UNIVERSITATEA POLITEHNICA DIN TIMIAOARA

FACULTATEA DE AUTOMATICA SI CALCULATOARE

Proiectare microsistem digital

- proiect pentru materia proiectarea microsistemelor digitate -

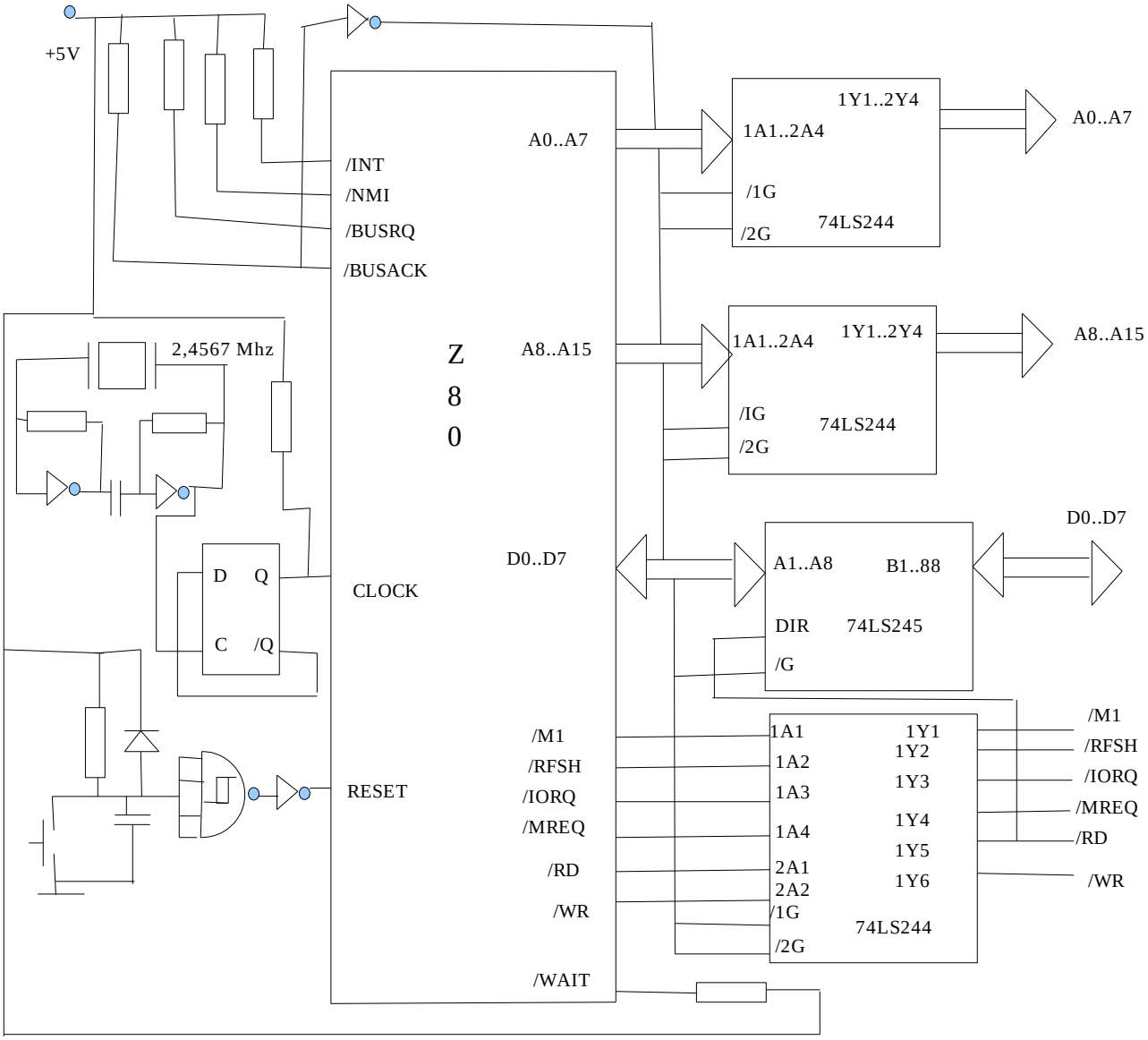
Achim Cristian
an 4 calculatoare gr 3.1
fost an 3 calculatoare gr 1.1

Enunt:

Sa se proiecteze un microsistem cu urmatoarele resurse:

- unitate centrala cu microprocesor Z80 sau microcontroler pe 8 biti ;
- 8 kiloocteti memorie SRAM si 8 kiloocteti memorie fixa ;
- interfata pentru citirea temperaturii de la 2 senzori ;
- interfata pentru afisarea temperaturii citite pe module de afisare cu segmente ;
- interfata seriala si interfata paralela ;
- temperaturile citite vor fi transmise la interfata seriala sau cea paralela la dorinta utilizatorului ;

Scheme:

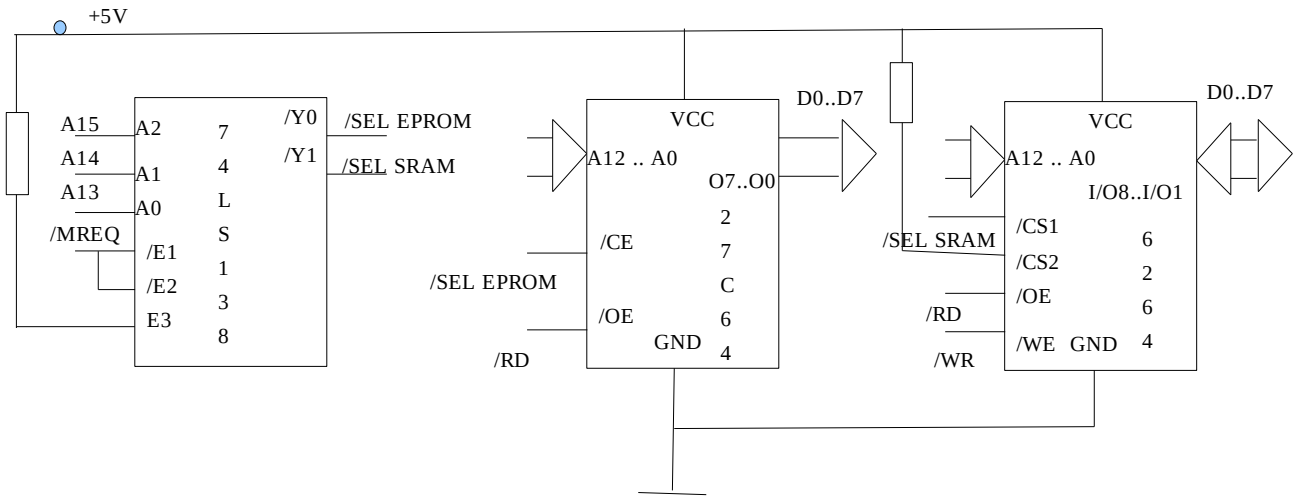


Unitate centrala cu microprocesor Z80 standard

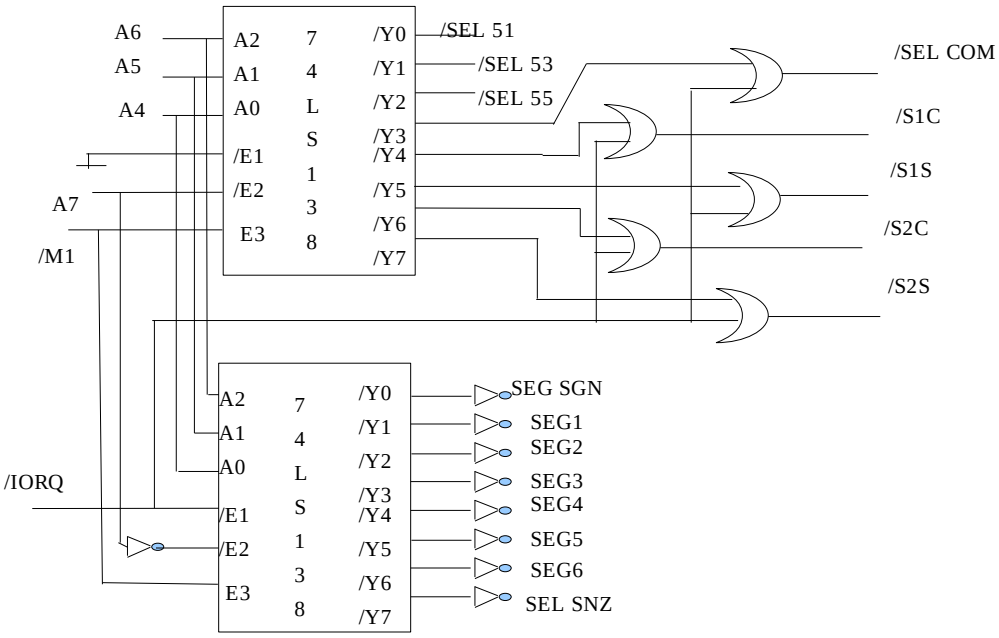
Proiectare decodificator memorii

* 8 kiloocteti EPROM 27C64 memorie fixa 0000H .. 1FFFH
* 8 kiloocteti SRAM 6264 memorie statica 2000H .. 3FFFH

decodificare completa :



Proiectare decodificator porturi si sistem intrare-iesire:

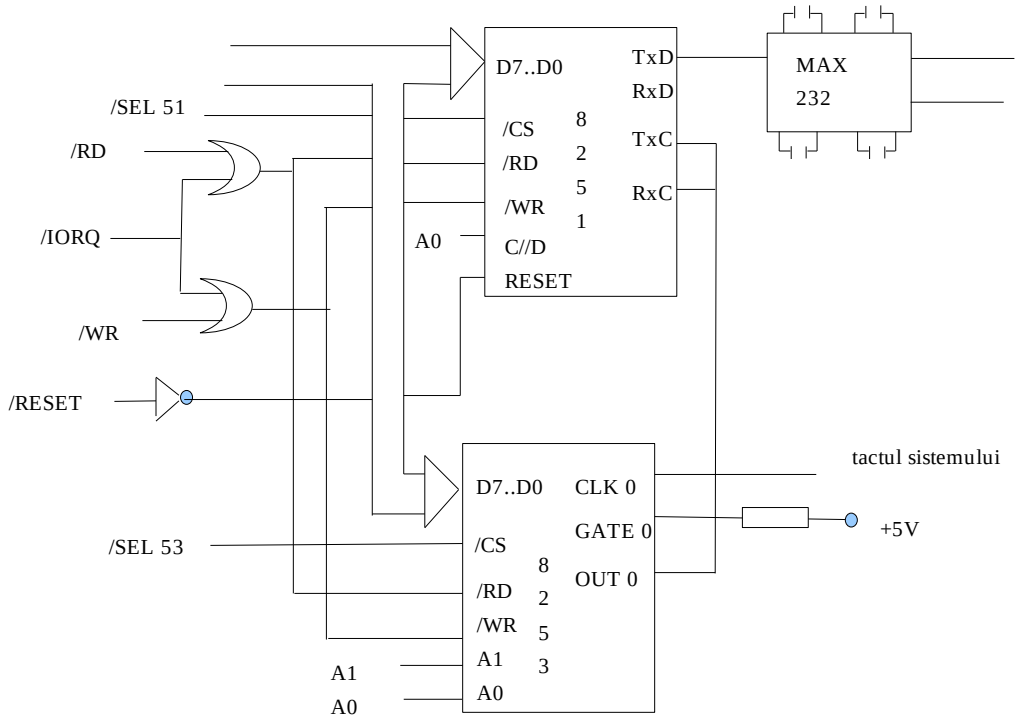


	51	53	55	com	s1c	s1s	s2c	s2s	sgn	seg1	seg2	seg3	seg4	seg5	seg6	snz
A7	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
A6	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
A5	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
A4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
hex	0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx

tabela adreselor de porturi

x pt A3 pana la A0; decodificare incompleta

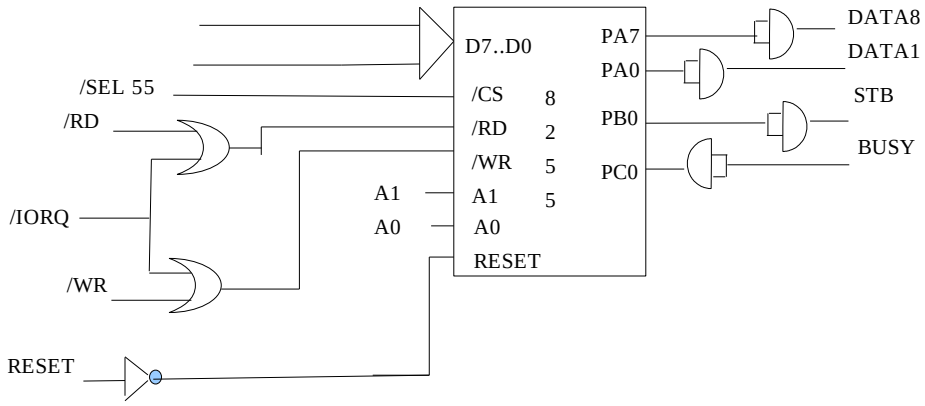
Conectarea si programarea circuitelor 8251 si 8253 (rutine din EPROM) :




```
PROGR_51 :      LD A , CEH ; cuvant mod 2 biti stop , fara paritate , 8 biti date , multiplicare x16
                OUT (00H) , A ; pt 9600 bps
                LD A , 15H ; cuvant comanda , fara reset intern , indice erori sau break
                OUT (00H) , A
                RET
EM_51 :         IN A , (01H) ; citirea starii circuitului , daca a transmis ultimul octet
                BIT 0 , A
                JP Z , EM_51
                LD A , C ; octetul de transmis
                OUT (00H) , A
                RET
```

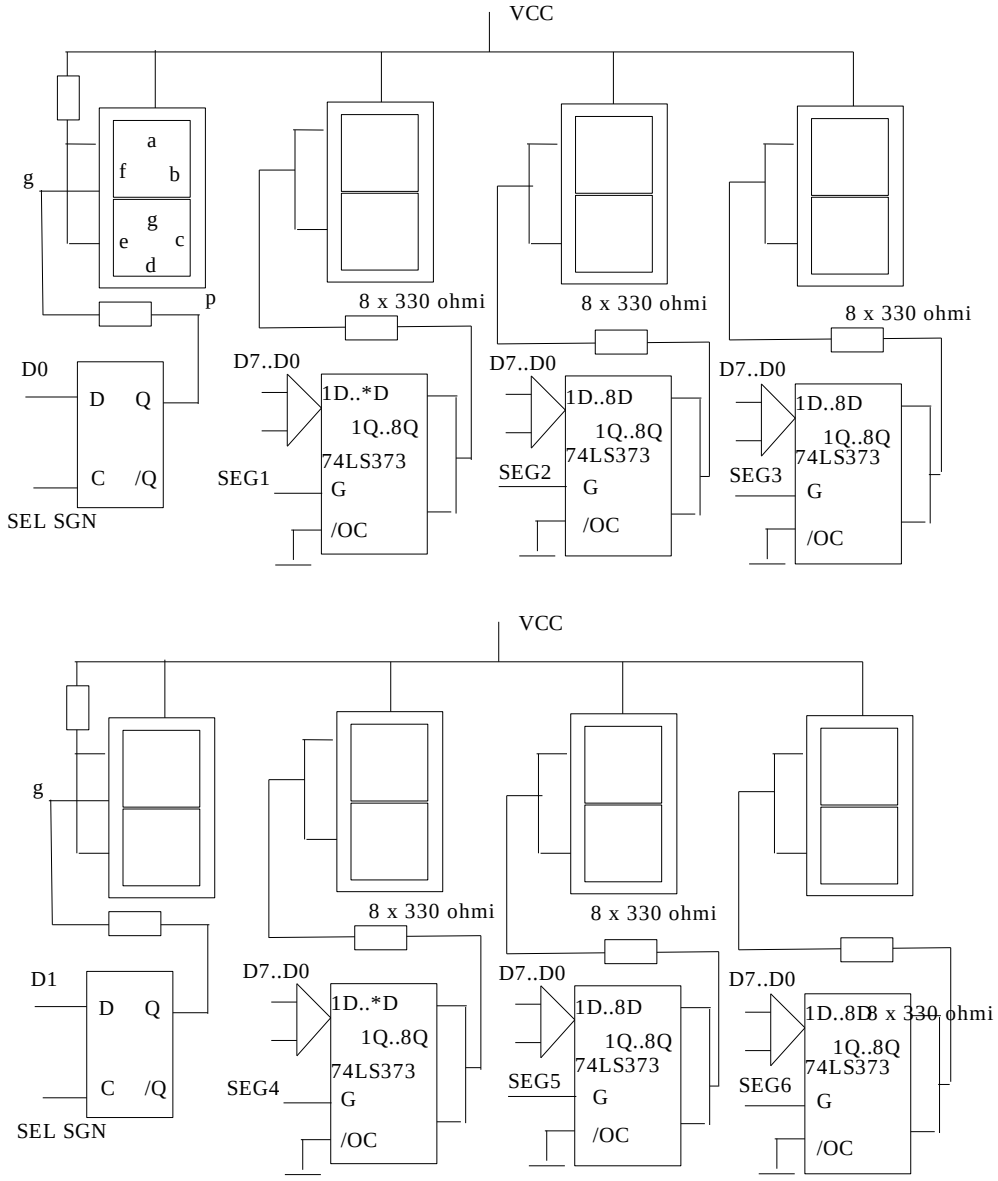
```
PROGR_53 :   LD A , 16H ; cuv com : contor 0 , incarcare octet mai putin semnificativ , genera-
             OUT (10H) , A ; tor semnale dreptunghiulare
             LD A , 10H ; 16     factor de divizare pt 9600 bps
             OUT (10H) , A
             RET
```

<u>Conectarea si programarea circuitului interfata paralela 8255 (cod in EPROM)</u> :




```
PROGR_55:    LD A , 81H ; mod 0 A si B , A iesire , B iesire , C superior iesire ,
             OUT (23H) , A ; C inferior intrare
             RET
EM_55:       IN A , (22H) ; citire + testare busy ; citire C
             BIT 0 , A ; C0 == 0 ? test busy
             JP NZ , EM_55
             LD A , C ; C     octet de transmis
             OUT (20H) , A ; la portul A
             SET 0 , A
             OUT (21H) , A ; /STB=1
             RES 0 , A
             OUT (21H) , A ; /STB=0
             SET 0 , A
             OUT (21H) , A ; /STB=1
             RET
```

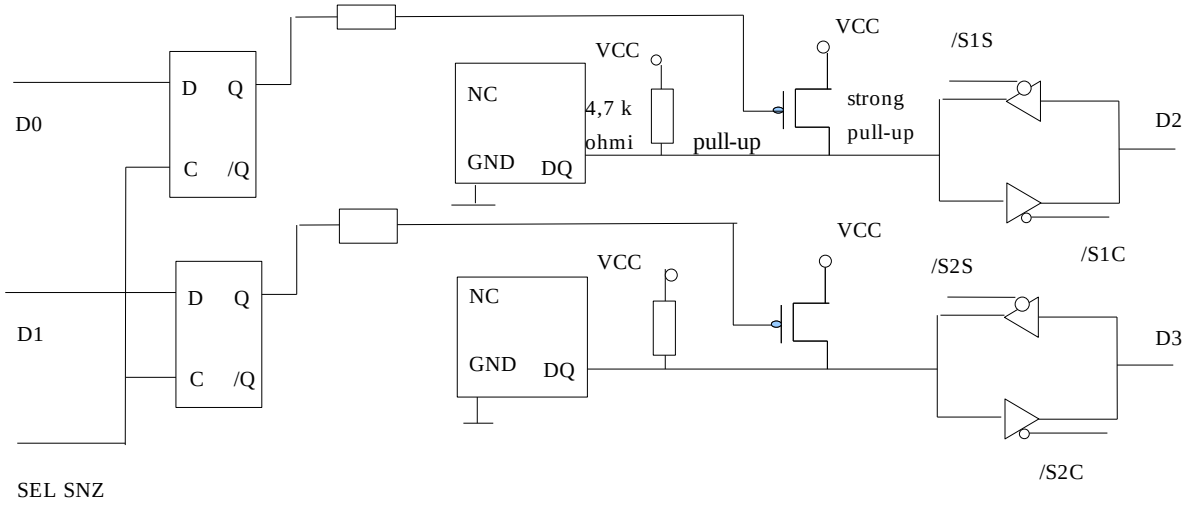
Conectare pentru modulele de afisare cu segmente si initializare (program in EPROM) :



Semnalul SEL SGN selecteaza cele 2 module care vor afisa semnul pentru cele 2 valori numerice citite de la senzori. Pentru luminarea unui LED din modul se transmite pentru LED-ul respectiv valoarea 0 logic , tensiune aproape de 0 V , si 1 logic adica ~ Vcc pentru stingerea unui LED.

```
PROGR_AFIS:     LD , 2005H , FFH ; ultima valoare scrisa la modulele de semn salvata
                LD A , FFH ; toate modulele  inactive
                OUT (80H) , A ; semnal SEL SGN activ
                OUT (90H) , A ; semnal SEG 1 activ
                OUT (A0H) , A ; semnal SEG 2 activ
                OUT (B0H) , A ; semnal SEG 3 activ
                OUT (C0H) , A ; semnal SEG 4 activ
                OUT (D0H) , A ; semnal SEG 5 activ
                OUT (E0H) , A ; semnal SEG 6 activ
                RET
```

Conectarea si programarea senzorilor de tip DS18B20 (program in EPROM) :



Cei 2 senzori se conecteaza pe linii de date separate , astfel ca la comunicare vor fi adresati fara a fi nevoie de identificare. Comunicarea se face printr-o singura linie de date , D2 sau D3 , prin intermediul unor porti cu 3 stari , cu ajutorul a 4 adrese de port , cate 2 pentru fiecare senzor pentru citire si scriere. Bistabilele comanda activarea sau nu a rezistentei de pull-up ce da o putere mai mare pe perioada conversiei analog numerice din senzor : 1 logic determina activarea , 0 logic determina inactivarea.

Senzorul are un registru intern de 9 octeti in care se afla informatiile de configurare si temperatura gasita in biti:

octet 0 : octet LSB temperatura

octet 1 : octet MSB temperatura

octet 4 : registru de configurare

unde octetii ceilalti inter 0 si 9 nu sunt importanti pentru aplicatie.

Temperatura are urmatorul format:

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
octet LS	2^3	2^2	2^1	2^0	2^-1	2^-2	2^-3	2^-4
	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
octet MS	S	S	S	S	S	2^6	2^5	2^4

Bitii 15 11 sunt identici si au valoarea semnului temperaturii : 1 negativa , 0 pozitiva , iar bitii 10 0 sunt bitii de la 2^6 pana la 2^-4 ai numarului ce reprezinta temperatura citita in complement de 2. Pentru aflarea valorii temperaturii se citesc cei 2 octeti.

Registrul de configurare va lua valoarea 0 R1 R0 11111 , unde R1 si R0 sunt valori ce vor fi scrise. In aplicatie se alege R1=0 , R0=0 , rezolutie de 9 biti si timp de conversie 93,75 ms.

Orice comunicare cu senzorul se numeste tranzactie si se realizeaza dupa urmatorul algoritm:

1 initializare;

2 rom command 1 octet pt adresarea unuia sau mai multora din senzorii de pe aceeasi linie de date urmat de eventuale transferuri ale altor biti;

3 function command 1 octet pt a transmite tipul comenzii catre senzor urmat de eventuale transferuri ale altor biti;

Initializarea : 0 logic pe linia de date catre senzor pentru mai mult de 480 us , apoi lasa linia libera , este adusa la 1 logic de rezistenta de pull-up . Senzorul detecteaza 1 logic si dupa intre 15 us si 60 us aduce linia la 0 logic , mentinand intre 60 us si 240 us . Apoi lasa senzorul linia libera , si la mai mult de 480 trece la faza rom command.

Rom command reprezinta alegerea unuia sau a tuturor senzorilor de pe o linie , urmata de transferul unui numar de octeti necesar pentru o eventuala identificare . In aplicatie se foloseste comanda skip rom CCH , care adreseaza toti senzorii de pe linie , intrucat e de fapt unul singur , prin intervale de citire sau scriere.

Function command transmite comanda de efectuat impreuna cu eventuali alti octeti. Prima comanda folosita este write scratchpad (scrierea registrelor interne) 4EH , urmata de trei octeti : TH , TL si octet configurare descris mai sus. Valorile lui TH si TL nu conteaza in aplicatie. A doua comanda este convert temperature 44H , care porneste conversia temperaturii , urmata de activarea de strong pull-up pentru a da energia necesara conversiei analog numerice. Apoi se asteapta 100 ms pentru a da timpul necesar , asteptare urmate de dezactivarea strong pull-up-ului. A treia comanda care realizeaza citirea temperaturii transmite read scratchpad BEH ce determina citirea tuturor celor 9 octeti din registrul intern al senzorului . Primii 2 sunt octetii cel mai putin semnificativ si cel mai semnificativ al valorii determinate a temperaturii , care se salveaza , dupa care se transmite un nou reset pentru a opri transferul celorlalti 7 octeti.

Citirea si scrierea unui bit se realizeaza prin intervale de citire si scriere create de z80 prin tranzitii pe linia de date detectate de senzor .

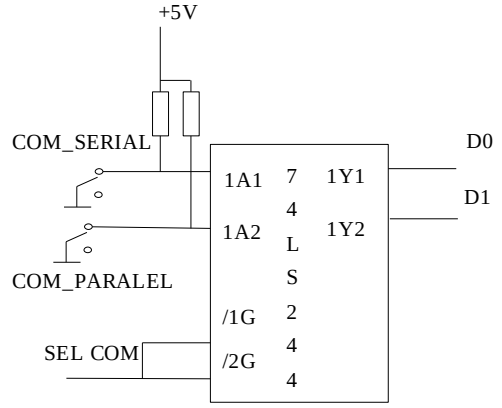
Intervalul de scriere la senaor 1 este realizat prin aducerea liniei la 0 logic pentru 15 us , apoi lasata libera pentru 45 us , revenind la 1 logic prin pull-up . Pentru scriere 0 se mentine 0 logic pentru toata durata de 60 us . Dupa cele 60 us se asteapta cel putin 1 us pentru revenire , 10 us in aplicatie , pana la scrierea unui alt bit .

In intervalul de citire de la senzor se aduce linia la 0 logic pentru mai mult de 1 us , 4 us in aplicatie . Dupa ce linia se lasa libera tinde la 1 logic , senzorul detecteaza si aduce linia la valoarea de transmis . La intre 0 us si 15 us de la lasarea libera a liniei z80 citeste linia de date , 7 us in aplicatie . Un nou interval de citire urmeaza la mai mult de 45 us dupa sfarsitul celor 15 us in care se poate realiza citirea , 50 us in aplicatie , deci 58 us cu cele 8 us dupa realizarea citirii valorii pana la initierea unui nou interval de citire in aplicatie.

```
PROGR_SENZ:     LD B , 00H ; B=00H => senzorul 1
                CALL PROGR_SENZ1
                LD B , 01H ; B=01H => senzorul 2
                CALL PROGR_SENZ1
                RET

PROGR_SENZ1:    CALL INITIALIZARE
                LD A , CCH ; skip rom
                CALL SEND_BYTE
                LD A , 4EH ; write scratchpad
                CALL SEND_BYTE
                LD A , 00H ; TH nesemnificativ
                CALL SEND_BYTE
                LD A , 00H ; TL nesemnificativ
                CALL SEND_BYTE
                LD A , 1FH ; = 00011111 binar registrul de configurare ; 9 biti semnificativi ,
                CALL SEND_BYTE ; 93,75 ms timp conversie
                RET
```

Conectarea comutatoarelor pentru a determina daca sa se transmita sau nu datele la porturile serial si paralel : la conectarea unui dispozitiv la unul din porturi trebuie activat comutatorul corespunzator :



Programe:

Organizarea memoriei SRAM pentru date(incepand cu 2000H) :

2000H : octet mai putin semnificativ citit de la senzorul 1
2001H : octet mai semnificativ citit de la senzorul 1
2002H : octet mai putin semnificativ citit de la senzorul 2
2003H : octet mai semnificativ citit de la senzorul 2
2004H : octet citit de la comutatoare
2005H : ultima valoarea transmisa la cele doua module de afisare pentru semn comandate simultan
2006H : 0 daca nu a fost activat pullup , 1 pullup pt senz 1 , 2 pullup pt senz 2

Programul principal :

```
                CALL PROGR_51
                CALL PROGR_53
                CALL PROGR_55
                CALL PROGR_SENZ
                CALL PROGR_AFIS
                LD D , 20H ; partea mai semnificativa a dublului registru DE pt adresarea ->
CICLU:          CALL CITIRE_TEMP ; -> alternativa a memoriei SRAM pt cei 2 senzori
                CALL CITIRE_COM
                CALL OUT_SERIAL
                CALL OUT_PARALEL
                CALL AFIS
                JP CICLU
```

Subrutinele :

```
CITIRE_TEMP:    LD E , 00H ; zona memorie pentru senz1
                CALL CITIRE_TEMP1
                LD E , 02H ; zona memorie pentru senz2
                CALL CITIRE_TEMP1
                RET

CITIRE_TEMP1:   CALL INITIALIZARE
                LD F , CCH ; skip rom
                CALL SEND_BYTE
                LD F , 44H ; convert temperature
                CALL SEND_BYTE
                CALL PULLUP
                CALL WAIT_100_MS
                CALL PULLUP
                CALL INITIALIZARE
                LD F , CCH ; skip rom
                CALL SEND_BYTE
                LD F , BEH ; read scratchpad
                CALL SEND_BYTE
                CALL READ_BYTE ; octet mai putin semnificativ
                LD A , 00H
                CALL SAVE_BYTE
                CALL READ_BYTE ; octet mai semnificativ
                LD A , 01H
                CALL SAVE_BYTE
                RET

CITIRE_COM:     IN A , (30H)
```

```
                    LD (2004H) , A
                    RET

OUT_SERIAL:     LD A , (2004H) ; val de la comutatoare
                    AND 01H
                    CP 01H
                    JP NZ , OUT_SERIAL1
                            RET
OUT_SERIAL1     LD A , (2000H)
                    LD C , A
                    CALL EM_51
                    LD A , (2001H)
                    LD C , A
                    CALL EM_51
                    LD A , (2002H)
                    LD C , A
                    CALL EM_51
                    LD A , (2003H)
                    LD C , A
                    CALL EM_51
                    RET

OUT_PARALEL:    LD A , (2004H)
                    AND 02H
                    CP 02H
                    JP NZ , OUT_PARALEL1
                            RET
OUT_PARALEL1:   LD A , (2000H)
                    LD C , A
                    CALL EM_55
                    LD A , (2001H)
                    LD C , A
                    CALL EM_55
                    LD A , (2002H)
                    LD C , A
                    CALL EM_55
                    LD A , (2003H)
                    LD C , A
                    CALL EM_55
                    RET

AFIS:               LD E , 00H ; senzorul 1
                    CALL AFIS1
                    LD E , 02H ; senzorul 2
                    CALL AFIS1
                    RET

AFIS1:              LD A , (DE) ; octetul mai purin semnificativ
                    AND 08H ; 0000 1000     izolare bitul ce reprezinta 0.0 sau 0.5 !
                    CP 08H
                    JP Z , AFIS2
                            LD B , FFH ; cod pt afisat 0 la un modul de afisare cu segmente
                    JP AFIS3
AFIS2:                      LD B , 48H ; cod pt afisat 5 la un modul de afisare cu segmente
AFIS3:              LD A , E; afisare 0 sau 5 la modulul 3 sau la modul 6
                    CP 00H ; E=00H     senz 1 modul 3 , E=02H senzor 2 modul 6
                    JP NZ , AFIS4
                            LD C , B0H ; adresa modul 3
                            OUT (C) , B
                    JP AFIS 5;
AFIS4:                      LD C , E0H ; adresa modul 6
```

```
                    OUT (C) , B
                ; .0 sau .5 afisat !
AFIS5:          LD A , (DE)
                AND F0H; 1111 0000 pt anularea bitilor nesemnificativi
                ROR
                ROR
                ROR
                ROR
                LD L , A ; cei 4 biti semnificativi salvati in L
                INC E
                LD A , (DE) ; octetul mai semnificativ
                AND 80H ; 10000000     testare semn pozitiv 0 , negativ 1
                JP NZ , AFIS6
                    LD B , A ; salvare
                    LD A , E
                    CP 00H
                    JP NZ ,  AFIS7
                    JP AFIS8
AFIS6:              LD B , A
                    LD A , E
                    CP 02H
                    JP NZ , AFIS9
                    JP AFIS10
AFIS7:          LD A , (2005H) ; semn pozitiv pentru senzor 1
                OR 01H ; inactivare bit semn     e trecut la 1
                LD (2005H) , A ; salvare pentru a nu corupe bitul de semn pentru celalalt senzor
                JP AFIS11
AFIS8:          LD A , (2005H); semn pozitiv pentr senzor 2
                OR 02H ; 0000 0010
                LD (2005H) , A
                JP AFIS11
AFIS9:          LD A , (2005H) ; semn negativ pentru senzor 1
                AND FEH ; 1111 1110 activare bit semn     cu 0
                LD (2005H) , A
                JP AFIS11
AFIS10:     LD A , (2005H) ; semn negativ pentru senzor 2
                AND FCH ; 1111 1101
                LD (2005H) , A
AFIS11:     OUT (80H) , A ; activare SEL_SGN
                ;sfarsit tratare bit semn

                LD A , (DE)
                AND 07H; 0000 0111 anulare biti de semn nesemnificativi , B retine semnul
                ROL
                ROL
                ROL
                ROL
                OR L ; toti bitii semnificativi in A
                LD L , A ; salvare in L
                LD A , B
                CP 80H ; pt Z e nr pozitiv , altfel conversie complement de 2     semn marime
                JP Z ,  AFIS15

; toti bitii din stanga primului bit nenul gasit din dreapta se complementeaza
                LD H , 00H ; H=00H inainte de a se intalni primul bit de 1
                LD A , 01H ; 0000 0001 ; selectie cel mai nesemnificativ bit valid
AFIS12:     LD B , A
                LD A , H
                CP 01H
                JP NZ , AFIS13
                    LD A , L
```

```
                    XOR B ; complementare bit curent
                    LD L , A ; salvare
AFIS13:     LD A , B
            AND L ; caut primul bit nenul in stanga bitului 3
            JP Z , AFIS14
                    LD H , 01H
AFIS14:     ROL
            CP 80H ; s-a ajuns la 1000 0000 ? iesire din bucla
            JP NZ , AFIS12
            ;sfarsit conversie complement de 2 negativ     semn marime pozitiv

AFIS15:     LD A , E
            CP 00H
            JP NZ , AFIS16
                    LD C , 90H ; SEG1     zecimale pt senzor1
            JP AFIS17
AFIS16:     LD C , C0H ; SEG4     zecimale pt senzor2
AFIS17:     LD A , 0AH; = 10 ; afisarea celor 2 cifre zecimale
            CP L
            JP P , AFIS18
                    OUT (C) , 02H ; afis 0
                    LD A , 00H ; numarul de scazut  din L
                    JP AFIS27
AFIS18:     LD A , 14H ; = 20
            CP L
            JP P , AFIS19
                    OUT (C) , 9EH ; afis 1
                    LD A , 0AH ; 10 de scazut
                    JP AFIS27
AFIS19:     LD A , 1EH ; = 30
            CP L
            JP P , AFIS20
                    OUT (C) ,  24H ; afis 2
                    LD A , 14H ; 20 de scazut
                    JP AFIS27
AFIS20:     LD A , 28H ; = 40
            CP L
            JP P , AFIS21
                    OUT (C) , 06H ; afis 3
                    LD A , 1EH ; 30 de scazut
                    JP AFIS27
AFIS21:     LD A , 32H ; = 50
            CP L
            JP P , AFIS22
                    OUT (C) , 98H ; afis 4
                    LD A , 28H ; 40 de scazut
                    JP AFIS27
AFIS22:     LD A , 3C ; = 60
            CP L
            JP P , AFIS23
                    OUT (C) , 48H ; afis 5
                    LD A , 32H , 50 de scazut
                    JP AFIS27
AFIS23:     LD A , 46H ; = 70
            CP L
            JP P , AFIS24
                    OUT (C) , 40H ; afis 6
                    LD A , 3CH ; 60 de scazut
                    JP AFIS27
AFIS24:     LD A , 50H ; = 80
            CP L
```

```
                    JP P , AFIS25
                        OUT (C) , 1EH ; afis 7
                        LD A , 46H ; 70 de scazut
                        JP AFIS27
AFIS25:     LD A , 5AH
                    CP L
                    JP P , AFIS26
                        OUT (C) , FEH ; afis8
                        LD A , 50H ; 80 de scazut
                        JP AFIS27
AFIS26:     OUT (C) , 08H ; afis9
                    LD A , 5AH ; 90 de scazut
AFIS27:     LD B , A
                    LD A , L
                    SUB B ;
                    INC C
                    CP 01H
                    JP P , AFIS28
                        OUT (C) , 03H ; 0 cu punct
                        JP AFIS37
AFIS28:     CP 02H
                    JP P , AFIS29
                        OUT (C) , 9FH ; 1 cu punct
                        JP AFIS37
AFIS29:     CP 03H
                    JP P , AFIS30
                        OUT (C) , 25H ; 2 cu punct
                        JP AFIS37
AFIS30:     CP 04H
                    JP P , AFIS31
                        OUT (C) , 07H ; 3 cu punct
                        JP AFIS37
AFIS31:     CP 05H
                    JP P , AFIS32
                        OUT (C) , 99H ; 4 cu punct
                        JP AFIS37
AFIS32:     CP 06H
                    JP P , AFIS33
                        OUT (C) , 49H ; 5 cu punct
                        JP AFIS37
AFIS33:     CP 07H
                    JP P , AFIS34
                        OUT (C) , 41H ; 6 cu punct
                        JP AFIS37
AFIS34:     CP 08H
                    JP P , AFIS35
                        OUT (C) , 1FH ; 7 cu punct
                        JP AFIS37
AFIS35:     CP 09H
                    JP P , AFIS36
                        OUT (C) , FFH ; 8 cu punct
                        JP AFIS37
AFIS36:     OUT (C) , 09H ; 9 cu punct
                    ;sfarsit afisare cifre zecimale si iesire functie
AFIS37:     RET

INITIALIZARE:   LD A , E
                    CP 00H
                    JP NZ , INITIALIZARE1
                        LD C , 50H; pt senz 1
                        LD A , FBH ; 1111 1011
```

```
                    JP INITIALIZARE2
INITIALIZARE1:      LD C , 70H; pt senz 2
                        LD A , F7H ; 1111 0111
INITIALIZARE2:  OUT (C) , A
                    LD B , D0H ; 250 us
                    CALL WAIT
                    LD B , D0H ; 250 us
                    CALL WAIT ; asteptare 500 us in total
                    LD A , FFH
                    OUT (C) , A
                    RET

SEND_BYTE:      LD A , E
                    CP 00H
                    JP NZ , SEND_BYTE1
                        LD L , FBH ; = 1111 1011 pt senz 1
                        LD C , 50H
                        LD A , F
SEND_BYTE1:     JP SEND_BYTE2
                        LD L , F7H ; = 1111 0111 pt senz 2
                        LD C , 70H
                        LD A , F
                        ROL
SEND_BYTE2:     ROL
                    ROL
                    ROL
                    LD H , 08H
SEND_BYTE3:     LD F , A
                    LD A , L
                    OR F
                    CP FFH
                    OUT (C) , 00H
                    JP Z , SEND_BYTE4
                        LD B , 03H
                        CALL WAIT
                        OUT (C) , FFH ; se transmite 1 la senzor , altfel 0
SEND_BYTE4:     LD B , 32H ; = 50 zecimal => 60 us asteptare
                    CALL WAIT
                    OUT (C) , FFH ; daca a fost 0 se revine
                    LD A , H
                    CP 00H
                    JP NZ , SEND_BYTE3 ; ciclare pt toti cei 8 biti
                    RET

READ_BYTE:      LD A , E
                    LD F , 00H
                    LD H , 08H
                    CP 00H
                    JP NZ , READ_BYTE1
                        LD C . 50H
                        LD L , 04H; 0000 0100
                    JP READ_BYTE2
READ_BYTE1:         LD C , 70H
                        LD L , 08H; 0000 1000
READ_BYTE2:     OUT (C) , 00H ; initiere secventa citire bit
                    OUT (C) , FFH
                    LD B , 03H
                    CALL WAIT ; asteptare ~ 5 sau 6 us;
                    IN A , (C)
                    LD B , 29H ; = 42 zecimal  , 50 us
                    AND L
```

```
                    CP 00H
                    LD A , F
                    ROL
                    JP Z , READ_BYTE3
                        ADD 01H ; bit de 1 citit , altfel 0
READ_BYTE3:     LD F , A
                    LD A , H
                    SUB 01H
                    JP NZ , READ_BYTE2
                    RET

SAVE_BYTE:          LD B , E
                    CP 00H
                    JP Z , SAVE_BYTE1
                        LD A , E
                        ADD 01H
                        LD E , A
SAVE_BYTE1:     LD A , F
                    OUT (DE) , A
                    LD E , B
                    RET

PULLUP:         IN A , (2006H)
                    CP 01H ; = 0000 0001    izolare bitul pt senz 1
                    JP NZ PULLUP2
PULLUP1:                LD A , 00H
                        OUT (FFH) , A
                        LD A , 00H
                        LD (2006H) , A
                        JP PULLUP6
PULLUP2:            CP 02H ; = 0000 0010    izolare bit pt senz 2
                    JP NZ PULLUP3
                        JP PULLUP1
PULLUP3:            LD A , E
                    CP 00H
                    JP NZ PULLUP4
                        LD A , 01H
                    JP PULLUP5
PULLUP4:                LD A , 02H
PULLUP5:            OUT (FFH) , A
                    LD (2006H) , A
PULLUP6:            RET

WAIT_100_MS:    LD L , 0BH ; = 10
WAIT_100_MS1:   LD A , 64H ; = 100
WAIT_100_MS2:       LD B , 54H ; ~ 83 zecimal => ~ 100 us
                        CALL WAIT
                        SUB 01H
                    JP NZ , WAIT_100_MS2
                    LD A , L
                    SUB 1
                    LD L , A
                    CP 00H
                    JP NZ , WAIT_100_MS1
                    RET

WAIT:               DJNZ WAIT ; 1.2 us pe ciclu
                    RET
```