat într-un singur ciclu, fiind însoțit de semnale de control care stabilesc direcția transferului și sincronizarea acestuia.

În cadrul microsistemului proiectat, comunicarea paralelă este realizată prin intermediul magistralei de date **D0–D7**, prin care se pot transmite atât informații brute (date), cât și cuvinte de control sau stare necesare funcționării periferiilor.



/CS	/RD	/WR	A1	A0	OPERAȚIE
0	1	0	0	0	Scriere în portul A
0	1	0	0	1	Scriere în portul B
0	1	0	1	0	Scriere în portul C
0	1	0	1	1	Scriere în portul cuvântului de comandă
0	0	1	0	0	Citire în portul A
0	0	1	0	1	Citire în portul B
0	0	1	1	0	Citire în portul C
0	0	1	1	1	Fără operație – magistrala de date este în a 3-a stare
0	1	1	X	X	Fără operație – magistrala de date este în a 3-a stare
1	X	X	X	X	Fără operație – magistrala de date este în a 3-a stare

Conectare porturi

Interfață serială varianta 1: - port comenzi: **0AF0H**
- port date: **0AF2H**

Interfață serială varianta 2: - port comenzi: **0BF0H**
- port date: **0BF2H**

Interfață paralelă varianta 1: -port A: **0D70H**
-port B: **0D72H**
-port C: **0D74H**
-port RCC: **0D76H**

Interfață paralelă varianta 2: -port A: **0C70H**
-port B: **0C72H**
-port C: **0C74H**
-port RCC: **0C76H**

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	INTRARE
0AF0H	0	0	0	0	1	0	1	0	1	1	1	1	0	0	0	0	S:1
0AF2H	0	0	0	0	1	0	1	0	1	1	1	1	0	0	1	0	
0BF0H	0	0	0	0	1	0	1	1	1	1	1	1	0	0	0	0	S:2
0BF2H	0	0	0	0	1	0	1	1	1	1	1	1	0	0	1	0	
0D70H	0	0	0	0	1	1	0	1	0	1	1	1	0	0	0	0	P:1
0D72H	0	0	0	0	1	1	0	1	0	1	1	1	0	0	1	0	
0D74H	0	0	0	0	1	1	0	1	0	1	1	1	0	1	0	0	
0D76H	0	0	0	0	1	1	0	1	0	1	1	1	0	1	1	0	
0C70H	0	0	0	0	1	1	0	0	0	1	1	1	0	0	0	0	P:2
0C72H	0	0	0	0	1	1	0	0	0	1	1	1	0	0	1	0	
0C74H	0	0	0	0	1	1	0	0	0	1	1	1	0	1	0	0	
0C76H	0	0	0	0	1	1	0	0	0	1	1	1	0	1	1	0	

În microsistemul proiectat, interfețele seriale (8251) și paralele (8255) sunt conectate la magistrala sistemului prin intermediul unui decodificator de adrese **74x138**, utilizat pentru generarea semnalelor de selecție (Chip Select).

Pentru domeniul de adrese în care se află interfețele seriale și paralele, intrările de autorizare sunt definite prin următoarele relații:

$$!E1 = A15 + A14 + A13 + A12 + A3$$

$$!E2 = !A11 + !A6 + !A5 + !A4$$

$$E3 = \sim (M/\sim IO)$$

Aceste ecuații asigură faptul că **decodificatorul este activ numai pentru adresele care aparțin spațiului destinat interfețelor I/O**, excluzând accesul din zona de memorie și garantând funcționarea corectă a sistemului.

Circuitul 8251 dispune de două adrese:

- port comenzi
- port date

Fiecare variantă a interfeței seriale (Varianta 1 sau Varianta 2) este selectată prin poziția comutatorului **S1**, care influențează biții A9, A8 și A7 din adresa furnizată decodului.

Semnalele de selecție devin:

- **S1 (varianta 1 0AF0H – 0AF2H):**

$$S1 = !A9 + A8 + !A7 = (\sim Y5)$$

- **S2 (varianta 2 0BF0H – 0BF2H):**

$$S2 = !A9 + !A8 + !A7 = (\sim Y7)$$

Similar interfeței seriale, interfața paralelă (8255) poate fi mapată în două zone de adresare diferite, controlate de comutatorul **S2**. Adresele pentru porturile A, B, C și registrul de control (RCC) sunt definite în funcție de valorile bitilor A9, A8 și A7.

Semnalele de selecție obținute din decodor sunt:

- **Pentru varianta 1 (0C70h–0C76h):**

$$P1 = A9 + !A8 + A7 = (\sim Y2)$$

- **Pentru varianta 2 (0D70h–0D76h):**

$$P2 = A9 + A8 + A7 = (\sim Y0)$$

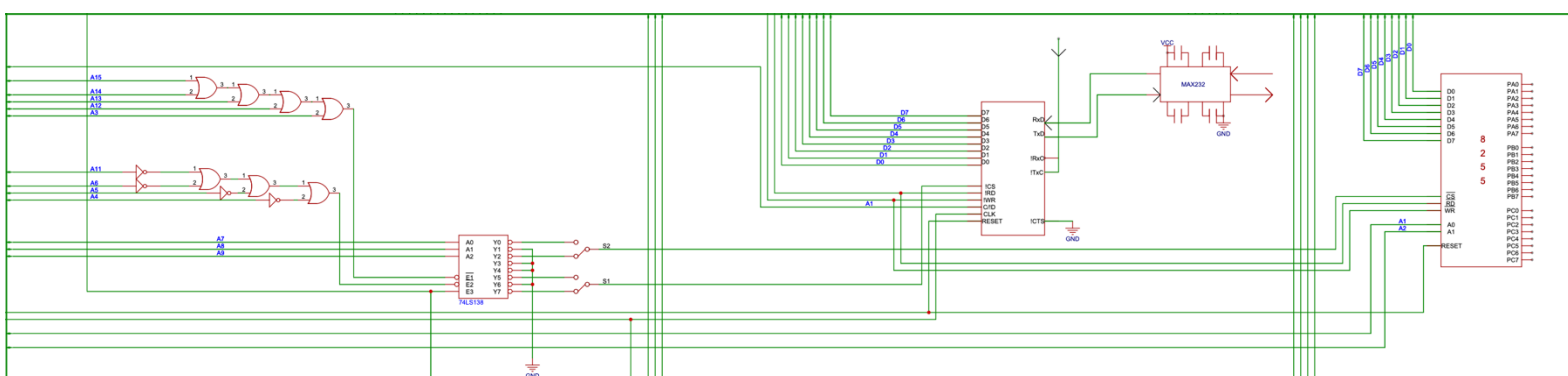


Figura 3. Interfața Serială și Paralelă în EasyEDA

CONECTAREA MINITASTATURII, LEDURILOR ȘI AFISAJELOR

Conectarea minitastaturii

Pentru conectarea minitastaturii cu 12 contacte (organizată în matrice 3×4), sunt necesare două porturi distincte:

- un **port de ieșire** pentru generarea semnalelor de scanare (coloanele tastaturii)
- un **port de intrare** pentru citirea liniilor tastaturii

Aceste porturi sunt implementate utilizând circuite latch/buffer și sunt accesate de microprocesor prin două adrese I/O dedicate.

Pentru generarea semnalelor de ieșire necesare activării coloanelor se utilizează portul **ST1 (0280h)**, implementat printr-un circuit **74LS373**. Ieșirile Q0–Q2 ale latch-ului sunt conectate direct la cele trei coloane ale tastaturii.

Cele patru linii reprezintă semnalele de intrare către sistem. Acestea sunt conectate la portul **ST2 (0290h)** prin intermediul unui buffer **74LS244/74LS245**, care izolează tastatura de magistrala de date. Fiecare linie este trasă în mod implicit la nivel logic '1' prin intermediul unor rezistențe de tip pull-up (10 k Ω), nivelul fiind coborât la '0' atunci când o tastă aflată pe coloana activă este apăsată.

Portul ST1 permite scanarea secvențială a coloanelor, iar portul ST2 permite citirea stării liniilor, software-ul putând identifica în mod precis tasta apăsată prin combinarea celor două informații.

- Portul de ieșire: **0280h**
- Portul de intrare: **0290h**

Conectarea LED-urilor

Modulul de LED-uri este conectat la portul de ieșire **SL1 (02A0h)**, implementat de asemenea cu un circuit **74LS373**. Biții D0–D5 ai magistralei de date sunt memorizați în latch, iar ieșirile Q0–Q5 controlează direct cele **șase** LED-uri.

Fiecare LED este conectat printr-o rezistență de limitare de **330 Ω** , asigurând un curent adecvat și protecția atât a LED-urilor, cât și a circuitelor logice. Activarea fiecărui LED depinde de starea logică a bitului corespunzător din registrul SL1, software-ul având posibilitatea de a aprinde sau stinge individual fiecare LED.

Această soluție oferă o conectare simplă, sigură și cu un număr minim de componente externe.

- Portul de ieșire: **02A0h**

Conectarea afișajelor cu 7 segmente și 6 ranguri

Afișajul numeric al microsistemului este realizat din șase module de 7 segmente, comandate printr-o soluție multiplexată pentru reducerea numărului de linii de control și a componentelor hardware necesare. Pentru comandarea segmentelor (a–g, dp) se utilizează portul **SA1 (02B0h)** implementat cu un circuit **74LS373**, ale cărui ieșiri Q0–Q7 sunt conectate prin rezistențe de **330 Ω** la segmentele corespunzătoare ale tuturor celor șase afișaje.

Selectarea rangurilor este realizată prin portul **SA2 (02C0h)**, implementat tot cu un **74LS373**. Ieșirile Q0–Q5 comandă, prin rezistențe de **5.1 k Ω** , bazele tranzistoarelor PNP (tip **BC557**) care alimentează anodul comun al fiecărui afișaj. Astfel, la un moment dat este activată o singură cifră, în timp ce segmentele sunt stabilite global prin SA1.

Multiplexarea dinamică permite afișarea tuturor celor șase cifre utilizând doar două porturi I/O. În această tehnică, segmentele sunt comune, iar rangurile sunt activate pe rând, pentru intervale scurte de timp. Procesorul scrie codul segmentelor în SA1, activează un singur rang prin SA2, apoi trece rapid la următorul. Dacă acest ciclu se repetă la frecvențe mai mari de aproximativ 25–30 Hz, afișajul este perceput ca fiind complet și stabil datorită inerției vizuale.

Avantajele metodei sunt consumul redus de curent, numărul minim de componente și flexibilitatea ridicată, iar singurul dezavantaj este necesitatea unei rutine software dedicate pentru comutarea rapidă a rangurilor.

- Portul de ieșire: **02B0h**
- Portul de ieșire: **02C0h**

Caracter	Valoare in Hex	Caracter	Valoare in Hex
0	C0H	5	92H
1	F9H	6	82H
2	A4H	7	F8H
3	B0H	8	80H
4	99H	9	90H

Tabelul cu porturi

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
0280h	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	ST1
0290h	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	ST2
02A0h	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	SL1
02B0h	0	0	0	0	0	0	1	0	1	0	1	1	0	0	0	0	SA1
02C0h	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	SA2

Pentru a permite selectarea corectă a fiecărui port din intervalul 0280h–02C0h, este necesară definirea condițiilor de activare ale decodificatorului 74x138. Intrările de enable (/E1, /E2 și E3) restricționează funcționarea decodificatorului doar la adresele din zona destinată interfețelor de I/O, iar cele trei linii de adresă A4, A5 și A6 determină ieșirea activă (Y0–Y4). În acest mod, fiecare port ST1, ST2, SL1, SA1 și SA2 primește propriul semnal de selecție, generat exclusiv atunci când CPU accesează adresa corespunzătoare.

$$!E1 = A15 + A14 + A13 + A12 + A11 + A10 + A8 + A3 + A2 + A1$$

$$!E2 = \sim A9 + \sim A7$$

$$E3 = \sim M / IO$$

$$ST1 = A6 + A5 + A4 = !Y0$$

$$ST2 = A6 + A5 + \sim A4 = !Y1$$

$$SL1 = A6 + \sim A5 + A4 = !Y2$$

$$SA1 = A6 + \sim A5 + \sim A4 = !Y3$$

$$SA2 = \sim A6 + A5 + A4 = !Y4$$

Circuitul 8253

Circuitul 8253 (Programmable Interval Timer) este utilizat în cadrul acestui proiect pentru realizarea temporizării necesare multiplexării modulului de afișare cu 7 segmente. Circuitul conține trei canale de numărare independente, fiecare având intrări dedicate de ceas (CLK), semnal de validare (GATE) și o ieșire (OUT).

În aplicația de față este utilizat un singur canal al temporizatorului, configurat în **Mode 2 (Rate Generator)**, mod de funcționare care permite generarea unui semnal periodic stabil. Semnalul de ceas aplicat pe

intrarea **CLK** este divizat în funcție de valoarea încărcată în contor, iar la ieșirea **OUT** se obține un tren de impulsuri cu frecvență determinată de această valoare. Intrarea **GATE** este menținută activ permanent, permițând funcționarea continuă a temporizatorului.

Semnalul generat la ieșirea **OUT** este utilizat pentru comutarea succesivă a celor 6 ranguri ale afișajului cu 7 segmente, realizând astfel multiplexarea acestora. Datorită frecvenței suficient de ridicate a semnalului de temporizare, afișarea este percepută ca fiind continuă și lipsită de pâlpâire. Utilizarea temporizării hardware cu ajutorul circuitului 8253 reduce încărcarea unității centrale și asigură o funcționare stabilă a sistemului de afișare.

Funcție 8253	Adresă
Counter 0	0300H
Counter 1	0302H
Counter 2	0304H
Control	0306H

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
0300H	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	Counter 0
0302H	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	Counter 1
0304H	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	Counter 2
0306H	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	Control

!E1 = A15 + A14 + A13 + A12 + A11 + A10 + A7 + A6 + A5 + A4 + A3

!E2 = ~A9 + ~A8

E3 =!(M/~IO

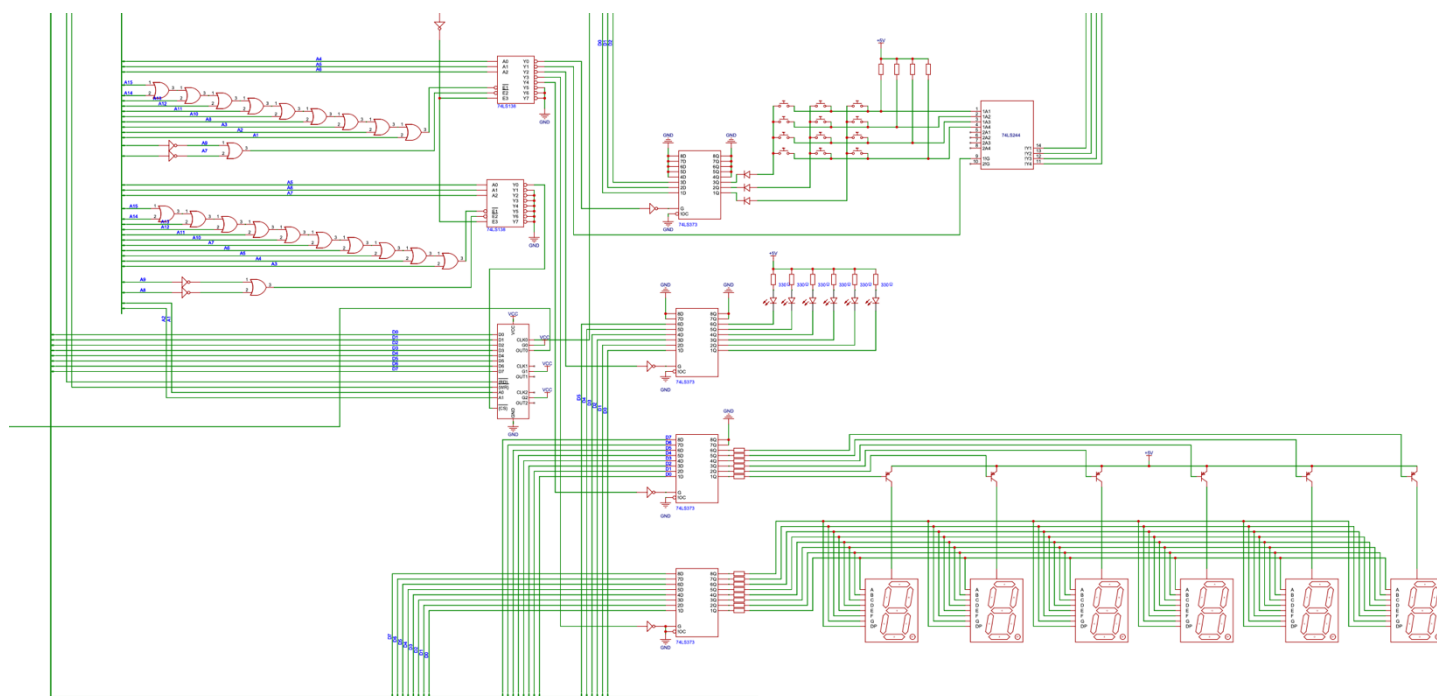


Figura 4. Ledurile, MiniTastatura si Afisajele in EasyEda

RUTINE

Rutina de programare 8251 (interfață serială)

```
INIT_8251 PROC
    MOV DX, 0AF0H          ; port comenzi 8251

    MOV AL, 0CEH           ; cuvânt de mod (async, 8 biti, fara paritate)
    OUT DX, AL             ; trimiți valoarea către 8251

    MOV AL, 15H            ; cuvânt de comandă (Tx/Rx enable)
    OUT DX, AL             ; activează transmisia și recepția

    RET
INIT_8251 ENDP
```

Rutina de EMISIE caracter – serial

```
SERIAL_TX PROC
    MOV DX, 0AF0H          ; port stare

TX_WAIT:
    IN AL, DX              ; citire cuvânt de stare
    RCR AL, 1              ; test TxRDY
    JNC TX_WAIT

    MOV AL, CL              ; caracter de transmis
    MOV DX, 0AF2H          ; port date
    OUT DX, AL

    RET
SERIAL_TX ENDP
```

Rutina de RECEPȚIE caracter – serial

```
SERIAL_RX PROC
    MOV DX, 0AF0H          ; port stare

RX_WAIT:
    IN AL, DX              ; citire cuvânt de stare
    RCR AL, 2              ; test RxRDY
    JNC RX_WAIT

    MOV DX, 0AF2H          ; port date
    IN AL, DX
    MOV CL, AL              ; caracter recepționat

    RET
SERIAL_RX ENDP
```

Rutina de programare 8255 (interfață paralela)

```
INIT_8255 PROC
    MOV DX, 0D76H        ; RCC 8255
    MOV AL, 81H          ; cuvânt de control
    OUT DX, AL
    RET
INIT_8255 ENDP
```

Rutina de EMISIE caracter – paralel

```
PARALLEL_TX PROC
    MOV DX, 0D74H        ; port C (BUSY)

PAR:
    IN AL, DX            ; citire BUSY
    RCR AL, 1            ; test BUSY
    JNC PAR              ; așteaptă cât timp BUSY = 0

    MOV AL, CL           ; caracterul de transmis
    MOV DX, 0D70H        ; port A (date)
    OUT DX, AL

    OR AL, 01H
    MOV DX, 0D72H        ; port B
    OUT DX, AL           ; STB = 1

    AND AL, 00H
    OUT DX, AL           ; STB = 0

    OR AL, 01H
    OUT DX, AL           ; STB = 1

    RET
PARALLEL_TX ENDP
```

Rutina de aprindere / stingere LED

LED-urile sunt **active pe 0** (anod comun).

	XX	
Led 1	11111110	FEH
Led 2	11111101	FDH
Led 3	11111011	FBH
Led 4	11110111	F7H
Led 5	11101111	EFH
Led 6	11011111	DFH

Rutina de aprindere a unui LED

```
MOV DX, 02A0h
MOV AL, XX ;
OUT DX, AL
```

Rutina de stingere a unui LED

```
MOV AL, FFH;
OUT DX, AL
```

Rutina de scanare a minitastaturii (4×3)

```
ST1 EQU 0290h      ; ieşire - coloane
ST2 EQU 0280h      ; intrare - linii
```

SCAN_KEYPAD:

REIA:

; Coloana 1 (1, 4, 7, *)

```
MOV AL, FEH        ; 1111 1110
OUT ST1, AL        ;selectez ST1
```

```
IN  AL, ST2
AND AL, 01H
JZ  TASTA1
```

```
IN  AL, ST2
AND AL, 02H
JZ  TASTA4
```

```
IN  AL, ST2
AND AL, 04H
JZ  TASTA7
```

```
IN  AL, ST2
AND AL, 08H
JZ  TASTA_STEA
```

;Coloana 2 (2, 5, 8, 0)

```
MOV AL, FDH        ; 1111 1101
OUT ST1, AL
```

```
IN  AL, ST2
AND AL, 01H
JZ  TASTA2
```

```
IN  AL, ST2
AND AL, 02H
JZ  TASTA5
```

```
IN  AL, ST2
AND AL, 04H
JZ  TASTA8
```

```
IN  AL, ST2
AND AL, 08H
JZ  TASTA0
```

; Coloana 3 (3, 6, 9, #)

```
MOV AL, 0FBH      ; 1111 1011
OUT ST1, AL
```

```
IN  AL, ST2
AND AL, 01H
JZ  TASTA3
```

```
IN  AL, ST2
AND AL, 02H
JZ  TASTA6
```

```
IN  AL, ST2
AND AL, 04H
JZ  TASTA9
```

```
IN  AL, ST2
AND AL, 08H
JZ  TASTA_DIEZ
```

```
JMP REIA
```

; Tratare TASTA 1

```
TASTA1: CALL DELAY      ; stabilizare contacte
AST1: IN  AL, ST2
      AND AL, 01H
      JZ  AST1           ; așteaptă eliberarea tastei

      CALL DELAY
      MOV CL, 00H
      RET
```

Analog pentru celelalte taste

Rutina pentru afisaje cu 6 ranguri

```
SEG_PORT    EQU 02B0H    ; latch segmente (cifra)
RANG_PORT    EQU 02C0H    ; latch rang (selectie digit)

TMR0_PORT    EQU 0300H    ; 8253 Counter 0
TMR1_PORT    EQU 0302H    ; 8253 Counter 1 (nefolosit)
TMR2_PORT    EQU 0304H    ; 8253 Counter 2 (nefolosit)
TMR_CTRL     EQU 0306H    ; 8253 Control Word Register
.data
CODES DB 0C0H, 0F9H, 0A4H, 0B0H
      DB 099H, 092H, 082H, 0F8H
      DB 080H, 090H        ; 0-9 (anod comun, activ pe 0 la
                           ; segmente)

RANG_MASK DB 01H, 02H, 04H, 08H, 10H, 20H ; rang 0..5 (one-hot)
```

```

DISPLAY    DB 1, 2, 3, 4, 5, 6
CURR_RANG DB 0

.code
; ISR - AFISAJ MULTIPLEXAT (INT 08h)

ISR_DISPLAY PROC
    PUSH AX
    PUSH BX
    PUSH DX
    PUSH SI

    ; index rang curent (0..5)
    MOV BL, CURR_RANG
    XOR BH, BH

    ; BLANKING (anti-ghosting): opreste toate rangurile
    MOV DX, RANG_PORT
    MOV AL, 00H
    OUT DX, AL

    ; cifra de afisat (0..9)
    MOV SI, OFFSET DISPLAY
    MOV AL, [SI + BX]

    ; conversie cifra -> cod segmente prin XLAT
    PUSH BX
    MOV BX, OFFSET CODES
    XLAT          ; XLAT foloseşte AL ca index în tabelul CODES
    POP BX

    ; scrie segmentele(trimis către latch-ul de segmente)
    MOV DX, SEG_PORT
    OUT DX, AL

    ; activeaza rangul curent
    MOV SI, OFFSET RANG_MASK
    MOV AL, [SI + BX]
    MOV DX, RANG_PORT
    OUT DX, AL

    ; rang urmator
    MOV AL, CURR_RANG
    INC AL
    CMP AL, 6
    JL  short ok
    MOV AL, 0
ok:
    MOV CURR_RANG, AL
    POP SI
    POP DX
    POP BX
    POP AX

```

```

    IRET
ISR_DISPLAY END
install_isr PROC ; INSTALARE VECTOR INT 08h IN IVT (0000:0000)
    PUSH AX
    PUSH BX
    PUSH ES

    CLI    ; Dezactivează întreruperile în timpul modificării IVT

    XOR AX, AX
    MOV ES, AX                ; ES=0000h (IVT)

    MOV BX, 8 * 4             ; INT 08h -> intrare la offset 0020h
    MOV AX, OFFSET ISR_DISPLAY
    MOV ES:[BX], AX           ; ES:[BX] = locația unde se salvează IP-ul ISR-ului
    MOV AX, CS
    MOV ES:[BX+2], AX         ; ES:[BX+2] = partea de segment a vectorului

    POP ES
    POP BX
    POP AX
    RET
install_isr ENDP

init8253 PROC                ; INIT 8253 - TIMER PENTRU AFISAJ (CH0, MODE 2)
    PUSH AX
    PUSH DX

    ; Control Word: Counter0, LSB+MSB, Mode2, Binary => 34h
    MOV DX, TMR_CTRL
    MOV AL, 34H
    OUT DX, AL

    ; Valoare numarare pentru ~1ms
    MOV DX, TMR0_PORT
    MOV AL, E8H                ; LSB
    OUT DX, AL
    MOV AL, 03H                ; MSB
    OUT DX, AL

    POP DX
    POP AX
    RET
init8253 ENDP

; MAIN
start:
    MOV AX, @data
    MOV DS, AX
    CALL install_isr
    CALL init8253
    STI ; Activarea întreruperilor globale.

main_loop:
    JMP main_loop
END start

```

