

Universitatea Politehnica Timișoara
Facultatea de Automatică și Calculatoare
Proiectarea Microsistemelor Digitale

Microsistem cu microprocesorul 8086

Student: Diriczi Zsolt-Csaba
2023-2024

1. Tema proiectului

Proiectul propus constă în proiectarea unui microsistem având următoarea configurație:

- Unitate centrală echipată cu microprocesorul 8086.
- Memorii EPROM de 128 KB, utilizând circuitele 27C1024.
- Memorii SRAM de 256 KB, utilizând circuitele 62512.
- Interfață serială cu circuitul 8251, plasată în zona 0DD0H – 0DD2H sau 0F50H – 0F52H, în funcție de poziția microcomutatorului S1.
- Interfață paralelă cu circuitul 8255, amplasată în zona 0150H – 0156H sau 0A50H – 0A56H, în funcție de poziția microcomutatorului S2.
- Minitastatură cu 16 contacte.
- 16 LED-uri.
- Două module de afișare cu segmente, având fiecare 4 ranguri.

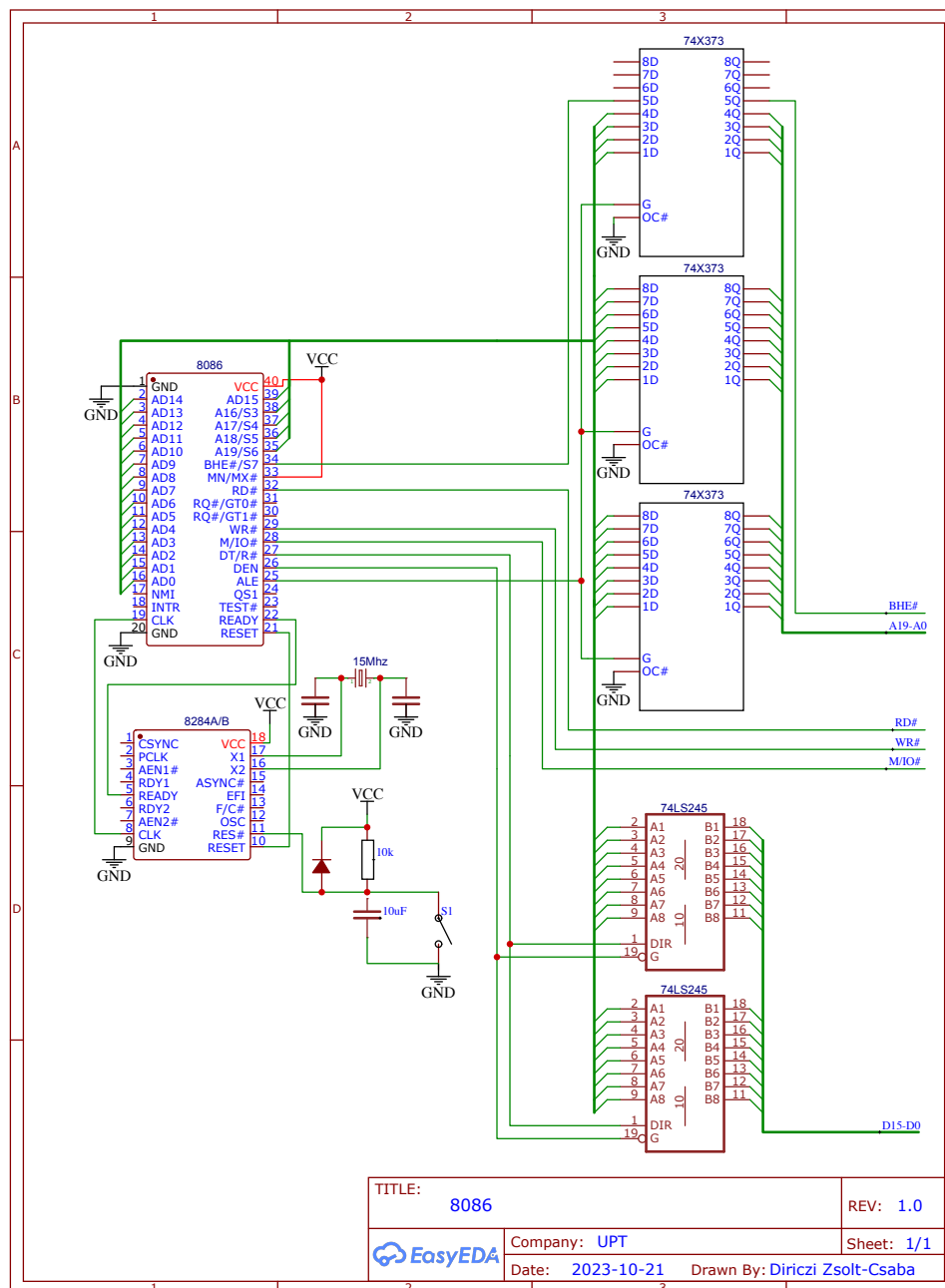
Toate programele în limbaj de asamblare vor fi concepute sub formă de subrutine.

Rutinele necesare sunt:

- Rutinele de programare ale circuitelor 8251 și 8255.
- Rutinele de emisie/recepție de caractere pe interfața serială.
- Rutina de emisie a unui caracter pe interfața paralelă.
- Rutina de scanare a minitastaturii.
- Rutina de aprindere/stingere a unui LED.
- Rutina de afișare a unui caracter hexazecimal pe un rang cu segmente.

2. Descriere Hardware

2.1 Unitatea centrala cu procesorul 8086



Circuite folosite:

- Microprocesorul 8086
- Circuit amplificator/ separare bidirecțională 74x245

Acest circuit este folosit pentru amplificarea și separarea magistralei de date bidirecționale ale microprocesorului 8086. În esență, acesta facilitează transferul bidirecțional de date între magistrala sistemului și alte componente ale sistemului.

- Circuit registru 74x373

Acest circuit este un registru cu 8 ranguri și 3 stări. Acest registru are 8 celule de stocare și poate opera în 3 stări diferite: Load, Store și Tri-state și este folosit pentru a menține adresele date de microprocesorul 8086 pe toată durata ciclului și a elibera magistrala pentru a putea fi folosită la transferul de date.

- Circuit generator de tact 8284A

Acest circuit folosește un cuarț de frecvență 15Mhz, însă la ieșirea CLK obținem un tact cu frecvență egală cu 1/3 din frecvența cuarțului (5Mhz), pe când la ieșirea PCLK obținem o frecvență egală cu 1/6 din cea a cuarțului (2,5Mhz).

În continuare prezentăm terminalele cele mai importante ale circuitului 8086, rolul și ce am conectat la acestea:

- AD15 – AD0 - Address Data Bus, ieșiri pentru magistrala de date și adresă multiplexată prin urmare sunt intrări pentru primele 2 registre 74X373(adrese) cât și pentru 2 circuite 74x245(date).
- A19 – A16 - rangurile 19-16 din magistrala de adrese prin urmare și acestea sunt conectate la un registru 74x373.
- BHE# - Bus High Enable, ieșire ce indică dacă are sau nu loc un transfer pe jumătatea superioară a magistralei de date conectată la registru 74x373.
- RD# - Read Control, ieșire cu 3 stări, activă atunci când microprocesorul execută un ciclu de citire sau de intrare.
- WR# - Write Control, ieșire cu 3 stări, activă atunci când microprocesorul execută un ciclu de scriere sau de ieșire.
- READY - Wait State Control, intrare pentru sincronizarea cu circuitele de memorie și porturile mai lente.
- RESET - System Reset, intrare pentru inițializarea microprocesorului și este conectat la ieșirea de RESET a generatorului de tact. Astfel când se apasă butonul S1 atât generatorul de tact cât și microprocesorul 8086 sunt reinițializate.
- CLK - System Clock, intrare de tact cu frecvență de 5Mhz și factor de umplere 1/3. Aici am conectat ieșirea CLK a generatorului de tact care furnizează un clock de frecvență 5Mhz.
- MN/MX# - intrare pentru selecția modului de lucru, am folosit modul minim legând acest pin la alimentare, modul maxim fiind folosit în aplicații mai complexe în care semnalele de comandă pentru memorii și porturi sunt generate de un controler de magistrală
- ALE - Address Latch Enable, ieșire care se activează atunci când pe magistrală multiplexată de adrese/date avem adrese, din acest motiv am conectat această ieșire la enable-ul registrelor 74X373 .
- DT/R# - Data Transmit/Receive, ieșire cu 3 stări, care indică sensul transferului pe magistrala de date: 1 indică transmisie și 0 recepție prin urmare am legat la intrarea DIR a circuitelor 74x245
- DEN# - Data Enable, ieșire cu 3 stări, care validează transferul de date pe magistrală și prin urmare l-am conectat la intrarea de enable(G#) a circuitului 74x245

2.2 Conectarea memoriilor

2.2.1 Harta memoriei

Circuit	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
SRAM1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SRAM1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
SRAM2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
SRAM2	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
SRAM3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SRAM3	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
SRAM4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
SRAM4	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
EPROM	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EPROM	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

2.2.2 Proiectarea decodicatorului pentru memorii

În continuare voi detalia proiectarea decodicatorului corespunzător următoarei hărți a memoriei:

- SRAM1, SRAM2 la adresa 20000H - 3FFFFH
- SRAM3, SRAM4 la adresa 40000H - 5FFFFH
- EPROM la adresa E0000H - FFFFFH

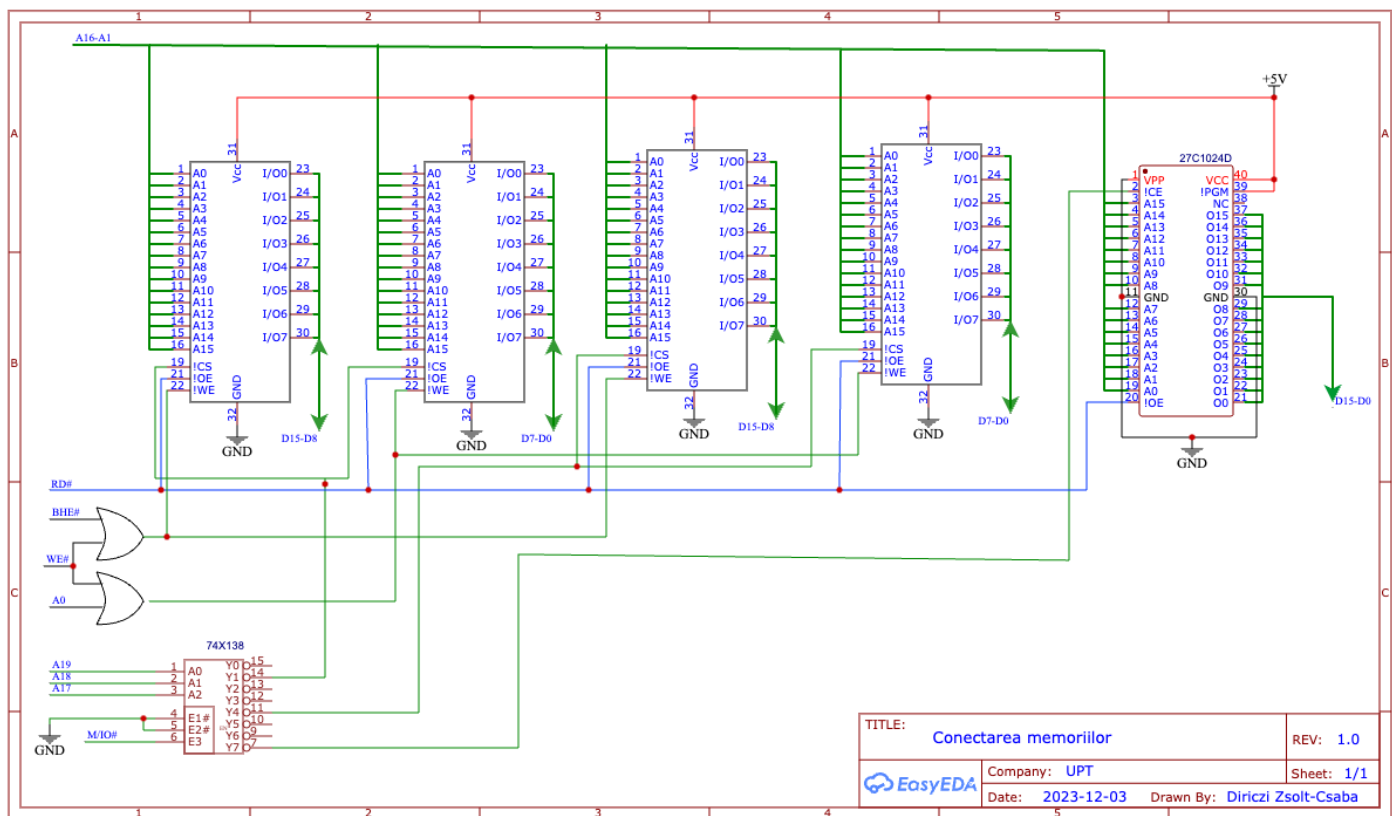
Pentru activarea decodicatorului de memorii am conectat la:

- E1#, E2#: fiind active pe nivel logic inferior le-am conectat la masa.
- E3: fiind activa pe nivel logic superior, am conectat M/IO#, deoarece atunci când acesta ieșire a microprocesorului 8086 este activa atunci se dorește adresarea memoriei și nu a porturilor.

Considerând o decodificare completa ecuațiile de selecție pentru memorii sunt:

1. $\overline{\text{Sel_SRAM1,2}} = \overline{A19 \& A18 \& A17} = A19 + A18 + \overline{A17}$
2. $\overline{\text{Sel_SRAM2,3}} = \overline{A19 \& A18 \& \overline{A17}} = A19 + \overline{A18} + A17$
3. $\overline{\text{Sel_EPROM}} = \overline{A19 \& A18 \& A17} = \overline{A19} + \overline{A18} + \overline{A17}$

2.2.3 Schema hardware pentru conectarea memoriilor



În cele ce urmează, voi detalia procesul prin care am interconectat modulele de memorie.

Având în vedere necesitatea unei memorii EPROM de 128 KB, am optat pentru circuitele 27C1024, având o dimensiune de 128 KB. Acestea sunt organizate în 65536 de cuvinte, fiecare cu o lungime de 16 biți. Astfel, am constatat că un singur astfel de circuit este suficient pentru a satisface cerințele noastre.

În contextul memoriei SRAM de 256 KB, am folosit circuitele 62512, având o dimensiune de 64KB fiecare. Acestea sunt dispuse într-o structură organizată în 65536 de cuvinte, fiecare cuvânt fiind pe 8 biți. Astfel, pentru a atinge capacitatea totală a memoriei SRAM de 256 KB, am integrat 4 astfel de circuite.

De menționat este faptul că, pentru cele 5 circuite utilizate, sunt necesare doar 3 linii de selecție din decodificator. Acest aspect se datorează faptului că, în cazul circuitelor SRAM, la fiecare adresă din 62512 avem doar 8 biți, în timp ce un word din memorie, asemenea circuitului EPROM, ocupă 16 biți. Prin urmare, rezultă o configurație de memorie SRAM adresabilă pe octeți, însă cu un word format din 2 octeți.

Așadar, prima ieșire din decodificator este conectată atât la semnalul CS# (Chip Select) al primului circuit SRAM, cât și la al celui de-al doilea. De asemenea, semnalul OE# este activat de semnalul RD# (Read Data) provenit de la microprocesorul 8086. Astfel, atunci când această intrare este activată, obținem "word"-ul de la adresa specificată de biții A16-A19.

În cazul în care dorim să scriem în SRAM, este necesar să avem din nou activat semnalul CS#. Prin intermediul unei porți logice "sau" între semnalul BHE# și WE# (Write Enable), semnale provenite de la microprocesorul 8086, conectată la intrarea WE# a memoriei, putem lua decizia de a scrie în octetul superior sau nu.

De asemenea, prin intermediul unei porți logice "sau" între semnalul A0# și WE#, putem decide dacă să scriem în octetul inferior. De remarcat este faptul că avem posibilitatea de a scrie atât în octetul superior, cât și în cel inferior sau în ambele simultan.

Aceleași principii au fost aplicate și pentru a doua pereche de circuite SRAM.

În cazul circuitelor EPROM microprocesorul 8086 poate doar să citească și nu poate scrie.

Acest circuit este activat prin intermediul unei ieșiri a decodicatorului conectată la semnalul CE# (Chip Enable). În acest context, la intrarea OE# am conectat aceeași linie cu RD# (Read Data) pentru a asigura sincronizarea adecvată a operațiilor de citire.

2.3 Interfata seriala si paralela

2.3.1 Harta porturilor

Circuit	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
IS1	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	1	0	0	0	0
IS1	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	1	0	0	1	0
IS2	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	1	0	0	0	0
IS2	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	1	0	0	1	0
IP1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0
IP1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	1	0
IP2	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0
IP2	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	1	1	0
Timer	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
Timer	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0

2.3.2 Proiectarea decodicatorului pentru interfața seriala, interfața paralela și timer

În continuare voi detalia proiectarea decodicatorului corespunzător următoarei hărți a porturilor:

- IS1 la adresa 00DD0H - 00DD2H
- IS2 la adresa 00F50H - 00F52H
- IP1 la adresa 00150H - 00156H
- IP2 la adresa 00A50H - 00A56H
- Timer la adresa 000C0H - 000C6H

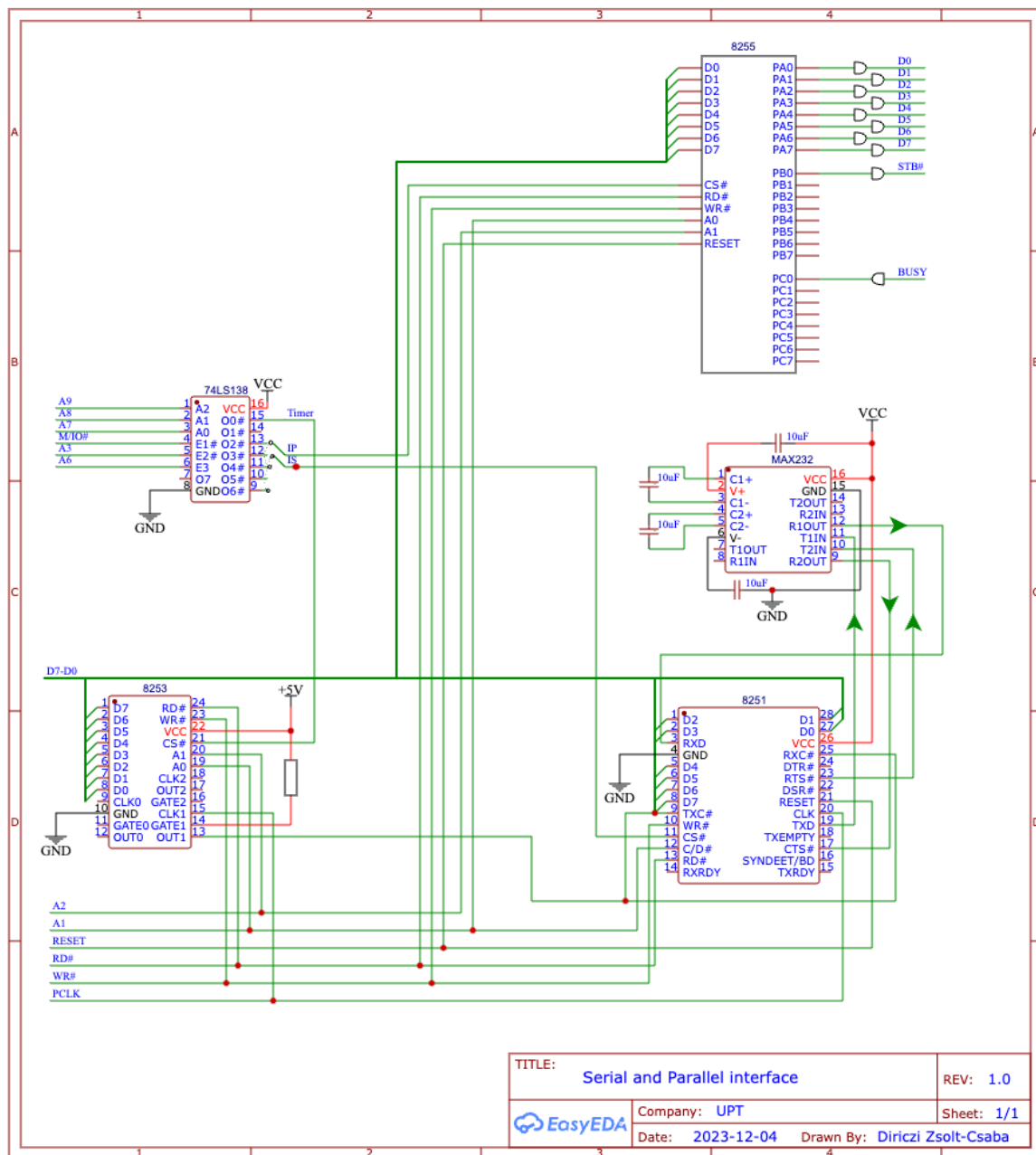
Pentru activarea decodicatorului de porturi am conectat la:

- La linia E1#, activă la nivel logic inferior, am conectat semnalul M/IO#, o ieșire a microprocesorului 8086. Acest semnal, când se află la nivel logic inferior, indică utilizarea porturilor.
- E2#, fiind activ pe nivel logic inferior, am conectat la bitul A3 deoarece acesta este mereu 0 atunci când lucrăm cu interfața paralela și seriala.
- Linia E3, activă la nivel logic superior, a fost conectată la A6. Această alegere are în vedere faptul că, în utilizarea interfețelor serială și paralelă, linia A6 este întotdeauna activă. Combinația binară dintre A6 și A3, 10b, ne ajută să facem distincția față de adresarea altor porturi.

Prin urmare ecuațiile de selecție pentru interfața seriala, paralela și timer sunt:

$$\begin{aligned}\overline{\text{Sel}}_{\text{IS1}} &= \overline{\text{A9}} \& \overline{\text{A8}} \& \overline{\text{A7}} = \text{A9} \mid \overline{\text{A8}} \mid \overline{\text{A7}} \\ \overline{\text{Sel}}_{\text{IS2}} &= \overline{\text{A9}} \& \overline{\text{A8}} \& \overline{\text{A7}} = \overline{\text{A9}} \mid \overline{\text{A8}} \mid \overline{\text{A7}} \\ \overline{\text{Sel}}_{\text{IP1}} &= \overline{\text{A9}} \& \overline{\text{A8}} \& \overline{\text{A7}} = \text{A9} \mid \overline{\text{A8}} \mid \overline{\text{A7}} \\ \overline{\text{Sel}}_{\text{IP2}} &= \overline{\text{A9}} \& \overline{\text{A8}} \& \overline{\text{A7}} = \overline{\text{A9}} \mid \overline{\text{A8}} \mid \overline{\text{A7}} \\ \overline{\text{Sel}}_{\text{Timer}} &= \overline{\text{A9}} \& \overline{\text{A8}} \& \overline{\text{A7}} = \text{A9} \mid \overline{\text{A8}} \mid \overline{\text{A7}}\end{aligned}$$

2.3.3 Schema hardware pentru conectarea interfeței seriale, interfeței paralele și timbrului



În cele ce urmează, voi detalia procesul prin care am interconectat interfața serială, paralelă și timer-ul necesar interfeței seriale.

2.3.3.1 Interfața serială

Pentru interfața serială am folosit circuitul specializat programabil 8251. Acest circuit face parte din categoria circuitelor de tip USART ce poate primi de la unitatea centrală în paralel un octet pe care îl transmite mai departe în mod serial (bit cu bit) către un echipament serial, sau poate să preia de la echipamente periferice seriale, un octet, să îl assembleze și să îl predea unității centrale.

Pentru a activa acest circuit, am conectat intrarea CS# la un comutator IS, inițial legat la ieșirea 3 a decodificatorului. Totuși, prin intermediul comutatorului, avem posibilitatea să îl mutăm la ieșirea 6 a decodificatorului. Această configurare ne permite să utilizăm circuitul cu două adrese diferite, în funcție de poziția comutatorului (zona 00DD0H-0DD2H sau 00F50H-00F52H).

De remarcat este faptul ca avem 2 adrese în fiecare din cele 2 zone, acest lucru este deoarece cu bitul 1 din adresa (A1) vom selecta dacă dorim să încărcăm cei 8 biți primiți la intrare în tamponul de date sau este un cuvânt de comanda/control. Astfel după cum se va vedea și din program, inițial când dorim inițializarea circuitului, setăm A1 pe valoarea 1 pentru a trimite cuvânt de comanda.

Prin urmare din acest motiv am conectat A1 la intrarea C/D# - Control/Data (când A1 este "1" avem cuvinte de comanda iar când A1 este "0" avem date ce trebuie transmise sau ce trebuie predate unității centrale).

La intrarea RD#, am conectat ieșirea RD# a microprocesorului 8086, care când este activa indica faptul ca vrem să preluăm datele recepționate de la 8251.

Similar, la intrarea WR#, am conectat ieșirea WR# a microprocesorului 8086, care când este activa indica faptul ca vrem să trimitem date la 8251, fie pentru a fi transmise serial, fie cuvinte de comanda și control.

Tactul acestui circuit este obținut prin conectarea ieșirii circuitului 8086 PCLK (frecvența de 2,5MHz) la intrarea de CLK a acesteia. În datasheet se menționează faptul ca frecvența acesteia trebuie să fie mai mare decât $30 \times \text{bit rate-ul receptorului/transmitatorului}$.

De remarcat este faptul ca avem nevoie și de circuitul MAX 232. Acest circuit este necesar deoarece nivelele de tensiune folosite de de interfața RS232 nu sunt TTL ci EIA adică

- -25V -3V pentru "1" logic
- 3V -25V pentru "0" logic.

Prin urmare circuitul MAX 232 este folosit pentru conversia din TTL la EIA la transmitere de date și din EIA în TTL la recepție de date. Astfel transferul serial de tip RS232 are un mare avantaj: prezintă o rezistență mai mare la perturbații decât cel paralel datorită distanței dintre nivelele de tensiune corespunzătoare celor 2 nivele logice.

Astfel acest circuit este folosit pentru conversia din TTL în EIA și invers a semnalelor RXD, TXD, CTS# și RTS#, DTR#, și DSR#.

Terminale 8251 folosite pentru comunicarea serială:

- TXD este un semnal de ieșire și este folosit pentru transmiterea unui bit.
- RXD este un semnal de intrare prin care se recepționează un bit.
- DTR# (Data Terminal Ready), un semnal de ieșire activ pe nivel logic inferior, este activat în momentul în care circuitul este inițializat și pregătit să comunice.
- DSR# (Data Set Ready), un semnal de intrare activ pe nivel logic inferior, este activat atunci când "interlocutorul" (un alt circuit cu interfața serială) este pregătit să comunice.
- RTS# (Request to Send), un semnal de ieșire activ pe nivel logic inferior, activat când circuitul vrea să transmită date.
- CTS# (Clear to Send), un semnal de intrare activ pe nivel logic inferior, activat când "interlocutorul" este pregătit să recepționeze date.

Prin urmare semnalul DTR# al unui circuit este conectat la intrarea DSR# al celuilalt circuit și invers. La fel semnalul RTS# al unui circuit este conectat la intrarea CTS# al celuilalt circuit și invers.

Așadar aceste semnale permit ca două circuite să comunice prin interfața serială, să poată anunța unul pe altul când sunt pregătiți și pot transmite, respectiv recepționa date (cu ajutorul celor 4 semnale se efectuează un "handshake").

Un lucru important de menționat este faptul ca în cazul în care 2 sisteme comunică prin interfața serială este obligatoriu ca ambele să fie conectate la o masă (ground) comună.

2.3.3.2 Timer

Pentru ca circuitul 8251 sa funcționeze corect este necesar și un timer 8253 care va determina intervalele de timp la care se transmit datele. Practic acesta va dicta frecvența de transmitere și de recepție de date, prin urmare ambele circuite trebuie sa transmita și sa recepționeze cu aceeași frecvență. Pentru acest lucru trebuie sa alegem un baud rate respectiv un factor de multiplicare. Valorile alese de mine pentru acestea vor fi prezentate la program.

Astfel avem 8253 va fi activat și el de o intrare din decodificator după cum a fost prezentat și la proiectarea decodificatorului.

2.3.3.3 Interfața paralela

Spre deosebire de transferul serial, la care transferul datelor se face bit după bit, la transferul paralel se transfera 8 biți simultan iar transferul este însoțit de semnale de dialog.

Pentru interfața paralela, am folosit circuitul 8255.

Acesta dispune de 24 linii de intrare/ieșire care pot fi configurate în mai multe feluri în funcție de modul de lucru ales.

Comunicarea cu circuitul 8255 este realizată printr-un set de 4 adrese de port, corespunzătoare porturilor A, B, C și portului destinat cuvântului de comandă. Din această cauză, am asignat câte 4 adrese pentru interfața paralelă în fiecare dintre cele două zone de memorie (0150H-0156H și 0A50H-0A56H). De exemplu, adresa 0150H se potrivește cu portul intern A, 0152H corespunde portului intern B, 0154H este asociat portului intern C, iar 0156H este destinat portului intern pentru cuvântul de comandă (RCC). Prin urmare, la intrările A1 și A0 ale circuitului 8255, am conectat biții A2 și A1 din magistrala de adrese.

În mod evident și la intrarea CS# a acestui circuit am conectat ieșirea switch-ului care poate fi conectat la una din cele 2 ieșiri ale decodificatorului: O2# sau O4# în funcție de poziția comutatorului.

La intrarea RD#, am conectat ieșirea RD# a microprocesorului 8086, care ne anunță prin activarea acesteia dacă dorim să citim date în mod paralel de la alt echipament, sau dacă citim starea acesteia.

Similar, la intrarea WR#, am conectat ieșirea WR# a microprocesorului 8086, care ne anunță prin activarea acesteia dacă dorim să transmitem date în mod paralel, sau dacă trimitem un cuvânt de comandă.

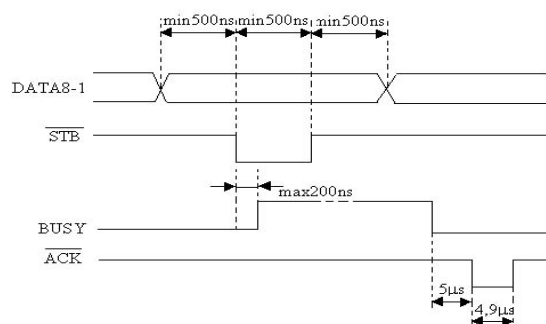
Modul de lucru pentru circuitul 8255 este astfel ales încât să lucreze în conformitate cu dialogul de tip CENTRONIX.

Astfel avem nevoie de intrarea BUSY cu ajutorul căreia putem verifica dacă receptorul este liber și poate recepționa datele. Avem nevoie și de semnalul de ieșire STB# care este activat când microprocesorul trimite date prin interfața paralelă.

De asemenea avem nevoie de 8 ieșiri prin care putem transmite datele.

Prin urmare vom folosi cele 3 porturi astfel

- Portul intern A - ieșire (pentru transmitere 8 biți)
- Portul intern B - ieșire (pentru ieșirea STB#)
- Portul intern C inferior - intrare (pentru intrarea BUSY)
- Portul intern C superior - nefolosit (intrare/ieșire)



2.4 Conectare afișaje și a minitastaturii.

2.4.1 Harta porturilor

Circuit	A1 9	A1 8	A1 7	A1 6	A1 5	A1 4	A1 3	A1 2	A1 1	A1 0	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Addr
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	08H
D1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	18H
D2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	28H
D3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	38H
D4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	48H
D5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	58H
D6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	68H
D7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	78H
K1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	88H
K2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	8AH
L1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	8CH
L2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	8EH

2.4.2 Proiectarea decodificatoarelor

În continuare voi detalia proiectarea decodicatorului corespunzător următoarei hărți a porturilor folosit pentru afișajele cu 7 segmente:

- D0 cu adresa 08H
- D1 cu adresa 18H
- D2 cu adresa 28H
- D3 cu adresa 38H
- D4 cu adresa 48H
- D5 cu adresa 58H
- D6 cu adresa 68H
- D7 cu adresa 78H

Pentru decodificarea de porturi folosesc circuitul decodicator LS138 cu 3 intrări de selecție și 8 ieșiri. De remarcat este faptul ca voi folosi o decodificare incompletă, astfel am marcat cu verde biții de adresa ce nu au fost considerați în procesul de decodificare, cu albastru biții care sunt folosiți pentru activarea decodicatorului, iar cu portocaliu biții de selecție pe baza căreia se selectează una din circuite (porturi).

Pentru activarea decodicatorului de afișaje am conectat la:

- E1#: fiind activa pe nivel logic inferior, am conectat ieșirea M/IO# din 8086 care atunci când este pe nivel logic inferior indica lucrul cu porturile de intrare/ieșire și prin urmare adresele folosite pentru adresarea memoriei nu vor determina suprapuneri cu cele ale porturilor.
- E2#: fiind activa tot pe nivel logic inferior, am conectat A7 .
- E3: fiind activa pe nivel logic superior, am conectat A3

Pentru selectarea unui display, având un număr de exact 8 display-uri, am ales sa baleiez biții A6, A5, A4 de la 000 la 111 astfel încât ieșirea 0 a decodicatorului va activa primul afișaj iar ieșirea 7 pe ultimul.

Prin urmare ecuațiile de selecție pentru afișaje sunt:

1. $\overline{\text{Sel_D0}} = \overline{A6} \& \overline{A5} \& \overline{A4} = A6 + A5 + A4$
2. $\overline{\text{Sel_D1}} = \overline{A6} \& \overline{A5} \& A4 = A6 + A5 + \overline{A4}$
3. $\overline{\text{Sel_D2}} = \overline{A6} \& A5 \& \overline{A4} = A6 + \overline{A5} + A4$
4. $\overline{\text{Sel_D3}} = \overline{A6} \& A5 \& A4 = A6 + \overline{A5} + \overline{A4}$
5. $\overline{\text{Sel_D4}} = A6 \& \overline{A5} \& \overline{A4} = \overline{A6} + A5 + A4$
6. $\overline{\text{Sel_D5}} = A6 \& \overline{A5} \& A4 = \overline{A6} + A5 + \overline{A4}$
7. $\overline{\text{Sel_D6}} = A6 \& A5 \& \overline{A4} = \overline{A6} + \overline{A5} + A4$
8. $\overline{\text{Sel_D7}} = A6 \& A5 \& A4 = \overline{A6} + \overline{A5} + \overline{A4}$

În continuare voi detalia proiectarea decodicatorului corespunzător următoarei hărți a porturilor folosit pentru LED-uri și minitastatura:

- K1 cu adresa 88H
- K2 cu adresa 8AH
- L1 cu adresa 8CH
- L2 cu adresa 8EH

La fel ca și în cazul anterior, folosesc o decodificare incompletă, culorile biților având aceeași semnificație.

Pentru activarea decodicatorului de minitastatura și LED-uri am conectat la:

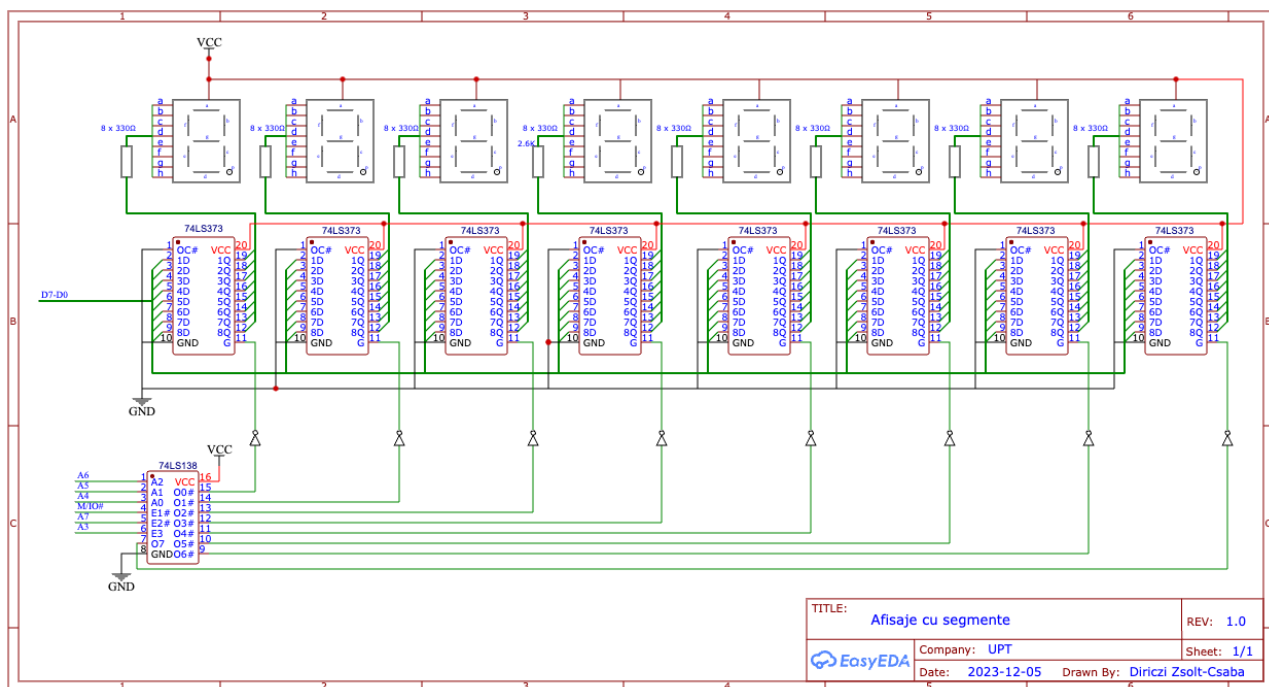
- E1#: fiind activă pe nivel logic inferior, am conectat ieșirea M/IO# din 8086 care în momentul în care este pe nivel logic inferior indica lucrul cu porturile de intrare/ieșire.
- E2#: fiind activă tot pe nivel logic inferior, am conectat A0 întrucât adresele de porturi sunt mereu pare și prin urmare A0 este mereu "0".
- E3: fiind activă pe nivel logic superior, am conectat A7 or A3 deoarece acești biți sunt în toate cele 4 cazuri pe nivel logic superior și ne ajută să diferențiem LED-urile și minitastatura față de celelalte periferice (afișaje, interfața serială, interfața paralelă, timer).

Pentru selectarea LED-urilor sau a minitastaturii având nevoie doar de 4 adrese a fost suficient să baleiez biții A2 și A1 de la 00 la 11.

Prin urmare ecuațiile de selecție pentru minitastatura și led sunt:

1. $\overline{\text{Sel_K1}} = A2 + A1$
2. $\overline{\text{Sel_K2}} = \overline{A2} \& A1 = A2 + \overline{A1}$
3. $\overline{\text{Sel_L1}} = A2 \& \overline{A1} = \overline{A2} + A1$
4. $\overline{\text{Sel_L2}} = \overline{A2} \& \overline{A1} = \overline{A2} + \overline{A1}$

2.4.3 Schema hardware



2.4.3.1 Afisaje cu segmente

Dupa cum se poate vedea am folosit afişaje cu anod comun, astfel activarea unui segment se face prin comanda de "0" logic.

Solutia prezentata este cea nemultiplexata și cere un registru pentru fiecare rang având atât avantaje cât și dezavantaje.

Avantaje: programul este simplu.

Dezavantaje: avem un număr mare de circuite, un număr mare de rezistente și prin urmare un consum mai ridicat.

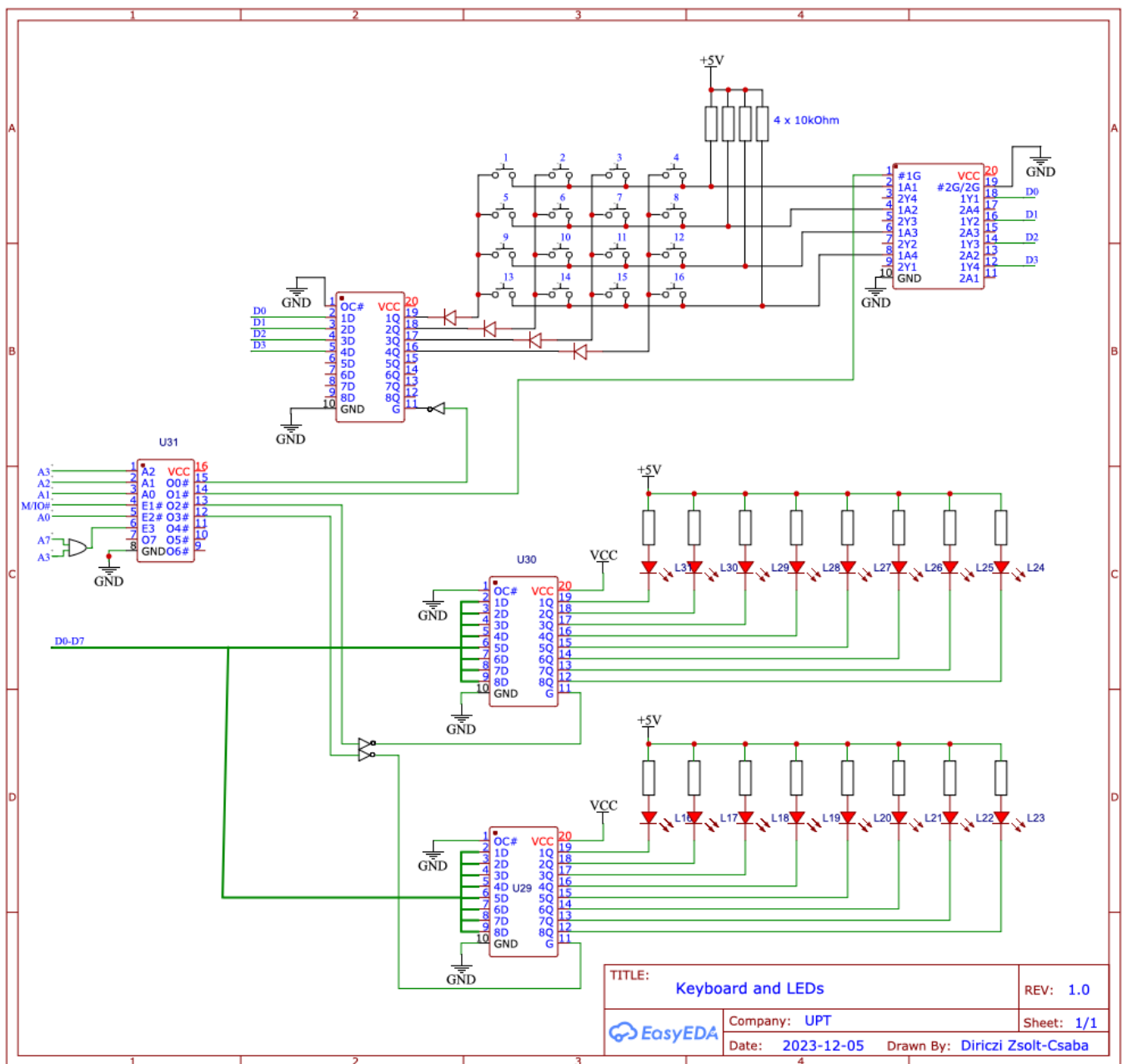
De remarcat este faptul ca ieşirile decodicatorului sunt active pe "0", în timp ce intrările de selecție a registrelor (CS#) sunt active pe "1", prin urmare înainte de a conecta ieşirile din decodicator la registre le-am negat.

Am folosit registre 74LS373 (cu 8 ranguri), având ca scop menținerea valorii afişajului pana la încărcarea unei alte valori.

Pentru a dimensiona rezistenta ataşată pe intrarea de date a afişajelor, am considerat ca fiecare LED (în total 8) necesita un curent de 10mA, cu o cădere de tensiune 1.6V, deci pentru un singur LED avem nevoie de $R = (5 - 1.6 - 0.2)V / 10mA = 320\Omega$ iar cea mai apropiata valoare uzuala este $R = 330\Omega$. Astfel pe fiecare intrare de date D0-D7 se va ataşa o rezistenta de 330Ω.

Aşadar pentru a afişa o cifra pe un astfel de display trebuie sa plasam 0 logic la intrările de LED care trebuie sa fie aprinse pentru a obtine cifra dorită. Aceste configurații binare pentru fiecare cifra hexazecimala vor fi prezentate în secțiunea de programe aferenta afişajelor.

2.4.3.2 Minitastatura si LED-uri



Pentru minitastatura cu cele 16 contacte și 16 LED-uri am folosit un decodicator separat.

2.4.3.2.1 LED-uri

Pentru a putea ataşa 16 LED-uri avem nevoie de 2 registre cu 8 ranguri. Astfel am folosit registre LS373 fiecare dintre cele 2 fiind activat de o ieşire din decodificator (O2# şi O3#). Astfel la intrările de enable(G) pentru registrii am legat iesiri din decodificator.

La fel ca și în cazul afișajelor cu segmente am ales configurația cu anod comun, prin urmare pentru a aprinde un led trebuie să furnizăm la intrare valoarea de “0” logic.

Important de mențion este necesitatea rezistențelor la anodul LED-urilor pentru a limita curentul ce trece prin dioda la 10mA. Soluția corectă este plasarea unei rezistențe la anodul fiecărui LED deoarece în cazul în care s-ar folosi o singură rezistență pentru toate cele 8 LED-uri, nu am putea dimensiona rezistența în mod corect, deoarece ar depinde de numărul de LED-uri ce sunt aprinse. Pentru a dimensiona rezistența atașată pe intrarea de date a LED-urilor, am considerat ca fiecare

LED necesita un curent de 10mA, cu o cădere de tensiune 1.6V, deci pentru un singur LED avem nevoie de $R = (5 - 1.6 - 0.2)V / 10mA = 320\Omega$ iar cea mai apropiată valoare uzuală este $R = 330\Omega$. Astfel la fiecare anod al LED-ului plasăm o rezistență de 330Ω .

Pentru a aprinde anumite LED-uri trebuie să furnizăm o combinație binară pe D0-D7 astfel încât la valoarea "0" a unui bit LED-ul corespunzător acestuia va fi aprins în schimb la valoarea "1" logic va fi stins.

2.4.3.2.1 Minitastatura

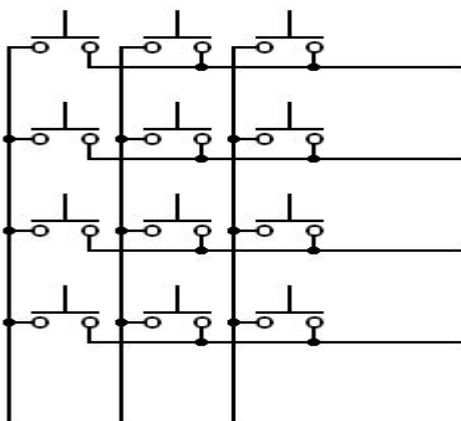
Conectarea minitastaturii are o structură matricială, la intersecția liniilor și a coloanelor găsim-se tastele.

Este necesar un port de ieșire cu posibilitate de memorare(registru) și un port de intrare(porti cu 3 stări). Astfel pentru portul de intrare am folosit circuitul LS373 iar pentru portul de ieșire am folosit circuitul LS244.

Datorită faptului că avem nevoie de cele 2 circuite prezentate mai sus, acestea vor fi activate de 2 ieșiri ale decodificatorului (K1 și K2) care se leagă la enable-ul acestora (G respectiv 1G#).

Pentru a scana tastatura, se baleiază coloanele pe care se plasează "0" logic și se citesc liniile. Astfel dacă butonul nu este apăsat la ieșire avem "1" logic fiind legat la 5V, altfel dacă este apăsat avem "0" logic.

Pentru a proteja ieșirile portului de ieșire (LS373) se conectează diode, însă acestea trebuie alese atent deoarece ridică nivelul de tensiune pentru "0" logic. Din acest motiv, pentru ca "0" logic să aibă valoare minimă, se recomandă folosirea diodelor cu germaniu în loc de cele de siliciu.



3. Programe

3.1 Rutina de afișare a unui caracter hexa pe un rang cu segmente

```
1 ; consideram ca avem valoarea de afisat in registrul AL
2 ; consideram ca avem portul de afisare in registrul AH
3 ; deci in AH putem avea 08H, 18H, 28H, 38H, 48H, 58H, 68H, 78H
4 afisare_hexa proc
5
6     D_0    CMP AL, 0 ; daca valoarea este 0 ramanem sa afisam 0
7           JNE D_1 ; altfel trecem la urmatoarea cifra
8           MOV AL, 0C0H ; C0H este codul pentru 0
9           OUT AH, AL ; afisam 0 la portul indicat de AH
10          JMP END ; iesim din procedura deoarece am afisat cifra
11
12     D_1    CMP AL, 1 ; daca valoarea este 1 ramanem sa afisam 1
13           JNE D_2 ; altfel trecem la urmatoarea cifra
14           MOV AL, 0F9H ; F9H este codul pentru 1
15           OUT AH, AL ; afisam 1 la portul indicat de AH
16           JMP END ; iesim din procedura deoarece am afisat cifra
17
18     D_2    CMP AL, 2 ; daca valoarea este 2 ramanem sa afisam 2
19           JNE D_3 ; altfel trecem la urmatoarea cifra
20           MOV AL, 0A4H ; A4H este codul pentru 2
21           OUT AH, AL ; afisam 2 la portul indicat de AH
22           JMP END ; iesim din procedura deoarece am afisat cifra
23
24     D_3    CMP AL, 3 ; daca valoarea este 3 ramanem sa afisam 3
25           JNE D_4 ; altfel trecem la urmatoarea cifra
26           MOV AL, 0B0H ; B0H este codul pentru 3
27           OUT AH, AL ; afisam 3 la portul indicat de AH
28           JMP END ; iesim din procedura deoarece am afisat cifra
29
30     D_4    CMP AL, 4 ; daca valoarea este 4 ramanem sa afisam 4
31           JNE D_5 ; altfel trecem la urmatoarea cifra
32           MOV AL, 99H ; 99H este codul pentru 4
33           OUT AH, AL ; afisam 4 la portul indicat de AH
34           JMP END ; iesim din procedura deoarece am afisat cifra
35
36     D_5    CMP AL, 5 ; daca valoarea este 5 ramanem sa afisam 5
37           JNE D_6 ; altfel trecem la urmatoarea cifra
38           MOV AL, 92H ; 92H este codul pentru 5
39           OUT AH, AL ; afisam 5 la portul indicat de AH
40           JMP END ; iesim din procedura deoarece am afisat cifra
41
42     D_6    CMP AL, 6 ; daca valoarea este 6 ramanem sa afisam 6
43           JNE D_7 ; altfel trecem la urmatoarea cifra
44           MOV AL, 82H ; 82H este codul pentru 6
45           OUT AH, AL ; afisam 6 la portul indicat de AH
46           JMP END ; iesim din procedura deoarece am afisat cifra
47
48     D_7    CMP AL, 7 ; daca valoarea este 7 ramanem sa afisam 7
49           JNE D_8 ; altfel trecem la urmatoarea cifra
50           MOV AL, 0F8H ; F8H este codul pentru 7
51           OUT AH, AL ; afisam 7 la portul indicat de AH
52           JMP END ; iesim din procedura deoarece am afisat cifra
```

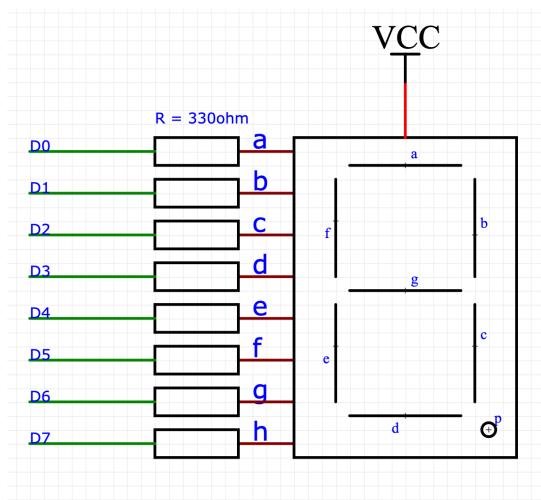
```
1     D_8    CMP AL, 8 ; daca valoarea este 8 ramanem sa afisam 8
2           JNE D_9 ; altfel trecem la urmatoarea cifra
3           MOV AL, 80H ; 80H este codul pentru 8
4           OUT AH, AL ; afisam 8 la portul indicat de AH
5           JMP END ; iesim din procedura deoarece am afisat cifra
6
7     D_9    CMP AL, 9 ; daca valoarea este 9 ramanem sa afisam 9
8           JNE D_A ; altfel trecem la urmatoarea cifra
9           MOV AL, 90H ; 90H este codul pentru 9
10          OUT AH, AL ; afisam 9 la portul indicat de AH
11          JMP END ; iesim din procedura deoarece am afisat cifra
12
13     D_A    CMP AL, 10 ; daca valoarea este 10 ramanem sa afisam A
14           JNE D_B ; altfel trecem la urmatoarea cifra
15           MOV AL, 88H ; 88H este codul pentru A
16           OUT AH, AL ; afisam A la portul indicat de AH
17           JMP END ; iesim din procedura deoarece am afisat cifra
18
19     D_B    CMP AL, 11 ; daca valoarea este 11 ramanem sa afisam B
20           JNE D_C ; altfel trecem la urmatoarea cifra
21           MOV AL, 83H ; 83H este codul pentru B
22           OUT AH, AL ; afisam B la portul indicat de AH
23           JMP END ; iesim din procedura deoarece am afisat cifra
24
25     D_C    CMP AL, 12 ; daca valoarea este 12 ramanem sa afisam C
26           JNE D_D ; altfel trecem la urmatoarea cifra
27           MOV AL, 0C6H ; C6H este codul pentru C
28           OUT AH, AL ; afisam C la portul indicat de AH
29           JMP END ; iesim din procedura deoarece am afisat cifra
30
31     D_D    CMP AL, 13 ; daca valoarea este 13 ramanem sa afisam D
32           JNE D_E ; altfel trecem la urmatoarea cifra
33           MOV AL, 0A1H ; A1H este codul pentru D
34           OUT AH, AL ; afisam D la portul indicat de AH
35           JMP END ; iesim din procedura deoarece am afisat cifra
36
37     D_E    CMP AL, 14 ; daca valoarea este 14 ramanem sa afisam E
38           JNE D_F ; altfel trecem la urmatoarea cifra
39           MOV AL, 86H ; 86H este codul pentru E
40           OUT AH, AL ; afisam E la portul indicat de AH
41           JMP END ; iesim din procedura deoarece am afisat cifra
42
43     D_F    CMP AL, 15 ; daca valoarea este 15 ramanem sa afisam F
44           JNE END ; altfel trecem la urmatoarea cifra
45           MOV AL, 8EH ; 8EH este codul pentru F
46           OUT AH, AL ; afisam F la portul indicat de AH
47           JMP END ; iesim din procedura deoarece am afisat cifra
48
49     END ret
50
51 afisare_hexa endp
```

Subprogramul de mai sus reprezintă o rutină destinată afișării unei cifre hexazecimale pe unul dintre afișajele cu segmente. La începutul procedurii, considerăm că în registrul AL avem cifra ce urmează să fie afișată, iar în registrul AH avem adresa A7-A0 pentru a selecta la care dintre afișaje să fie afisata cifra.

Procesul începe prin compararea valorii de afișat cu zero. În situația în care valoarea de afișat este zero, se încarcă în registrul AL valoarea C0H. Ulterior, instrucțiunea OUT este utilizată pentru a trimite la portul specificat în registrul AH valoarea din registrul AL, reprezentând combinația de biți necesară pentru a obține pe afișaj cifra 0. După aceasta, se realizează un salt către eticheta END, urmat de instrucțiunea de return.

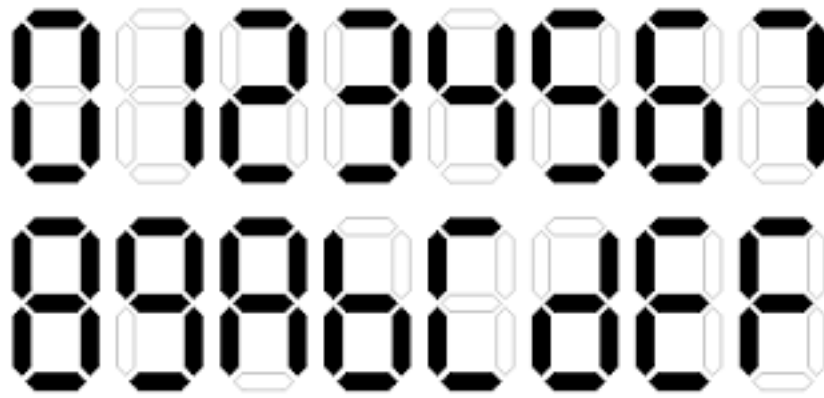
În cazul în care valoarea de afișat nu este zero, se efectuează un salt la codul destinat următoarei cifre și se reia același proces precum în cazul cifrei 0, până când se ajunge la codul pentru valoarea care trebuie afișată.

Pentru determinarea configurației binare pentru afișarea fiecărei cifre hexazecimale am considerat ca biții de date sunt legate la afișaje astfel:



Deoarece am folosit afișaje cu catod comun, pentru a aprinde un led trebuie ca la intrare sa furnizam "0" logic iar în cazul în care pe intrare avem "1" logic acel led va rămâne stins. Astfel am construit un tabel care prezintă configurația binară pentru fiecare cifra hexazecimale.

Cifra	D7 - p	D6 - g	D5 - f	D4 - e	D3 - d	D2 - c	D1 - b	D0 - a	Hex
0	1	1	0	0	0	0	0	0	C0H
1	1	1	1	1	1	0	0	1	F9H
2	1	0	1	0	0	1	0	0	A4H
3	1	0	1	1	0	0	0	0	B0H
4	1	0	0	1	1	0	0	1	99H
5	1	0	0	1	0	0	1	0	92H
6	1	0	0	0	0	0	1	0	82H
7	1	1	1	1	1	0	0	0	F8H
8	1	0	0	0	0	0	0	0	80H
9	1	0	0	1	0	0	0	0	90H
A	1	0	0	0	1	0	0	0	88H
B	1	0	0	0	0	0	1	1	83H
C	1	1	0	0	0	1	1	0	C6H
D	1	0	1	0	0	0	0	1	A1H
E	1	0	0	0	0	1	1	0	86H
F	1	0	0	0	1	1	1	0	8EH



3.2 Rutina pentru initializare interfața seriala

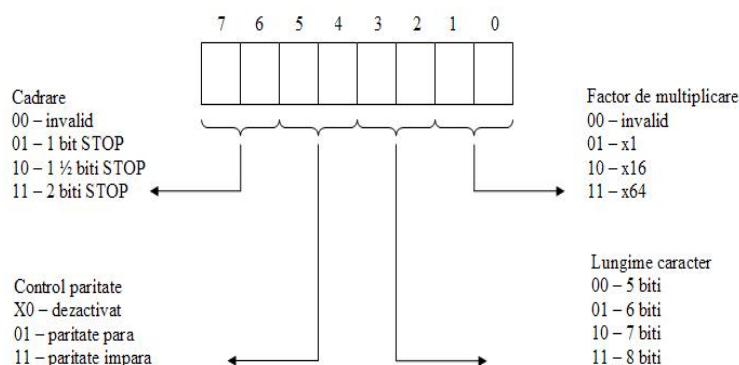
```
1  init_serial_interface proc
2      ; 8 biti de date
3      ; fara paritate
4      ; 2 biti de stop
5      ; factor de multiplicare 16
6      ; rata de transfer 9600 bps
7      MOV AL, 0CEH ; cuvant de mod
8      OUT 0DD2H, AL
9      ; sincronizare: operare normala
10     ; reinitializare: operare normala
11     ; comanda RTS#: RTS# = 1
12     ; comanda indicatori eroare: anulare
13     ; break: operare normala
14     ; comanda receptie: activare
15     ; comanda DTR#: DTR# = 1
16     ; comanda transmitere: activare
17     MOV AL, 015H ; cuvant de comanda
18     OUT 0DD2H, AL
19  init_serial_interface endp
```

Pentru sincronizarea între emițător și receptor este necesar să se stabilească caracteristicile transferului de caractere. Astfel eu am ales:

- 2 biti de stop
- fara control de paritate
- 8 biti de date (pentru a putea trimite un caracter ASCII)
- factor de multiplicare 16

Cadrare		Control paritate		Lungime caracter		Factor de multiplicare	
1	1	0	0	1	1	1	0
2 biti de stop		Dezactivat		8 biti		X16	

Prin urmare cuvântul de mod este 11001111b = CEh.

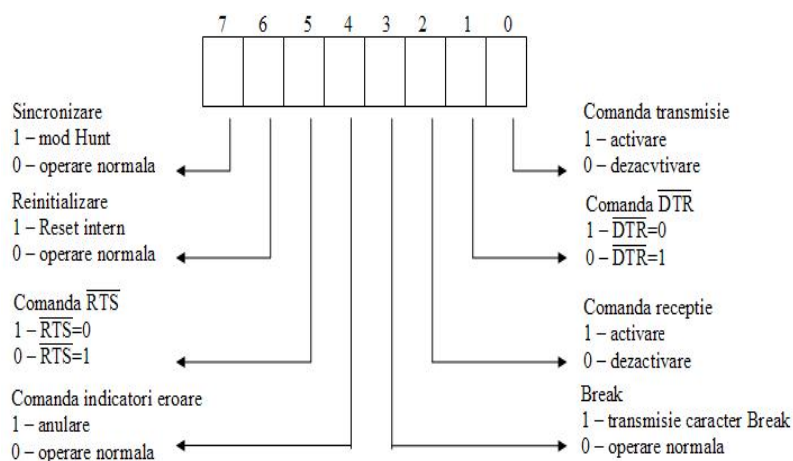


Pentru inițializarea circuitului 8051 am ales:

- sincronizare în operarea normala
- reinitializare in operare normala
- RTS# = 1
- comanda indicatori de eroare anulată
- Break in operare normala
- comanda recepție activa
- DTR# = 1
- comanda transmisie activ

Sincronizare	Reinitializare	RTS#	Flag eroare	Break	Recepție	DTR#	Transmisie
0	0	0	1	0	1	0	1
op. normala	op. normala	RTS# = 1	anulare	op. Normala	activare	DTR# = 1	activare

Prin urmare cuvântul de comanda este 00010101b = 15h.

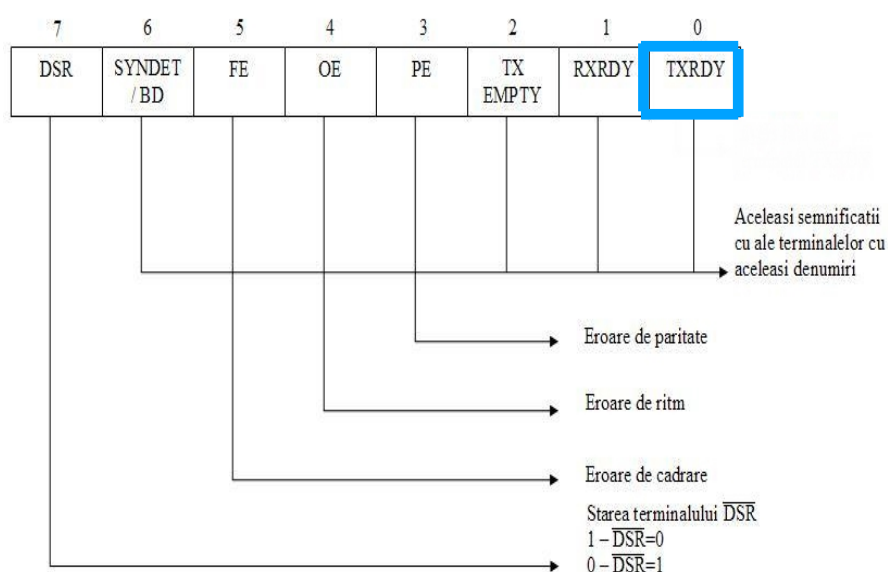


3.3 Rutina de emisie caracter pe interfața seriala

```
1 ; rutina de transmitere a unui caracter
2 transmit_character proc
3 TR:    IN AL, 0DD2H ; citire stare
4        RCR AL, 1    ; testare TxRDY
5        JNC TR        ; daca TxRDY = 0, repetare
6        MOV AL, CL    ; incarcare caracter
7        OUT 0DD0H, AL ; transmitere caracter
8        RET
9 transmit_character endp
```

Pentru a emite un caracter, se citește starea circuitului 8251 în registrul AL, de la portul cu A2 = 1. Deoarece valoarea lui TxRDY este reprezentată de cel mai puțin semnificativ bit din AL, prin efectuarea unei rotații la dreapta cu carry, în flag-ul de carry se setează valoarea lui TxRDY. Datorită faptului că TxRDY este activ pe nivel logic superior, în cazul în care acesta are valoarea 0, se face un salt la început și se verifică din nou, iterând prin această buclă până când TxRDY devine 1.

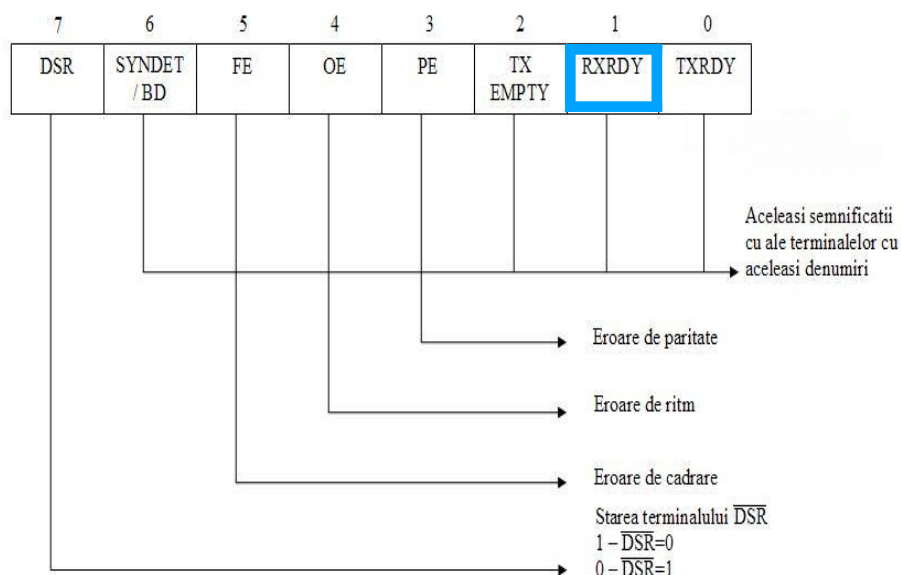
Odată ce TxRDY devine 1, considerând că în registrul CL avem caracterul ce trebuie transmis, încărcăm valoarea acestuia la intrarea de date a interfeței seriale, care o poate încărca în mod paralel în buffer-ul de date. Prin intermediul semnalului TxRDY, circuitul 8251 anunță procesorul că a terminat de transmis un caracter pe linie, l-a serializat și îi solicită un alt caracter.



3.4 Rutina de recepție caracter pe interfața serială

```
1 ; rutina de receptie a unui caracter
2 receive_character proc
3
4 RC:    IN AL, 0DD2H ; citire stare
5        RCR AL, 2    ; testare RxRDY
6        JNC RC        ; daca RxRDY = 0, repetare
7        IN AL, 0DD0H ; citire caracter
8        MOV CL, AL    ; salvare caracter
9        RET
10 receive_character endp
```

La fel ca în cazul emisieii unui caracter, începem prin citirea stării circuitului 8251 în registrul AL. Conform schemei de mai jos, valoarea lui RxRDY se află pe a doua poziție din AL. Astfel, efectuând o rotație cu carry la dreapta pe AL, obținem valoarea lui RxRDY în flag-ul de carry. Dacă aceasta este 0, facem un salt la început și reluăm verificarea stării, iterând prin această buclă până când RxRDY devine 1. Prin RxRDY, circuitul 8251 anunță procesorul că a terminat de preluat un caracter pe linia serială, l-a asamblat și este pregătit să-l predea.



3.5 Rutina de programare pentru timer

```

1  init_timer proc
2      ; Selectie contor: contor 1
3      ; Comanda: incarcare octet mai putin semnificativ
4      ; Mod: mod 3
5      ; Mod binar/BCD: binar
6      MOV AL, 056H ;cuvant comanda
7      OUT 0C6H, AL; OUT la RCC (A2A1 = 11)
8      ; constanta 2457600 / 153600 = 16 pentru contor 0
9      MOV AL, 010H; constanta 16
10     OUT 0C2H, AL; OUT la timer 1 (A2A1 = 01)
11     RET
12 init_timer endp

```

Pentru a folosi circuitul 8253, trebuie sa inițializăm contorul pe care dorim sa îl folosim. Am ales următoarele opțiuni de inițializare:

- Contor 1
- Incarcare octet mai puțin semnificativ
- Mod de lucru 3
- Mod binar

Selectie contor		Comanda		Mod de lucru			Mod
0	1	0	1	0	1	1	0
Contor 1		Incarcare octet inferior		Mod 3			Binar

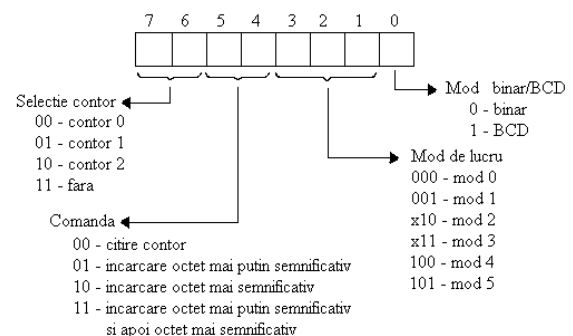
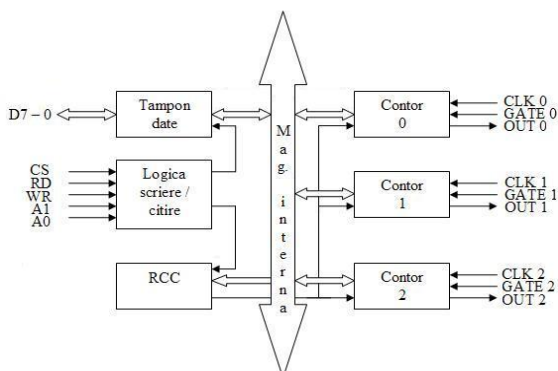
Prin urmare, cuvântul de comanda este 01010110b = 56h

În modul 3, ieșirea contorului va genera un semnal dreptunghiular cu perioada egală cu perioada tactului înmulțită cu valoarea constantei.

Deoarece am ales un baudRate de 9600bps și factorul de multiplicare este x16, contorul trebuie să genereze un semnal de frecvență 1536000Hz.

Astfel constanta pentru contorul 1 este egală cu $2457600/153600 = 16$

Cu A2, A1 selectăm una din cele 4 porturi interne: 00 Contor 0, 01 Contor 1, 10 Contor 2, 11 pentru RCC. Astfel cuvântul de comanda este transmis la RCC iar constanta 16 este încărcată în Contor 1.



3.6 Rutina de programare pentru interfața paralela

```

1  init_paralel_interface proc
2  ; mod 0 iesire pt A
3  ; mod 0 iesire pt B
4  ; mod 0 intrare pt C inferior
5  ; mod 0 iesire pt C superior
6      MOV AL, 81H ; cuvant de comanda
7      OUT 0156H, AL ; scriere in registrul de comanda/control RCC
8
9      RET
10 init_paralel_interface endp

```

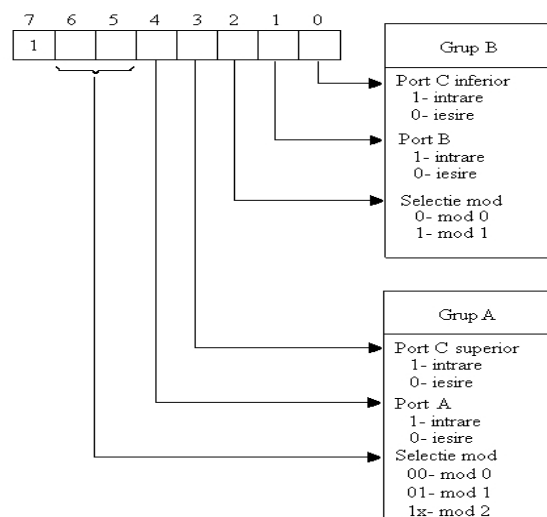
Pentru a folosi circuitul 8255, trebuie mai întâi inițializat.

Pentru inițializare am ales:

- Selectie mod grup A: mod 0
- Port A: ieșire
- Port C superior: ieșire
- Selectie mod grup B: mod 0
- Port B: ieșire
- Port C inferior: intrare

Rezervat	Grup A				Grup B		
	Selectie mod		Port A	Port C sup.	Sel. mod	Port B	Port C inf.
1	0	0	0	0	0	0	1
Rezervat	Mod 0		iesire	iesire	Mod 0	iesire	Intrare

Prin urmare cuvântul de comanda este 10000001b = 81h.

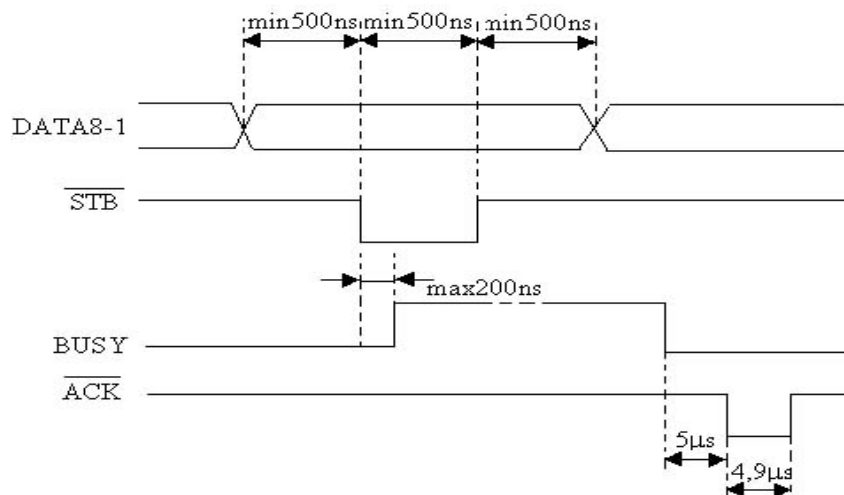


3.7 Rutina de emisie caracter pe interfața paralela

```
1  transmit_character_paralel proc
2  PAR : IN AL, 0154H ;citire si testare BUSY
3      RCR AL, 1
4      JC PAR
5      MOV AL, CL ; incarcare caracter
6      OUT 150H, AL ; transmitere caracter
7      OR AL, 01H ; setare bit STROBE
8      OUT 152H, AL ; transmitere STROBE
9      AND AL, 00H ; resetare bit STROBE
10     OUT 152H, AL ; transmitere STROBE
11     OR AL, 01H ; setare bit STROBE
12     OUT 152H, AL ; transmitere STROBE
13     RET
14  transmit_character_paralel endp
```

Rutina de emisie a unui caracter prin interfața serială este descrisă mai sus. Procesul începe prin citirea liniei BUSY de la portul C. Pentru a testa valoarea BUSY, știind că se află pe cel mai puțin semnificativ bit, efectuăm o rotație cu carry la dreapta, obținând astfel valoarea BUSY în flag-ul de carry. Dacă BUSY este 1, se face un salt la început pentru a relua aceiași pași, deoarece trebuie să așteptăm ca circuitul receptor să fie disponibil pentru a putea recepționa date. În cazul în care BUSY are valoarea "0" logic, continuăm prin a copia valoarea din registrul CL (în care inițial avem octetul de transmis) în AL și apoi o trimitem la portul de ieșire A.

Conform protocolului CENTRONIX, în momentul în care se transmit date, circuitul transmitător trebuie să activeze ieșirea STB#. Astfel, inițial o setăm pe 1 și o trimitem la portul de ieșire B, apoi pe 0 și în final o setăm din nou pe 1.



3.8 Rutina de scanare minitastatura

```
1 scan_keyboard proc
2
3     ; setam 0 pe prima coloana si verificam daca a fost apasat butonul 1,5,9,13
4     REIA: MOV AL, FEH; punem 0 pe prima coloana
5     OUT 88H, AL; trimitem pe portul 88H
6     IN AL, 8AH; citim de pe portul 8AH in AL
7     AND AL, 01H; verificam daca a fost apasat butonul 1
8     JZ TASTA_1; daca a fost apasat butonul 1, sarim la eticheta TASTA_1
9     IN AL, 8AH; citim de pe portul 8AH in AL
10    AND AL, 02H; verificam daca a fost apasat butonul 5
11    JZ TASTA_5; daca a fost apasat butonul 5, sarim la eticheta TASTA_5
12    IN AL, 8AH; citim de pe portul 8AH in AL
13    AND AL, 02H; verificam daca a fost apasat butonul 9
14    JZ TASTA_9; daca a fost apasat butonul 9, sarim la eticheta TASTA_9
15    IN AL, 8AH; citim de pe portul 8AH in AL
16    AND AL, 08H; verificam daca a fost apasat butonul 13
17    JZ TASTA_13; daca a fost apasat butonul 13, sarim la eticheta TASTA_13
18
19    ; setam 0 pe a doua coloana si verificam daca a fost apasat butonul 2,6,10,14
20    REIA: MOV AL, FDH; punem 0 pe a doua coloana
21    OUT 88H, AL; trimitem pe portul 88H
22    IN AL, 8AH; citim de pe portul 8AH in AL
23    AND AL, 01H; verificam daca a fost apasat butonul 2
24    JZ TASTA_2; daca a fost apasat butonul 2, sarim la eticheta TASTA_2
25    IN AL, 8AH; citim de pe portul 8AH in AL
26    AND AL, 02H; verificam daca a fost apasat butonul 6
27    JZ TASTA_6; daca a fost apasat butonul 6, sarim la eticheta TASTA_6
28    IN AL, 8AH; citim de pe portul 8AH in AL
29    AND AL, 02H; verificam daca a fost apasat butonul 10
30    JZ TASTA_10; daca a fost apasat butonul 10, sarim la eticheta TASTA_10
31    IN AL, 8AH; citim de pe portul 8AH in AL
32    AND AL, 08H; verificam daca a fost apasat butonul 14
33    JZ TASTA_14; daca a fost apasat butonul 14, sarim la eticheta TASTA_14
34
35    ; setam 0 pe a treia coloana si verificam daca a fost apasat butonul 3,7,11,15
36    MOV AL, FBH; punem 0 pe a treia coloana
37    OUT 88H, AL; trimitem pe portul 88H
38    IN AL, 8AH; citim de pe portul 8AH in AL
39    AND AL, 01H; verificam daca a fost apasat butonul 3
40    JZ TASTA_3; daca a fost apasat butonul 3, sarim la eticheta TASTA_3
```

```
1     IN AL, 8AH; citim de pe portul 8AH in AL
2     AND AL, 02H; verificam daca a fost apasat butonul 7
3     JZ TASTA_7; daca a fost apasat butonul 7, sarim la eticheta TASTA_7
4     IN AL, 8AH; citim de pe portul 8AH in AL
5     AND AL, 02H; verificam daca a fost apasat butonul 11
6     JZ TASTA_11; daca a fost apasat butonul 11, sarim la eticheta TASTA_11
7     IN AL, 8AH; citim de pe portul 8AH in AL
8     AND AL, 08H; verificam daca a fost apasat butonul 15
9     JZ TASTA_15; daca a fost apasat butonul 15, sarim la eticheta TASTA_15
10
11    ; setam 0 pe a patra coloana si verificam daca a fost apasat butonul 4,8,12,16
12    REIA: MOV AL, FEH; punem 0 pe a patra coloana
13    OUT 88H, AL; trimitem pe portul 88H
14    IN AL, 8AH; citim de pe portul 8AH in AL
15    AND AL, 01H; verificam daca a fost apasat butonul 4
16    JZ TASTA_4; daca a fost apasat butonul 4, sarim la eticheta TASTA_4
17    IN AL, 8AH; citim de pe portul 8AH in AL
18    AND AL, 02H; verificam daca a fost apasat butonul 8
19    JZ TASTA_8; daca a fost apasat butonul 8, sarim la eticheta TASTA_8
20    IN AL, 8AH; citim de pe portul 8AH in AL
21    AND AL, 02H; verificam daca a fost apasat butonul 12
22    JZ TASTA_12; daca a fost apasat butonul 12, sarim la eticheta TASTA_12
23    IN AL, 8AH; citim de pe portul 8AH in AL
24    AND AL, 08H; verificam daca a fost apasat butonul 16
25    JZ TASTA_16; daca a fost apasat butonul 16, sarim la eticheta TASTA_16
26
27    JMP REIA; reluam baleierea
28
29    ;tratare actionare tasta 1
30    TASTA1: CALL DELAY; asteptam stabilizarea
31    AST1: IN AL, 8AH; asteptam dezactivarea tastei
32    AND AL, 01H; verificam daca inca e apasat butonul 1
33    JZ AST1; daca inca e apasat butonul 1, sarim la eticheta AST1
34    CALL DELAY; asteptam stabilizarea
35    ;operatie pentru tasta 1
36
37    JMP REIA; reluam scanarea tastaturii
38
39    RET
40
41 scan_keyboard endp
```

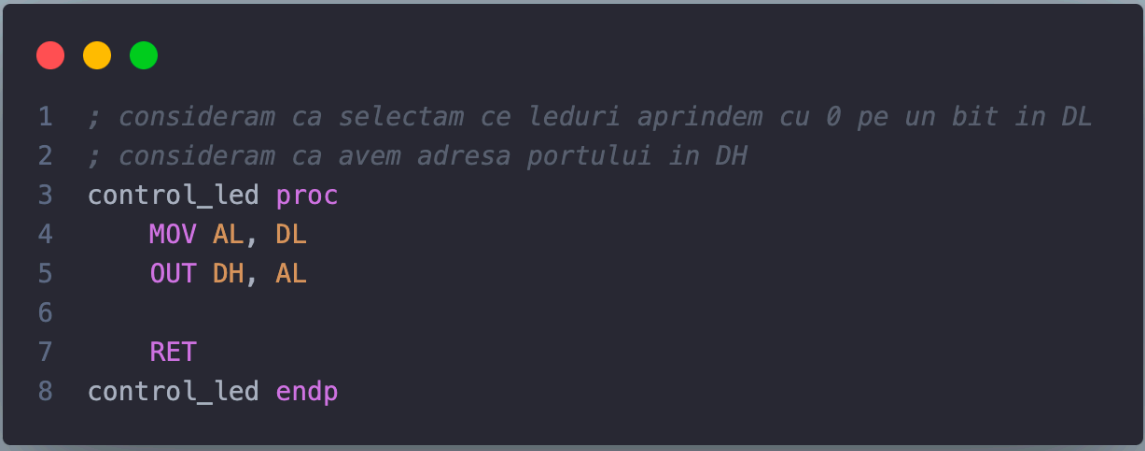
Conectarea minitastaturii se realizează printr-o structură matricială, iar tastele sunt situate la intersecția liniilor și coloanelor. O metodă simplă pentru a verifica dacă o tastă este apăsată este să setăm pe o coloană valoarea "0" logic și să verificăm apoi valoarea logică de la ieșirile corespunzătoare tastelor de pe coloana respectivă. Dacă nivelul logic de la o ieșire este "0", înseamnă că tasta este apăsată, altfel nu.

Astfel, programul începe prin setarea valorii 0 pe prima coloană și verificarea fiecărei ieșiri. Operația logică AND este utilizată pentru a determina nivelul logic al uneia dintre cele 4 taste de pe coloana curentă. Se efectuează o operație AND logic cu 01H pentru prima tastă de pe coloană, 02H pentru a doua tastă și așa mai departe. În cazul în care la o ieșire avem un nivel logic inferior, înseamnă că tasta a fost apăsată și se face un salt la tratarea apăsării acelei taste.

În cadrul tratării apăsării tastei, se observă că inițial apelăm o procedură DELAY, care introduce o întârziere necesară pentru stabilizarea nivelului logic în urma apăsării. Apoi, verificăm dacă tasta este încă apăsată. Dacă este cazul, se face un salt la instrucțiunea de citire a stării și iterăm în această buclă până când tasta este dezactivată. După dezactivarea tastei, mai introducem o întârziere pentru stabilizarea nivelului logic și apoi efectuăm acțiunea stabilită pentru tasta respectivă. În final, se face un salt la începutul procedurii de scanare pentru a repeta același proces.

De remarcat este că la finalul scanării tuturor tastelor trebuie să avem și un salt necondiționat la începutul procedurii, deoarece este posibil ca nicio tastă să nu fie apăsată.

3.9 Rutina de aprindere/ stingere a unui led



```
1 ; consideram ca selectam ce leduri aprindem cu 0 pe un bit in DL
2 ; consideram ca avem adresa portului in DH
3 control_led proc
4     MOV AL, DL
5     OUT DH, AL
6
7     RET
8 control_led endp
```

Deoarece LED-urile sunt folosite în configurație cu anod comun înseamnă că pentru a avea un LED aprins trebuie să plasăm pe intrarea registrului LS373 corespunzătoare acelui LED valoarea “0” logic. În cazul în care pe o intrare plasăm “1” logic, LED-ul va rămâne stins și prin urmare comanda se face la nivel logic inferior.

În programul de mai sus am mutat în registrul AL configurația binară care ne definește care LED-uri sunt aprinse, considerând că la începutul subrutinei avem în DL deja această configurație. Apoi trimitem această valoare la portul specificat. În cazul nostru avem două porturi: una care corespunde LED-urilor de rang 0-7 și una pentru cele de rang 8-15. Prin urmare DL poate lua valoarea: 8Ch sau 8Eh.

Bibliografie

1. Curs Proiectarea Microsistemelor Digitale 2023-2024
2. https://en.wikibooks.org/wiki/Serial_Programming/MAX232_Driver_Receiver
3. http://www.gabrielececchetti.it/Teaching/CalcolatoriElettronici/Docs/i8086_instruction_set.pdf
4. https://www.inf.pucrs.br/~calazans/undergrad/orgcomp_EC/mat_microproc/intel-8086_datasheet.pdf
5. <https://www.onsemi.com/pdf/datasheet/mc74lvx373-d.pdf>
6. <https://electronicsdesk.com/8284-clock-generator.html>
7. https://www.ti.com/lit/ds/symlink/sn54ls245-sp.pdf?ts=1702483098380&ref_url=https%253A%252F%252Fwww.google.com%252F
8. <https://datasheetspdf.com/pdf/489560/MacronixInternational/27C1024/1>
9. <https://pdf.dzsc.com/88889/43335.pdf>
10. https://www.ti.com/lit/ds/symlink/max232.pdf?ts=1702484456967&ref_url=https%253A%252F%252Fwww.google.com%252F
11. https://map.grauw.nl/resources/midi/intel_8251.pdf
12. <https://www.futurlec.com/80Series/8253.shtml>
13. <http://aturing.umcs.maine.edu/~meadow/courses/cos335/Intel8255A.pdf>
14. <https://www.ti.com/lit/ds/symlink/sn74s244.pdf>