

PROIECTAREA SISTEMELOR NUMERICE

TIMER

Proiectat si implementat de catre:

Gog Rares-Flaviu

Grupa 30414

Laborant: Cristian Gabriel Albu

**CUPRINS**

**CONTENTS**

**SPECIFICATIA PROIECTULUI............................................................................................3**

**SCHEMA BLOC.......................................................................................................................4**

**DIAGRAMA PROIECTULUI.................................................................................................5**

**COMPONENTE UTILIZATE.................................................................................................6**

**CODUL COMPONENTELOR & EXPLICATIA ACESTORA...........................................6**

* **NUMARATOR MODULO 10......................................................................................6**
* **NUMARATOR MODULO 6........................................................................................8**
* **DIVIZOR DE FRECVENTA .....................................................................................10**
* **DISPLAY .....................................................................................................................11**
* **DEBOUNCER ............................................................................................................14**
* **UNITATEA DE CONTROL .......................................................................................17**

**INSTRUCTIUNI DE UTILIZARE .......................................................................................21**

**FISIERUL DE CONSTRANGERI .......................................................................................22**

**CREAREA FISIERULUI .BIT ..............................................................................................25**

**JUSTIFICARE SOLUTIE ALESE .......................................................................................26**

**POSIBILITATI DE DEZVOLTARE ULTERIOARE .........................................................26**

**SPECIFICATIA PROIECTULUI**

Să se proiecteze un timer cu următoarea funcţionalitate: dispozitivul are 4 afişaje BCD - 7 segmente. Primele două afişaje sunt pentru minute, următoarele două pentru secunde. Astfel, valoarea maximă care poate fi afişată este de 99 minute şi 59 secunde.

Dispozitivul are 3 butoane:

\*M (de la Minute)

\*S (de la Secunde)

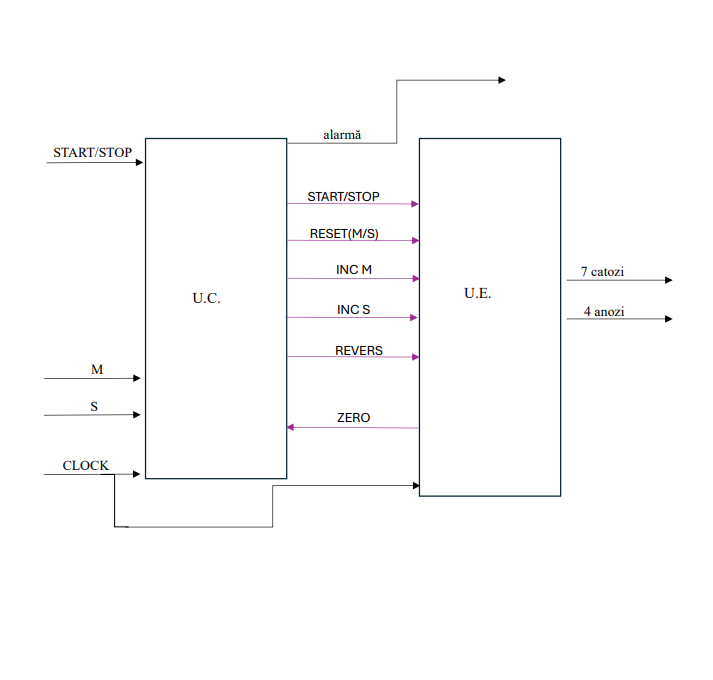
\*START / STOP.

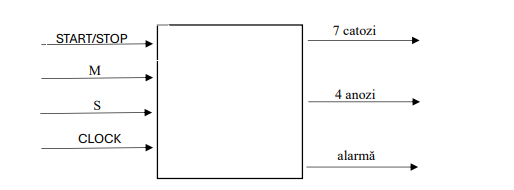
Presupunând că iniţial este în starea ZERO, dacă se apasă butonul START / STOP, timerul începe să numere crescător. Dacă se apasă din nou butonul START / STOP, timerul se opreşte la valoarea atinsă în momentul respectiv. Dacă se apasă din nou butonul START / STOP, timerul continuă să numere etc. Dacă ajunge la 99 de minute şi 59 de secunde, urmează din nou ZERO. Dacă se apasă simultan butoanele M (de la Minute) şi S (de la Secunde), timerul se resetează (devine ZERO).

În orice stare, dacă se apasă butonul M, se va incrementa şi afişa valoarea minutelor. În orice stare, dacă se apasă butonul S, se va incrementa şi afişa valoarea secundelor. O dată ce s-a setat o valoare pentru minute şi / sau secunde (prin apăsarea butoanelor M sau S), când se apasă butonul START / STOP, timerul începe să numere descrescător de la valoarea curentă „Minute / Secunde” până la ZERO, iar când se ajunge în starea ZERO se emite un semnal sonor (alarmă) pentru o perioadă de timp care poate fi setată iniţial.

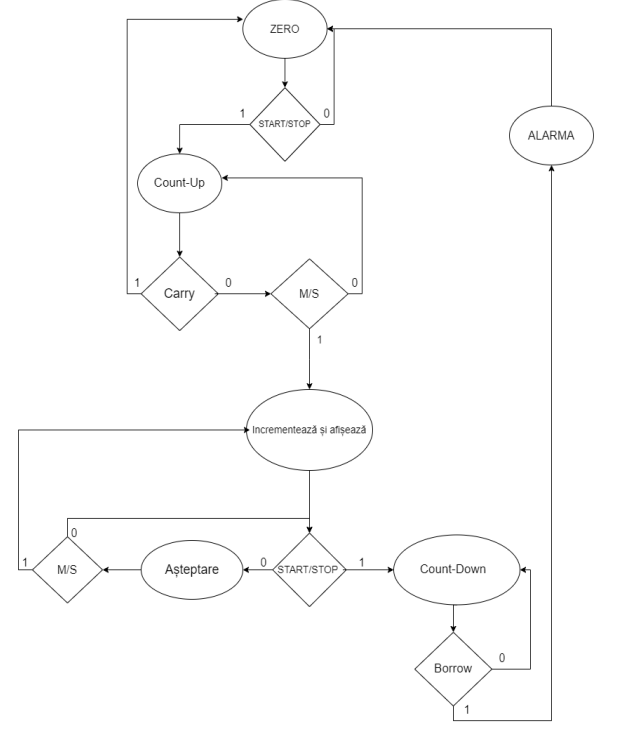
Se consideră că există disponibil un semnal periodic cu frecvenţa de 1 Hz. Proiectul va fi realizat de 1 student.

**SCHEMA BLOC**

****

****

**DIAGRAMA PROIECTULUI**

****

**COMPONENTE UTILIZATE**

1. Numarator modulo 10 ( 3 componente)
2. Numarator modulo 6 ( 1 componenta)
3. Debouncer ( 3 componente)

\*2 debouncere mentin valoarea 1

\*1 debouncer are valoarea 1 o singura data

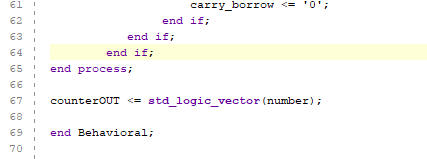
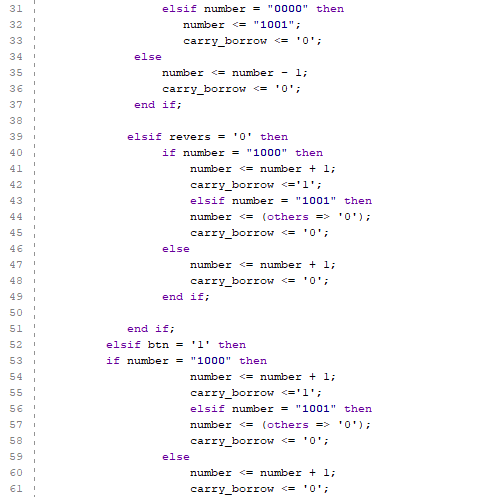
1. Display : 7-SEGMENT + Anode picker ( 1 componenta)
2. Clock Divider 100MHZ –> 1HZ (1 componenta)
3. Unitatea de comanda (1 componenta)

**CODURILE COMPONENTELOR**

**NUMARATOR MODULO 10**

A screenshot of a computer code

Description automatically generated



In numaratorul modulo 10, denumit „MOD10CTR” , avem ca intrari:

* Clk -> un ceas cu frecventa de 1hz
* Btn -> daca este cazul , butonul pentru incrementarea minutelor/secundelor
* Rst -> semnal utilizat pentru pentru resetare
* Revers -> semnal utilizat pentru determinarea mersului invers
* Enable -> semnal utilizat pentru a functionarea numaratorului

Iesirile din numarator sunt:

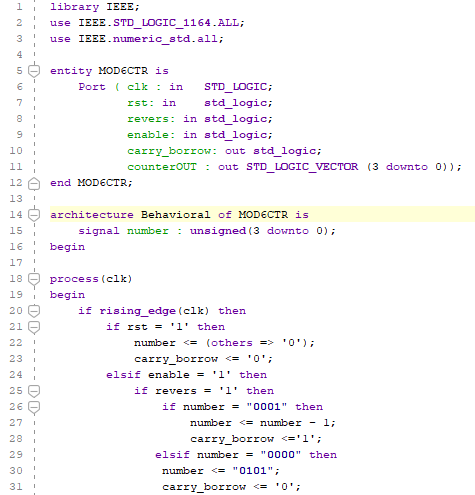
* Carry\_borrow -> semnal utilizat pentru determinarea folosirii urmatorului numarator
* counterOUT -> semnal cu marimea de 4 biti pentru a fi utilizat la afisarea cifrei

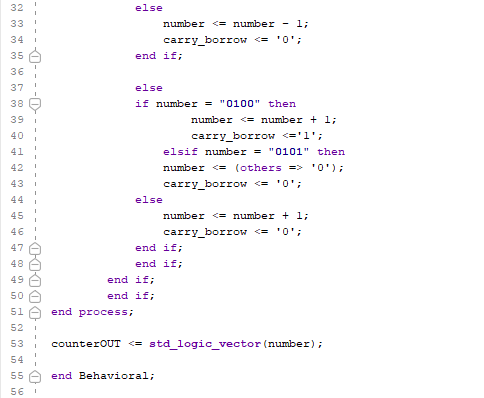
De asemenea este declarat un semnal, „number”, prin intermediul caruia vom numara in sus sau in jos.

Procesul localizat in acest numarator functioneaza pe baza clock-ului de 1HZ. In cazul in care semnalul acestuia este 1 se pot intampla urmatoarele lucruri:

* daca semnaul de resetare este 1 ( rst = ’1’ ) atunci numararea se reseteaza, semnalul de 4 biti si semnalul de carry/borrow fiind setate la 0 ;
* daca semnalul de enable este 1 ( enable = ’1’) atunci, pe baza semnaulului „revers” se va numara crescator de la 0 la 9, respectiv descrescator de la 9 la 0; semnalul „carry\_borrow” are valoare 1 in cazul in care se ajunge la 8, respectiv 1, pentru a transmite carry/borrow-ul urmatorului numarator ( valorile 1 si 8 au fost alese pentru a nu se produce un delay intre numaratoare si ca acestea sa numere corespunzator cerintei) ;
* in cazul in care semnalul btn este activ prin intermediul apasarii butonului de pe placa se va incrementa in sus din punctul in care se afla numaratorul inaintea apasarii butonului (semnaul carry\_borrow functioneaza identic cu explicatia de mai sus) ;
* la finalul numaratorului semnalul counterOUT primeste cifra in biti pentru a fi transmisa mai departe urmatoarei componente ;

**NUMARATOR MODULO 6**

****

****

In numaratorul modulo 6, denumit „MOD6CTR” , avem ca intrari:

* Clk -> un ceas cu frecventa de 1hz
* Rst -> semnal utilizat pentru pentru resetare
* Revers -> semnal utilizat pentru determinarea mersului invers
* Enable -> semnal utilizat pentru a functionarea numaratorului

Iesirile din numarator sunt:

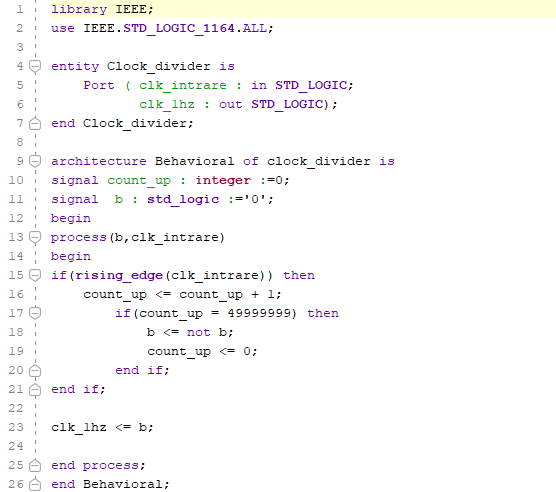
* Carry\_borrow -> semnal utilizat pentru determinarea folosirii urmatorului numarator
* counterOUT -> semnal cu marimea de 4 biti pentru a fi utilizat la afisarea cifrei

De asemenea este declarat un semnal, „number”, prin intermediul caruia vom numara in sus sau in jos.

Procesul localizat in acest numarator functioneaza pe baza clock-ului de 1HZ. In cazul in care semnalul acestuia este 1 se pot intampla urmatoarele lucruri:

* daca semnaul de resetare este 1 ( rst = ’1’ ) atunci numararea se reseteaza, semnalul de 4 biti si semnalul de carry/borrow fiind setate la 0 ;
* daca semnalul de enable este 1 ( enable = ’1’) atunci, pe baza semnaulului „revers” se va numara crescator de la 0 la 5, respectiv descrescator de la 5 la 0; semnalul „carry\_borrow” are valoare 1 in cazul in care se ajunge la 4, respectiv 1, pentru a transmite carry/borrow-ul urmatorului numarator ( valorile 1 si 4 au fost alese pentru a nu se produce un delay intre numaratoare si ca acestea sa numere corespunzator cerintei) ;
* la finalul numaratorului semnalul counterOUT primeste cifra in biti pentru a fi transmisa mai departe urmatoarei componente ;

**DIVIZOR DE FRECVENTA**

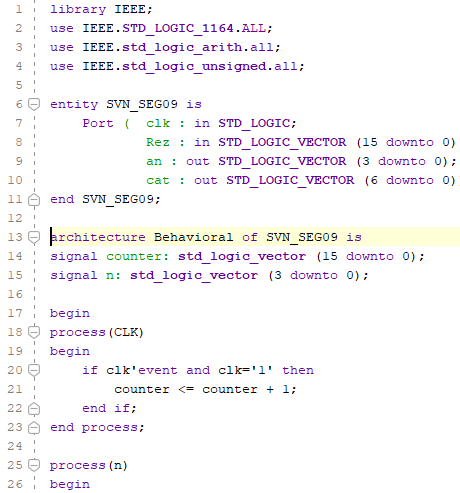
****

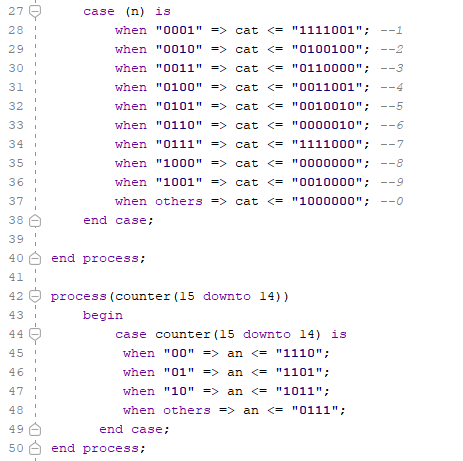
In acest divizor de frecventa, denumit „clock\_divider”, avem ca intrarea clock-ul placutei de 100MHZ si ca iesire, un nou clock de 1HZ. Avem 2 semnale declarate in arhitectura componentei:

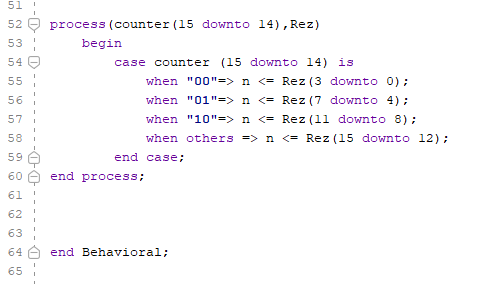
* count\_up, de tip integer, prin intermediul acesteia vom numara in sus pentru a diviza frecventa clock-ului
* b, de tip std\_logic, prin care vom transmite noului clock semnalul corespunzator pentru a functiona corect

Procesul localizat in aceasta componenta functioneaza pe baza clock-ului de 100MHZ. In cazul in care avem „rising edge” la acest clock se va numara in sus pana se ajunge la ~50 milioane. Cand se ajunge la aceasta valoare semnalul „b” va primi opusul lui si numararea se va reseta. La finalul componentei noul clock, „clk\_1hz” primeste valoarea lui b.

**DISPLAY**

****

****

****

In aceasta componenta de afisare pe placa, denumita „SVN\_SEG09”, avem ca intrari:

* Clk -> clock-ul intern al placii
* Rez -> semnal de 16 biti prin intermediul caruia sunt primite cele 4 cifre in BCD

Iesirile din aceasta componenta sunt:

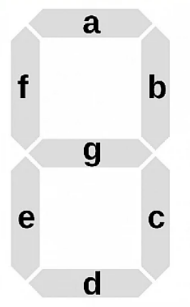
* An -> semnal de 4 biti prin intermediul caruia se alege anodul
* Cat -> semnal de 7 biti prin intermdeiul caruia se alege catodul

De asemenea, sunt declarate 2 semnale:

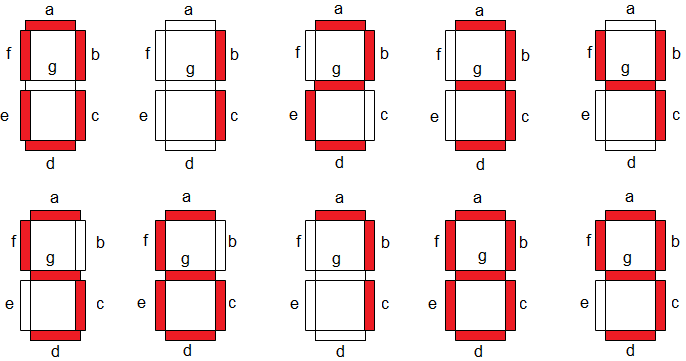
* Counter -> semnal utilizat pentru alegerea anodului si a cifrei
* n -> semnal utilizat pentru alegerea cifrei corespunzatoare

In aceasta componeta avem 4 procese:

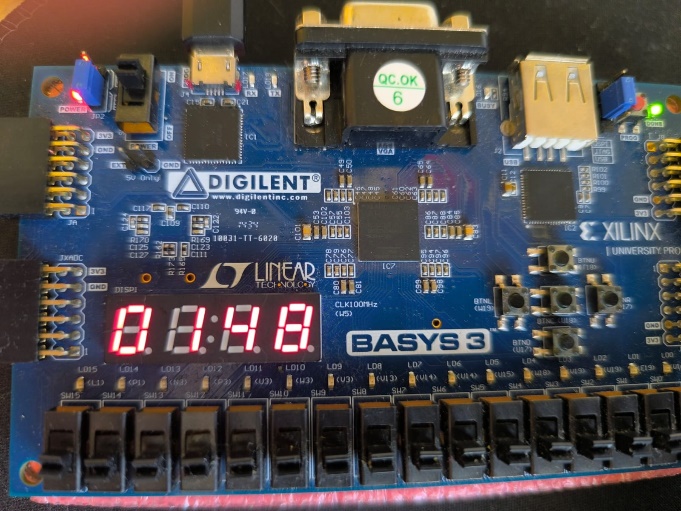
Primul proces numara in sus prin intermediul semnalului counter, pe baza clock-ului primit ca intrare in componenta. Al doilea proces realizeaza cifra corespunzatoare fiecarui caz primit in semnalul „n”. De exemplu, daca „n” este 0101 (5) atunci semnalul primit de cat, in logica negativa, va fi 0010010 (gfedcba). In urmatoarea imagine este reprezentat fiecare catod cu litera (functia) corespunzatoare.



Astfel, va fi reprezentata fiecare cifra in functie de semnalul primit in „n”.

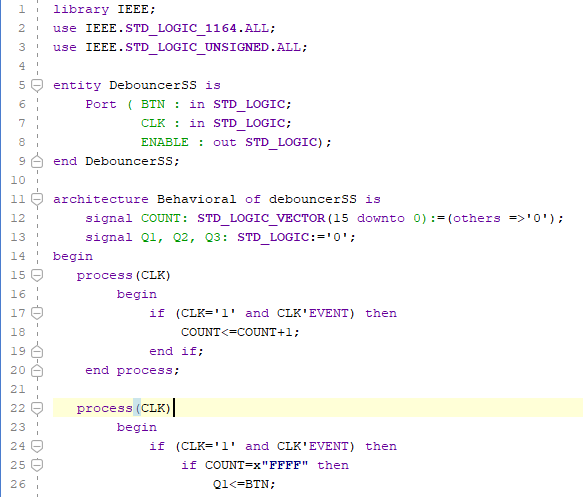


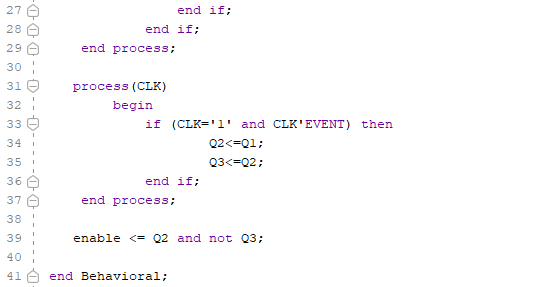
In al treilea proces, prin intermediul ultimilor 2 biti ai semnalului „counter”, „an” primeste valoarea anodului corespunzator acestor biti. In al patrulea proces, tot prin intermediul ultimilor 2 biti ai semnalului „counter”, „n” va primi cifra corespunzatoare in BCD. Astfel, prin intermediul ultimelor doua procese, ficare anod va primi cifra care ii conrespunde. De exemplu, pentru primul anod „an” este 1110 si „n” va lua valoarea primilor 4 biti ai semnalului „Rez”. In final, prin intermediul acestor procese va fi afisat pe display timpul realizat de numaratoare.



**DEBOUNCER**

**1.DebouncerSS**

****

****

In aceasta componenta, denumita „DebouncerSS”, avem urmatoarele intrari:

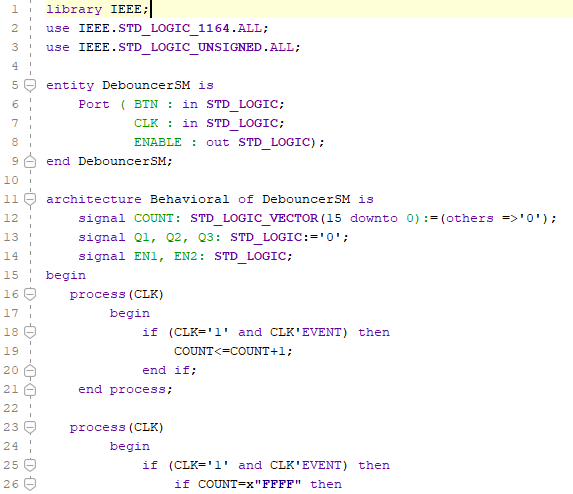
* btn -> semnaul transmis de buton in momentul apasarii
* Clk -> clock-ul intern al placii

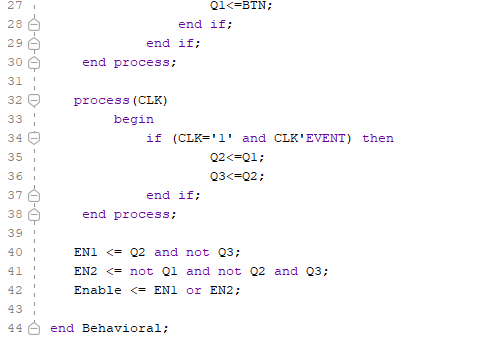
Ca iesire avem semnaulul „Enable” care va fi butonul trecut prin acest Debouncer. De asemenea, avem semnale declarate precum : count, Q1, Q2 si Q3.

Prin intermediul acestui proces am realizat 3 bistabile. Primele 2 vor fi 1 cat timp este apasat butonul, iar ultimul va fi 1 cand se va ridica degetul de pe buton pentru o perioada scurta de timp. Astfel, cand se va apasa butonul enable va lua valoare 1 o singura data si in acest mod putem considera ca butonul a fost apasat o singura data.

Acest tip de debouncer va fi folosit numai pentru butonul Start/Stop.

**2.DebouncerSM**

****



In aceasta componenta, denumita „DebouncerSM”, avem urmatoarele intrari:

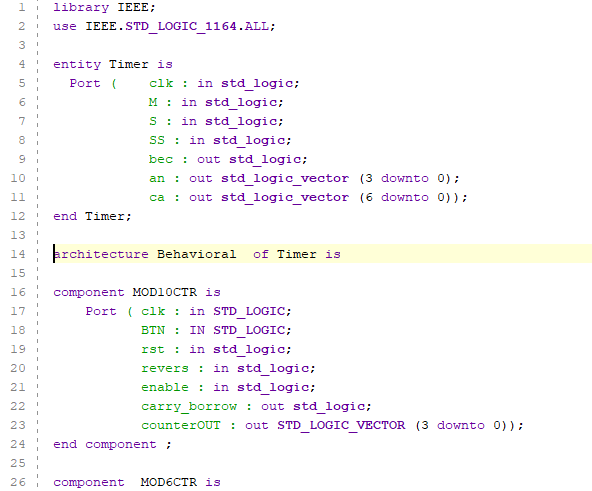
* btn -> semnaul transmis de buton in momentul apasarii
* Clk -> clock-ul intern al placii

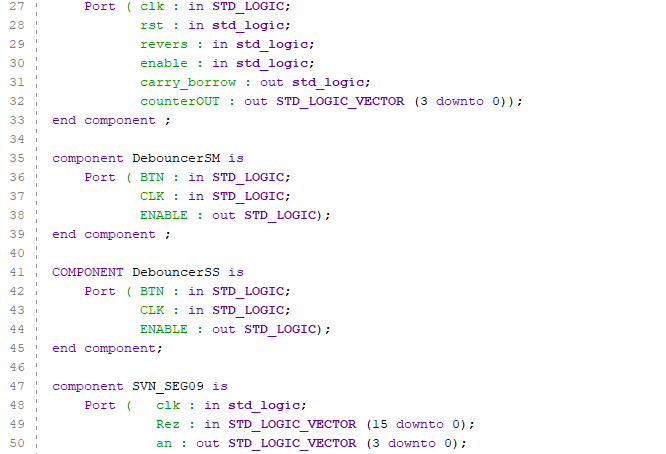
Ca iesire avem semnaulul „Enable” care va fi butonul trecut prin acest Debouncer. De asemenea, avem semnale declarate precum : count, Q1, Q2,Q3,EN1 si EN2.

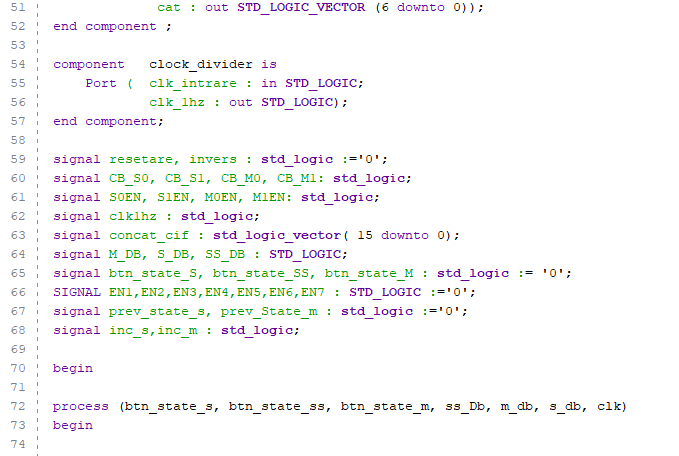
Prin intermediul acestui proces am realizat 3 bistabile. Primele 2 vor fi 1 cat timp este apasat butonul, iar ultimul va fi 1 cand se va ridica degetul de pe buton pentru o perioada scurta de timp. Prin intermediul semnalelor EN1 si EN2 vom verifica starea bistabilelor. Astfel, butonul va f i considerat 1 cat timp este apasat, valoarea lui fiind 0 in momentul in care ridicam degetul de pe buton. Semnalul „Enable” primind doar o valoare dintre cele doua semnale.

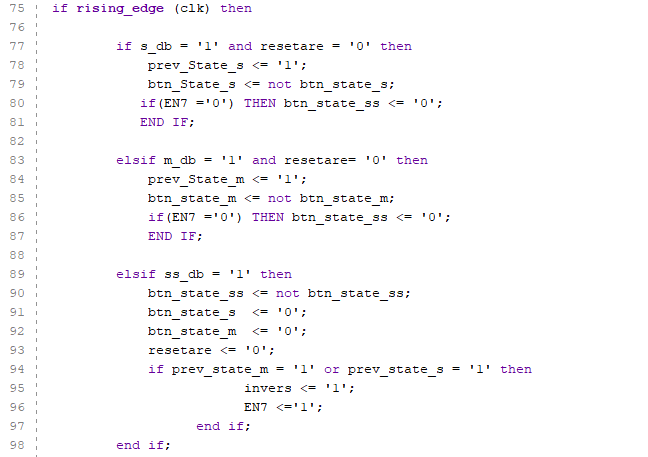
Acest tip de debouncer va fi folosit numai pentru butonoanele de incrementare a minutelor si a secundelor (S si M).

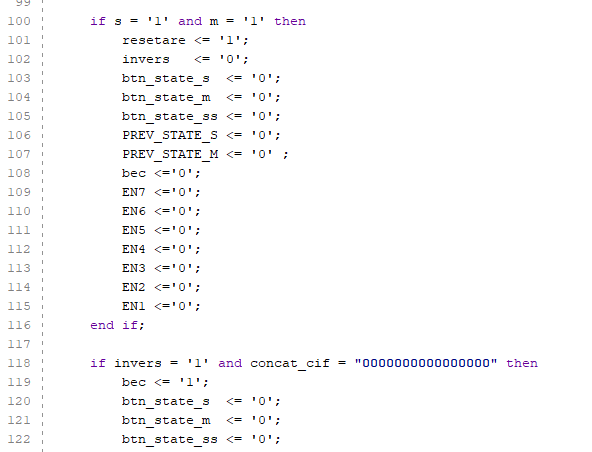
**UNITATEA DE CONTROL**

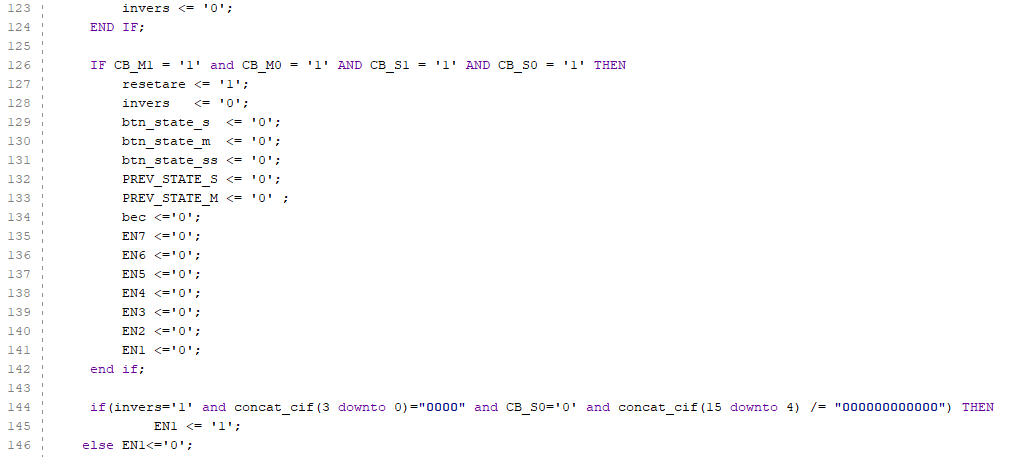


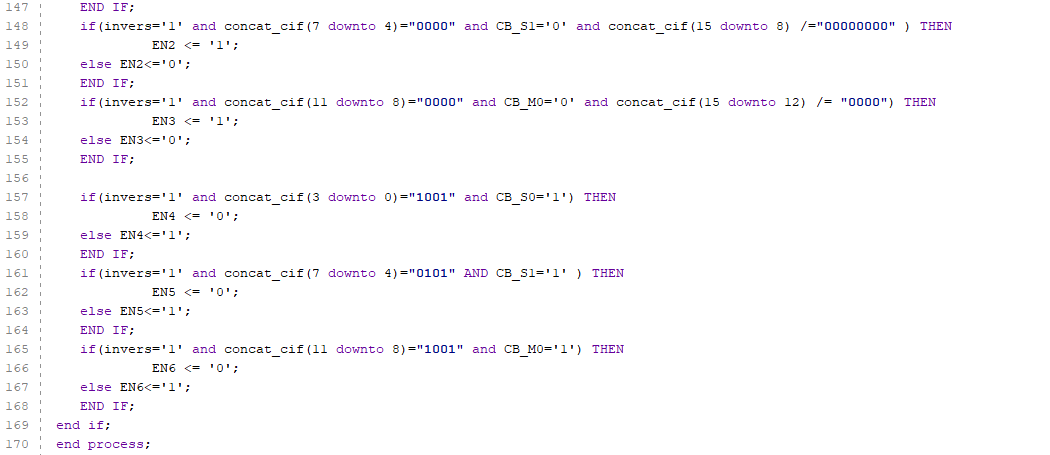


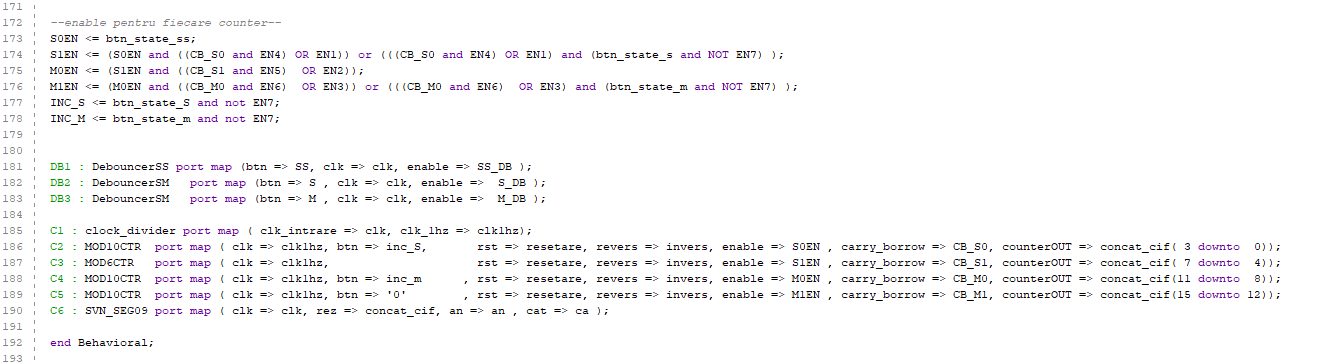












In unitatea de control, denumita „Timer”, avem ca intrari:

* Clk -> clock-ul intern al placii
* S -> butonul pentru incrementarea secundelor si resetare
* M -> butonul pentru incrementarea minutelor si resetare
* Ss -> butonul pentru Start/Stop

Unitatea de control are urmatoarele iesiri:

* An -> semnalul de selectare a anodului
* Cat -> semnalul de selectare a catodului
* Bec -> semnalul de aprindere a becului

In unitatea de control se regasesc toate componentele declarate la inceputul arhitecturii pentru a fi utilizate. De asemenea, avem semnale declarate in aceasta componenta:

* Resetare -> semnal cu nume sugestiv, in cazul in care acest semnal este 1, programul se va reseta
* Invers -> semnal cu nume sugestiv, in cazul in care acest semnal este 1, numărarea se va face in sens invers
* CB\_S0 , CB\_S1, CB\_M0, CB\_M1 -> aceste semnale reprezinta carry/borrow -ul fiecarui numarator, daca oricare este 1, acesta va intra in logica de pornire a urmatorului numarator
* S0EN, S1EN, M0EN, M1EN -> semnale care permit numaratoarelor sa functioneze
* Clk1hz -> semnal pentru clock -ul de 1HZ
* Concat\_cif -> semnal cu dimensiunea de 16 biti, care permite memorarea cifrelor in BCD pentru a fi transmise componentei de display
* M\_DB, S\_DB, SS\_DB -> semnale ce reprezinta fiecare buton dupa ce a trecut printr-un debouncer
* Btn\_state\_s , Btn\_state\_m, Btn\_state\_SS -> semnale ce reprezinta starea in care se afla butonul desemnat acesteia
* EN1, EN2, EN3, EN4, EN5, EN6, EN7 -> semnale ce primesc valoarea 1 in diferite cazuri pentru a repara anumite erori

In procesul localizat in unitatea de control se vor prelucra mai multe elemente. Aceste elemente vor fi explicate mai jos in functie de liniile de cod unde se regasesc:

* 77 – 98 -> In cazul in care semnalul s\_Db este 1 si resetare este 0 atunci programul va memora faptul ca butonul a fost apasat, in prev\_state\_s, si va schimba starea semnalului btn\_state\_s pentru a incrementa secundele. De asemenea, daca numararea din start/stop era activa si mergea crescator, se va pune pauza.

-> In cazul in care semnalul m\_Db este 1 si resetare este 0 atunci programul va memora faptul ca butonul a fost apasat, in prev\_state\_m, si va schimba starea semnalului btn\_state\_m pentru a incrementa minutele. De asemenea, daca numararea din start/stop era activa si mergea crescator, se va pune pauza.

-> In cazul in care semnalul ss\_Db este 1 atunci btn\_state\_ss isi va schimba schima valoare cu opusul ei. Astfel, numaratoarea porneste sau se opreste. Totodata, incrementarile si resetarea primesc valoarea 0 si, in cazul in care a fost realizata o incrementare, va incepe numaratoarea inversa si EN7 primeste valoarea 1.

* 100 – 116 -> In cazul in care sunt apasate ambele butoane programul se va reseta, toate semnalele primind valoarea 0 si resetare primind valoarea 1.
* 118 – 124 -> In cazul in care numaratoarea inversa ajunge la 00:00 atunci se va aprinde un bec si programul se va opri.
* 126 – 142 -> In cazul in care numaratoarea crescatoare ajunge la 99:59 codul va trece in starea initiala.
* 144 – 155 -> In aceasta parte a codului se verifica, in momentul cand se numara invers si se ajunge la 0, daca exista cifre diferite de 0 in stanga celorlalte cifre si nu exista borrow. Daca se intampla acest lucru atunci EN1, EN2 sau EN3, in functie de pozitia cifrei care a intampinat aceasta problema, va primi valoarea 1.
* 157 – 169 -> In cazul in care se incrementeaza pana la 9, respectiv 5 in cazul secundelor, va fi preluat un carry. Acest carry ramane si daca incepe numaratoarea inversa. Aceasta parte din cod previne acest lucru.

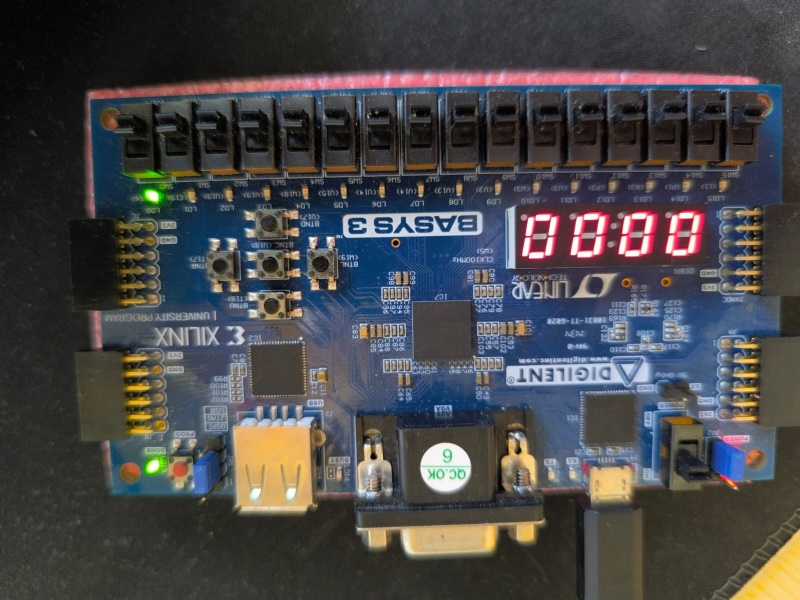
Liniile 172 – 178 prelucreaza datele primite pentru a permite functionarea numaratoarelor. Aceasta parte nu este regasita in proces. De asemenea, in aceast cod se permite incrementarea numai in cazul in care nu functioneaza numaratoarea inversa. In liniile 181 - 190 sunt utilizate componetele specificate mai sus. Fiecare semnal de intrare/iesire fiind conectat cu un alt semnal astfel incat este realizata conexiunea corecta intrea componente.

**INSTRUCTIUNI DE UTILIZARE**

Placa utilizata are 5 butoane, din care proiectul utilizeaza numai 3. Butonul din mijloc este butonul de START/STOP, butonul din stanga este butonul de incrementare a minutelor (M), iar butonul din dreapta este butonul de incrementare a secundelor (S). Butoanele S si M vor incrementa cate o secunda/minut, la un interval de 1 secunda, numai daca este mentinut degetul apasat pe butonul respectiv. In momentul in care utilizatorul apasa ambele butoane numararea se va reseta. In cazul in care se doreste incrementarea dupa resetare este necesara o apasare pe butonul START/STOP pentru a activa butoanele S si M. Butonul START/STOP, cat timp nu a fost apast butonul S sau M, in momentul apasarii va incepe numaratoarea crescatoare sau va pune pauza acestei numaratori. In cazul in care a fost apasat unul dintre celelalte doua butoane atunci, in momentul apasarii butonului START/STOP va incepe numaratoarea inversa, butoanele S si M devenind indisponibile incrementarii. In momentul in care se ajunge la 00:00 numaratoarea se va opri si se va aprinde un bec.

Timpul este afisat pe partea digitala a placii timpul fiind reprezentat astfel: (XY:ZT)

Zeci de minute(X) ~ Minute(Y) : Zeci de scunde(Z) ~ Secunde(T)



Afisarea timpului pe display.

Buton pentru incrementarea minutelor.

Butonul START/STOP.

Butonul pentru incrementarea secundelor.

Becul ce se aprinde cand timpul ajunge la 00: 00.

**FISIERUL DE CONSTRANGERI**

## Clock signal

set\_property PACKAGE\_PIN W5 [get\_ports clk]

set\_property IOSTANDARD LVCMOS33 [get\_ports clk]

create\_clock -add -name sys\_clk\_pin -period 10.00 -waveform {0 5} [get\_ports clk]

## LEDs

set\_property PACKAGE\_PIN U16 [get\_ports {bec}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {bec}]

#set\_property PACKAGE\_PIN E19 [get\_ports {led[1]}]

#set\_property IOSTANDARD LVCMOS33 [get\_ports {led[1]}]

#set\_property PACKAGE\_PIN U19 [get\_ports {led[2]}]

#set\_property IOSTANDARD LVCMOS33 [get\_ports {led[2]}]

#set\_property PACKAGE\_PIN V19 [get\_ports {led[3]}]

#set\_property IOSTANDARD LVCMOS33 [get\_ports {led[3]}]

#set\_property PACKAGE\_PIN W18 [get\_ports {led[4]}]

#set\_property IOSTANDARD LVCMOS33 [get\_ports {led[4]}]

#set\_property PACKAGE\_PIN U15 [get\_ports {led[5]}]

#set\_property IOSTANDARD LVCMOS33 [get\_ports {led[5]}]

#set\_property PACKAGE\_PIN U14 [get\_ports {led[6]}]

#set\_property IOSTANDARD LVCMOS33 [get\_ports {led[6]}]

#set\_property PACKAGE\_PIN V14 [get\_ports {led[7]}]

#set\_property IOSTANDARD LVCMOS33 [get\_ports {led[7]}]

#set\_property PACKAGE\_PIN V13 [get\_ports {led[8]}]

#set\_property IOSTANDARD LVCMOS33 [get\_ports {led[8]}]

#set\_property PACKAGE\_PIN V3 [get\_ports {led[9]}]

#set\_property IOSTANDARD LVCMOS33 [get\_ports {led[9]}]

#set\_property PACKAGE\_PIN W3 [get\_ports {led[10]}]

#set\_property IOSTANDARD LVCMOS33 [get\_ports {led[10]}]

#set\_property PACKAGE\_PIN U3 [get\_ports {led[11]}]

#set\_property IOSTANDARD LVCMOS33 [get\_ports {led[11]}]

#set\_property PACKAGE\_PIN P3 [get\_ports {led[12]}]

#set\_property IOSTANDARD LVCMOS33 [get\_ports {led[12]}]

#set\_property PACKAGE\_PIN N3 [get\_ports {led[13]}]

#set\_property IOSTANDARD LVCMOS33 [get\_ports {led[13]}]

#set\_property PACKAGE\_PIN P1 [get\_ports {led[14]}]

#set\_property IOSTANDARD LVCMOS33 [get\_ports {led[14]}]

#set\_property PACKAGE\_PIN L1 [get\_ports {led[15]}]

#set\_property IOSTANDARD LVCMOS33 [get\_ports {led[15]}]

##7 segment display

set\_property PACKAGE\_PIN W7 [get\_ports {ca[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {ca[0]}]

set\_property PACKAGE\_PIN W6 [get\_ports {ca[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {ca[1]}]

set\_property PACKAGE\_PIN U8 [get\_ports {ca[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {ca[2]}]

set\_property PACKAGE\_PIN V8 [get\_ports {ca[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {ca[3]}]

set\_property PACKAGE\_PIN U5 [get\_ports {ca[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {ca[4]}]

set\_property PACKAGE\_PIN V5 [get\_ports {ca[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {ca[5]}]

set\_property PACKAGE\_PIN U7 [get\_ports {ca[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {ca[6]}]

#set\_property PACKAGE\_PIN V7 [get\_ports dp]

#set\_property IOSTANDARD LVCMOS33 [get\_ports dp]

set\_property PACKAGE\_PIN U2 [get\_ports {an[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {an[0]}]

set\_property PACKAGE\_PIN U4 [get\_ports {an[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {an[1]}]

set\_property PACKAGE\_PIN V4 [get\_ports {an[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {an[2]}]

set\_property PACKAGE\_PIN W4 [get\_ports {an[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {an[3]}]

##Buttons

set\_property PACKAGE\_PIN U18 [get\_ports SS]

set\_property IOSTANDARD LVCMOS33 [get\_ports SS]

#set\_property PACKAGE\_PIN T18 [get\_ports btnU]

#set\_property IOSTANDARD LVCMOS33 [get\_ports btnU]

set\_property PACKAGE\_PIN W19 [get\_ports M]

set\_property IOSTANDARD LVCMOS33 [get\_ports M]

set\_property PACKAGE\_PIN T17 [get\_ports S]

set\_property IOSTANDARD LVCMOS33 [get\_ports S]

#set\_property PACKAGE\_PIN U17 [get\_ports btnD]

#set\_property IOSTANDARD LVCMOS33 [get\_ports btnD]

**CREAREA FISIERULUI .BIT**

**A screenshot of a computer program

Description automatically generated**

Se apasa butonul “Generate Bitstream”

**A screenshot of a computer

Description automatically generated**

Apasam butonul “OK”

Asteptam pana se termina prelucrarea datelor introduse.

A screenshot of a computer

Description automatically generated

Selectam “Open Hardware Manager” pentru implementarea pe placa.

Apasam “OK”.

Bitstreamul a fost creat cu success.

**JUSTIFICAREA SOLUTIEI ALESE**

Am ales aceasta solutie deoarece, prin intermediul informatiei primite, aceasta varianta de rezolvare a fost prima optiune l-a care m-am gandit. Pe parcursul proiectului am intampinat dificultati, dar am reusit sa gasesc solutiile potrivite pentru a ajunge la scopul final. Probabil nu este cea mai optima rezolvare a proiectului, dar este functionala. In concluzie, solutia realizata este functionala, dar poate fi modificata intr-o varianta simplificata.

**POSIBILITATI DE DEZVOLTARE ULTERIOARE**

Din punct de vedere al dezvoltării ulterioare, pot fi luate in considerare mai multe aspecte:

1. Adaugarea orei in afisarea pe display (XX:XX:XX). Astfel, timpul afisat poate fi folosit ca un ceas uzual.
2. Adaugarea unui buton, utilizat numai pentru resetare, pentru simplificarea proiectului si a modului de functionare.
3. Adaugarea unui buton utilizat pentru incrementarea orelor, daca este realizata dezvoltarea de la subpunctul 1.
4. Adaugarea unei alarme cand numararea este crescatoare. De exemplu, daca se doreste aprinderea becului dupa 99 de minute sa nu fie nevoie ca utilizatorul sa apese butonul de incrementare a minutelor de 99 de ori, doar sa astepte dupa ce apasa START/STOP pana ajunge numaratoarea la 99 de minute.