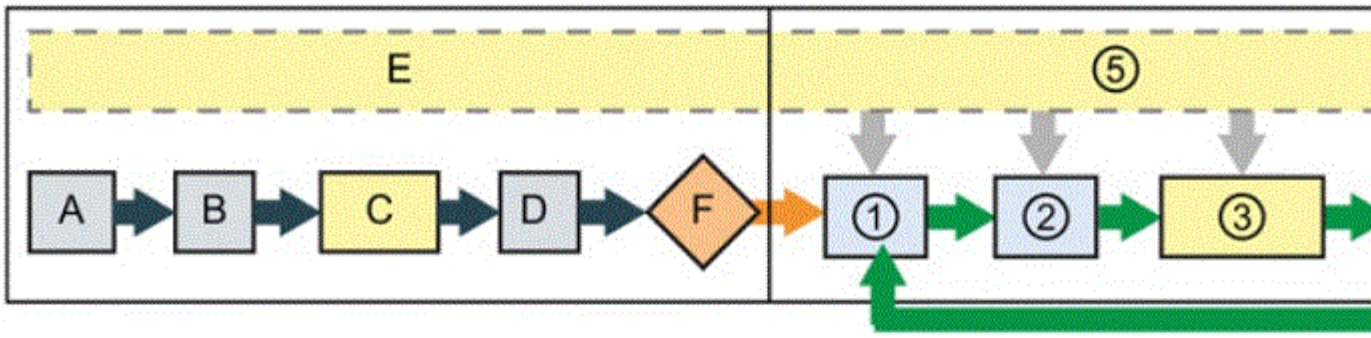
Programarea PLC-urilor SIMATIC S7-1200/1500

- **Obiective**
  - Prezentarea limbajului SCL (Structured Control Language) pentru programarea SIMATIC S7-1200/1500, mod de utilizare, particularitati
  - Prezentarea modului de realizarea a functiilor
  - Prezentarea instructiunilor decizionale, repetitive si realizarea de aplicatii cu aceste instructiuni
  - Prezentarea si exemplificarea facilitatilor de programarea in realizarea aplicatiilor SCADA
- **Organizarea sarcinilor de lucru**
  - Parcurgeti cele trei capitole ale cursului.
  - In cadrul fiecarui capitol urmariti exemplele ilustrative si incercati sa le realizati in medul de dezvoltare "Citect".
  - Fixati principalele idei ale cursului, prezentate în rezumat.
  - Completati testul de autoevaluare.
  - Timpul de lucru pentru parcurgerea testului de autoevaluare este de 15 minute.

Aplicatiile SCADA sunt rulate in general de serverele SCADA sau de terminalele HMI. In aplicatiile Siemens SCADA aplicatiile HMI sunt dezvoltate pe baza scripturilor VB Scripts (Visual Basic Scripts). In cazul aplicatiilor SCADA care necesita pe langa monitorizarea diverselor procese, comanda si controlul acestora, pentru a creste performanta intregului sistem SCADA, functiile de comanda si control ale proceselor trebuie transferate spre PLC-uri(Programmable Logic Controller).



STARTUP

A	Clears the I (image) memory area
B	Initializes the Q output (image) memory area with either zero, the last value, or the substitute value, as configured, and zeroes PB, PN, and AS-i outputs
C	Initializes non-retentive M memory and data blocks to their initial value and enables configured cyclic interrupt and time of day events. Executes the startup OBs.
D	Copies the state of the physical inputs to I memory
E	Stores any interrupt events into the queue to be processed after entering RUN mode
F	Enables the writing of Q memory to the physical outputs

RUN

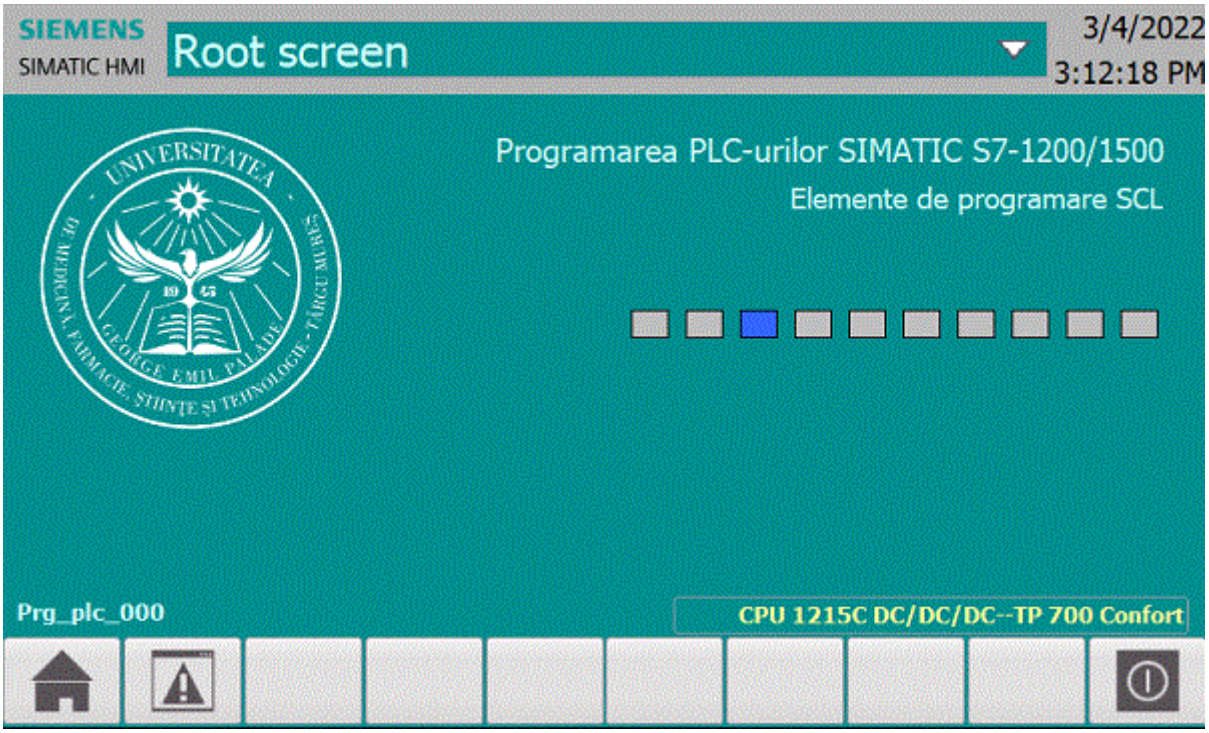
①	Writes Q memory to the physic
②	Copies the state of the physical memory
③	Executes the program cycle OB
④	Performs self-test diagnostics
⑤	Processes interrupts and comm during any part of the scan cycl

Mediul de dezvoltare SCADA TIA-Portal include pe langa instrumentele de dezvoltare HMI-uri, elemente pentru programarea PLC-urilor de tipul SIMATIC S7-1200/1500.

1.Elemente de programare SCL

SCL (Structured Control Language) inclus TIA Portal este un limbaj de programare de nivel inalt bazat pe limbajul de programare Pascal.

Vom dezvolta in continuare proiectul numit **Prg_plc_000** in cadrul caruia vom realiza diverse screen-uri in care vom folosi facilitatile de programarea PLC-urilor de tipul SIMATIC S7-1200/1500 oferite de limbajul SCL.



SCL este un limbaj bazat pe:

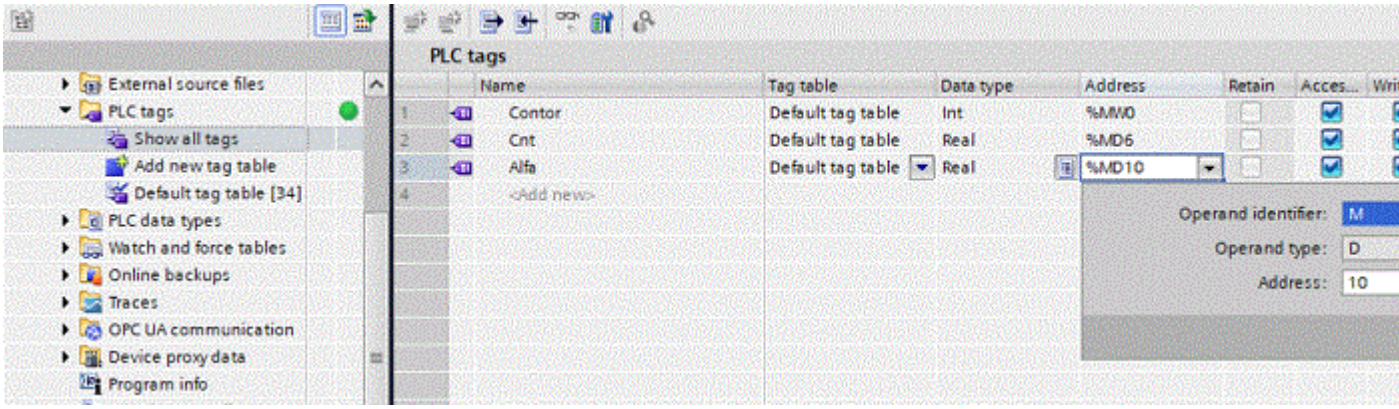
- Instructiuni care se termina cu separatorul ";"
- O instructiune se poate scrie pe mai multe randuri
- Nu este case sensitiv

Comentarii

// Pentru o singura linie de comentariu
(*
Mai multe linii de comentariu
*)

Definirea variabilelor

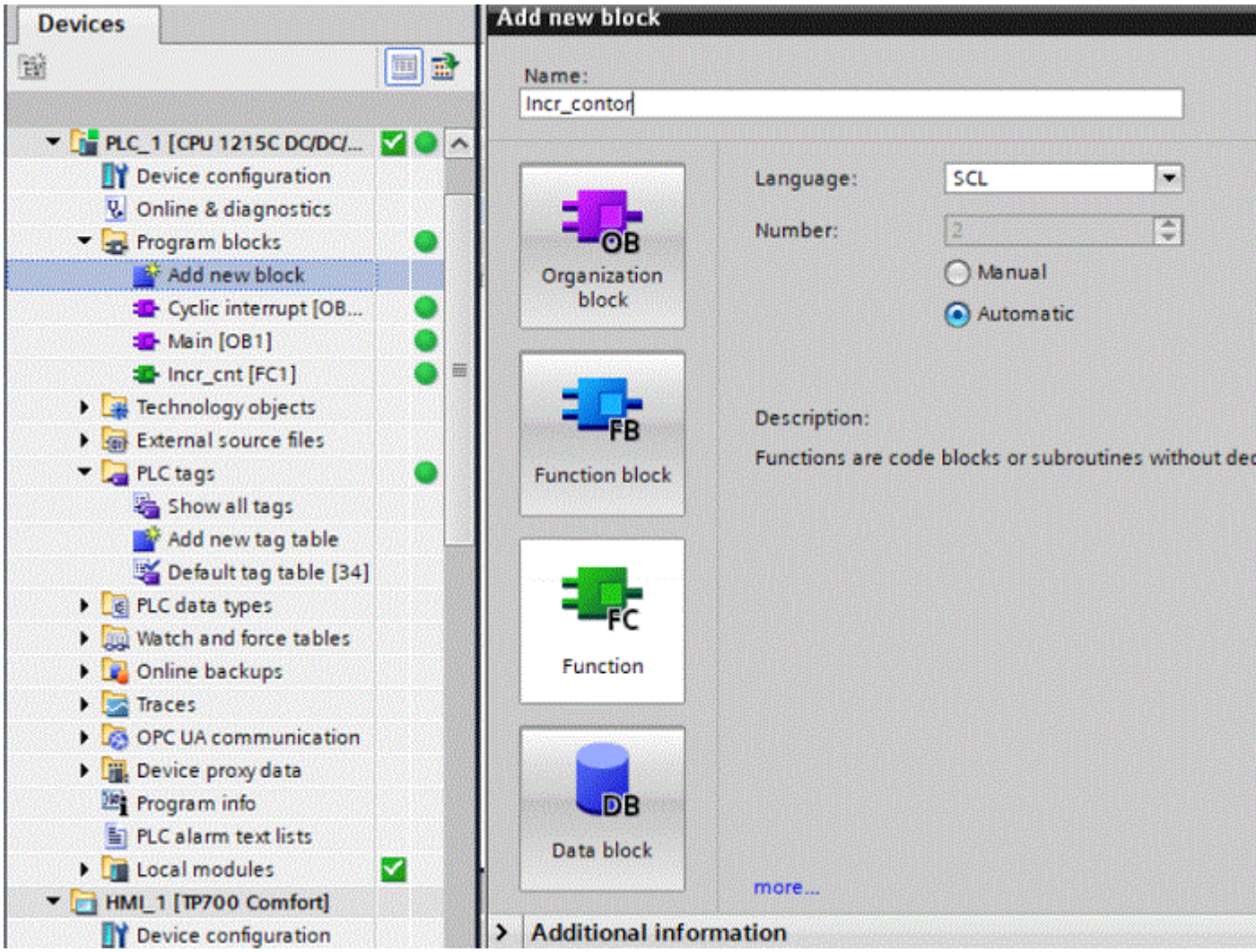
In SCL variabilele se definesc prin intermediul Tag-urilor, de tip (I sau Q) pentru variabile de intrare sau iesire si de tipul M daca e o variabila generala.



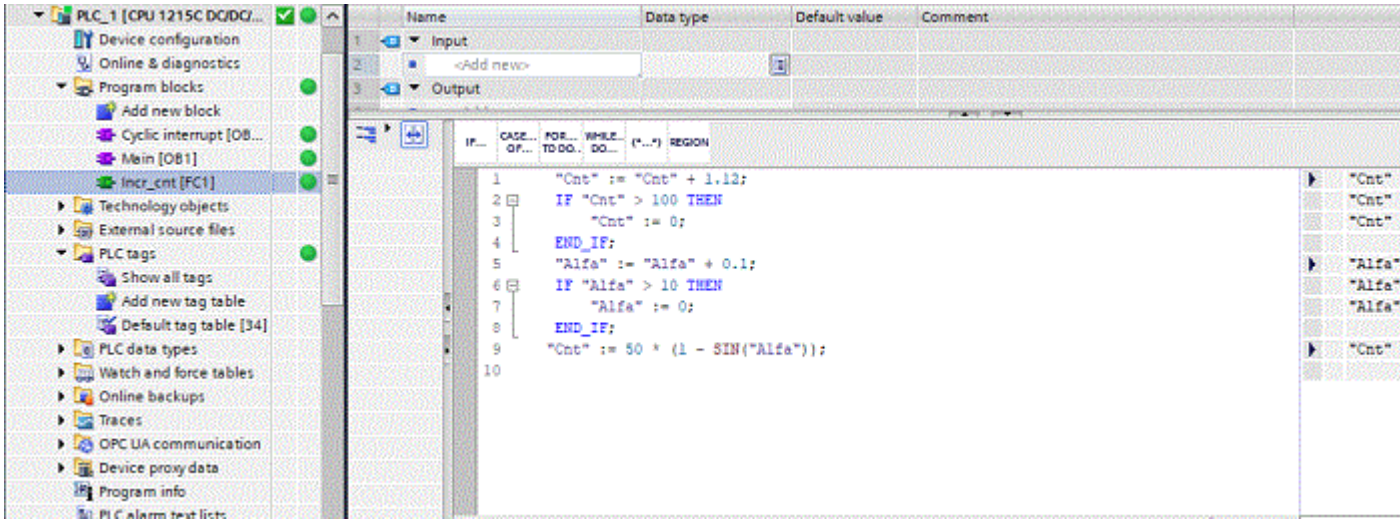
Daca e o variabila generala trebuie stabilit tipul acesteia(Int, Real..)

Crearea functiilor

Program blocks --> Add new block --> Function -->SCL



Dupa ce s-a creat functia, cu dublu click pe numele functiei, se editeaza continutul functiei.



2. Instructiuni

Instructiunea IF :

Instructiunea **IF** se foloseste pentru a selecta executia unei instructiuni (sau a unui grup de instructiuni) functie de valoarea logica a unei expresii relationale

Formatul instructiunii:
Instructiunea **IF** are urmatoarele formate:

IF expresie relationala **THEN**
instructiune(instructiuni)
END_IF

sau

IF expresie relationala **THEN**
instructiune(instructiuni)
ELSE
instructiune(instructiuni)
END_IF

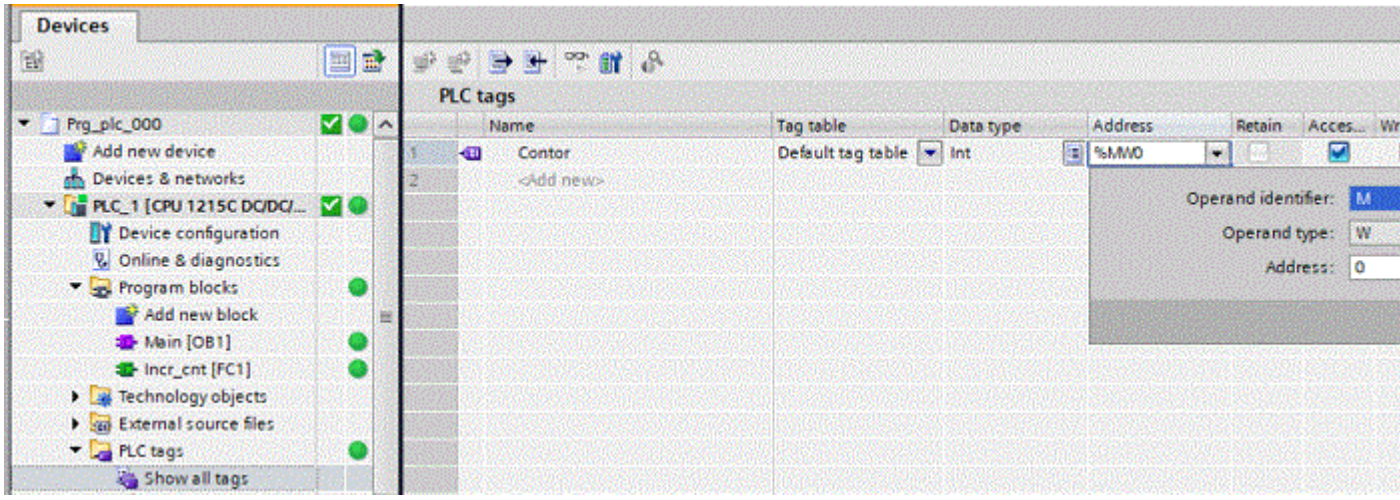
Instructiunea CASE :

CASE "variabila" **OF**
0:
declaratii;
1:
declaratii;
.
.
.

n:
declaratii;
END_CASE;

Aplicatii care utilizeaza instructiuni decizionale

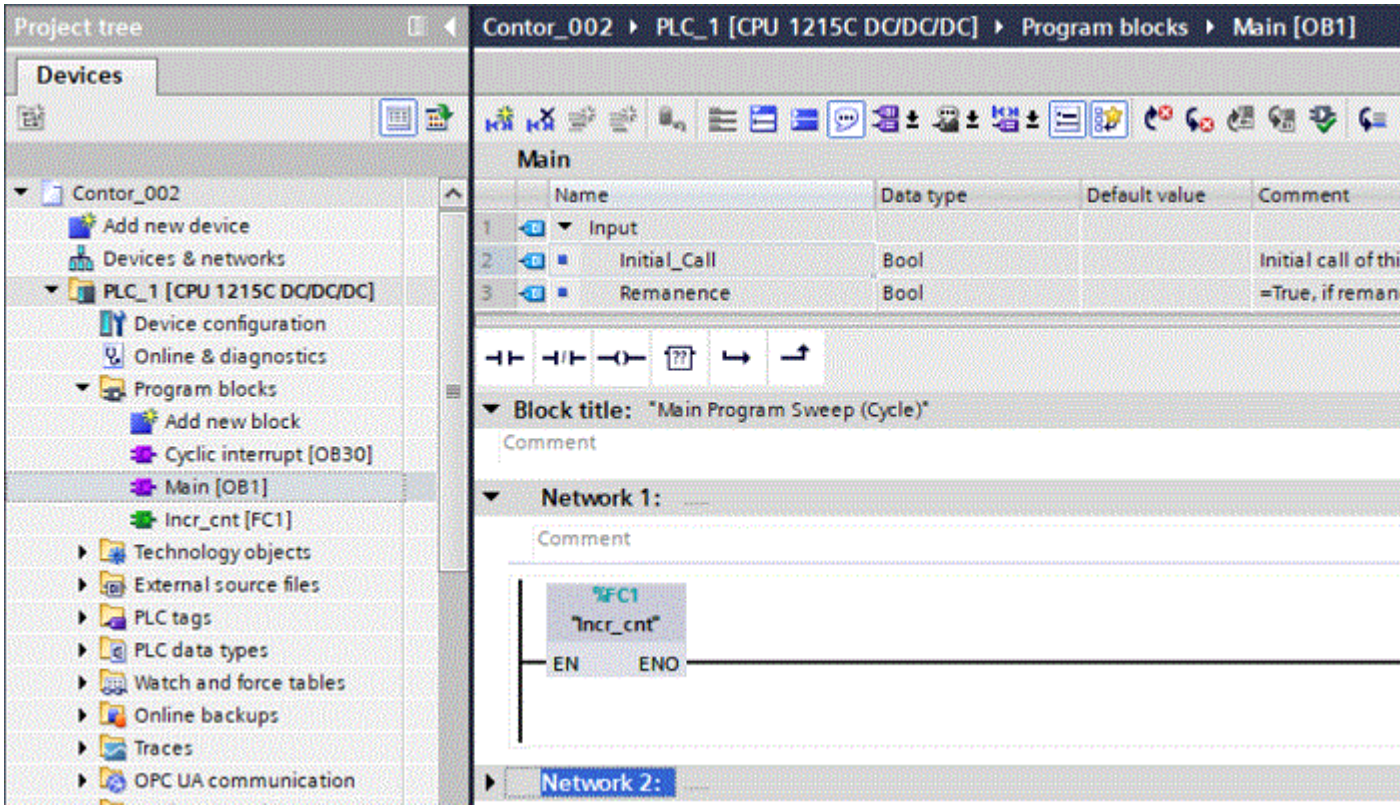
Ne propunem, in cadrul proiectului "Prg_plc_000" sa definim un PLC tag de tip int tag numit "Contor"



Vom defini in cadrul proiectului "Prg_plc_000" functia SCL in: Program blocks -->"Incr_cnt".

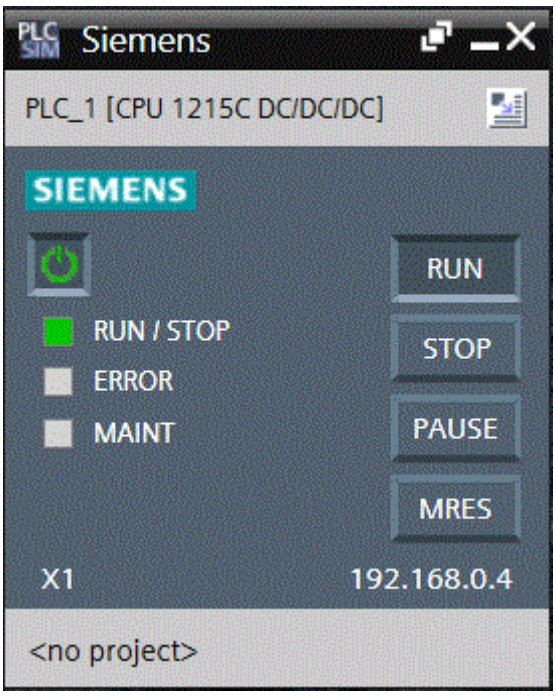
```
"Contor" := "Contor" + 1;
IF "Contor" > 100 THEN
    "Contor" := 0;
    ;
END_IF;
```

Pentru a fi lansata in executie functia "Incr_cnt", trebuie apelata din functia "Main" functie cu care se incepe rularea programului in PLC. Includerea apelului functiei "Incr_cnt" in funcata "Main" se face prin operatia de Click pe functia "Incr_cnt" si Drag pe "Network1" a functiei "Main".

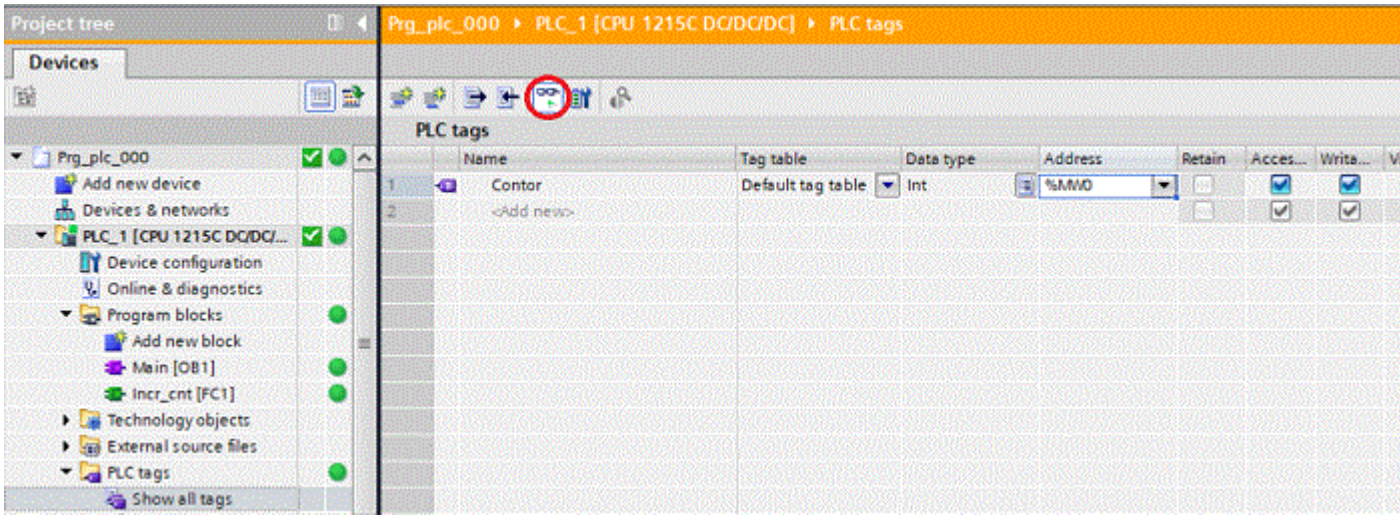


Dupa compilarea si transferarea programului in PLC (butonul "Compile" si butonul "Download to device"), functia "Main" se lanseaza automat in executie. Functia "Main" apeleaza la randul ei functia "Incr_cnt".

In cazul ca nu dispunem de PLC, rularea programului poate fi simulata din butonul "Stat simulation" din meniul principal.

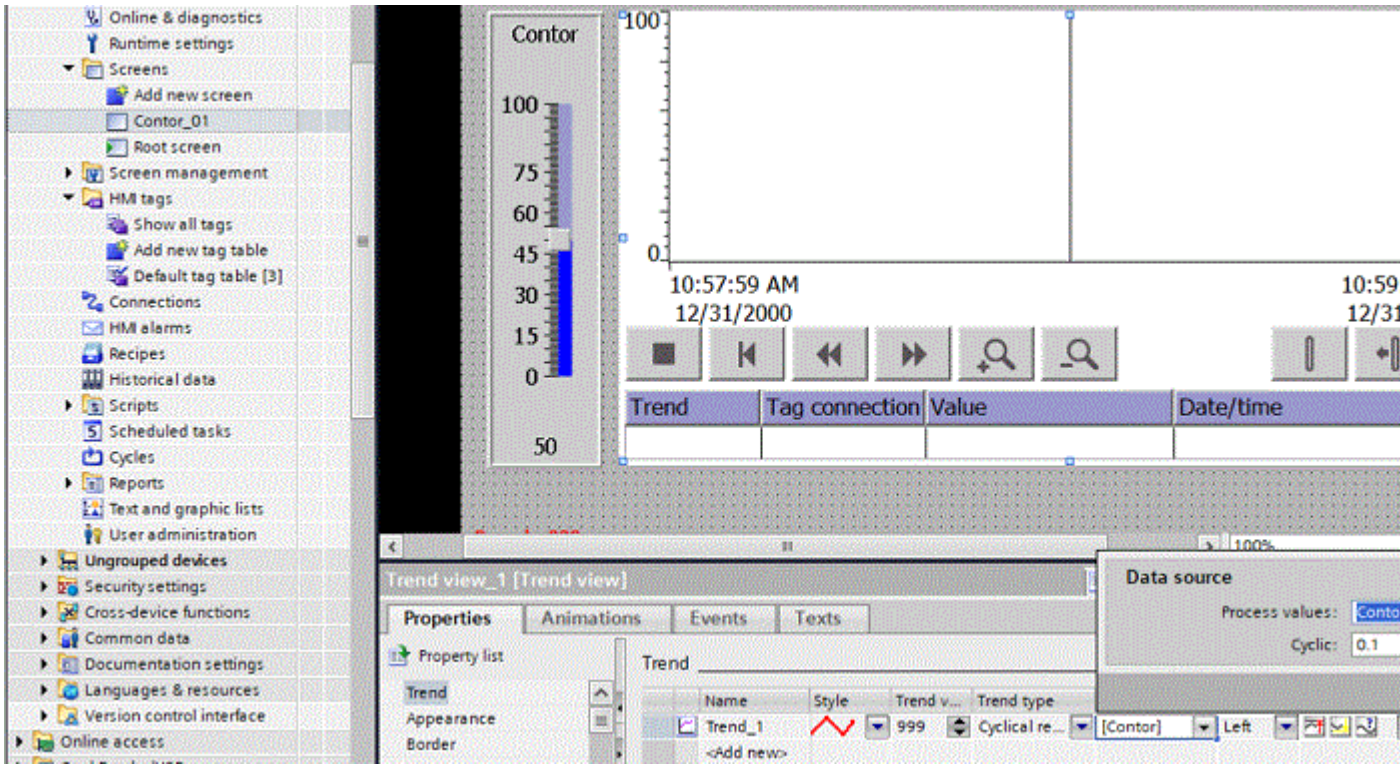


Selectand PLC Tags-->Show all tags si apasand butonul "Monitor all" (incercuit cu rosu), putem vedea in mod dinamic schimbarea valorii acestui tag.

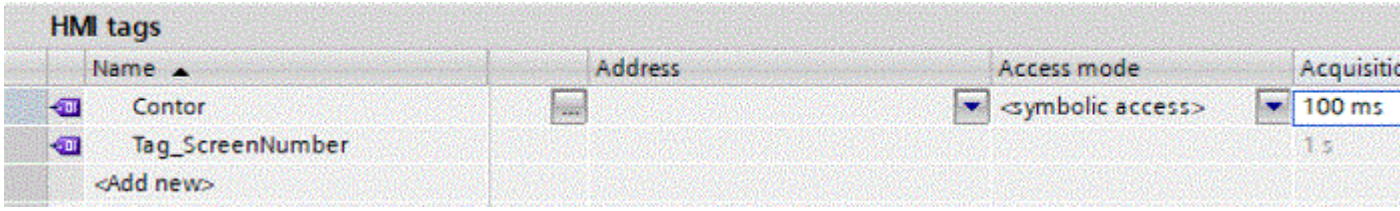


Valorile sunt afisate la interval de 0.1 secunde iar modificarea valorii acestu tag se face cu o viteza mult mai mare, deci vom vedea numere aleatoare intre 0 si 100, nu vom vedea numere afisate in ordine de la 0 la 100.

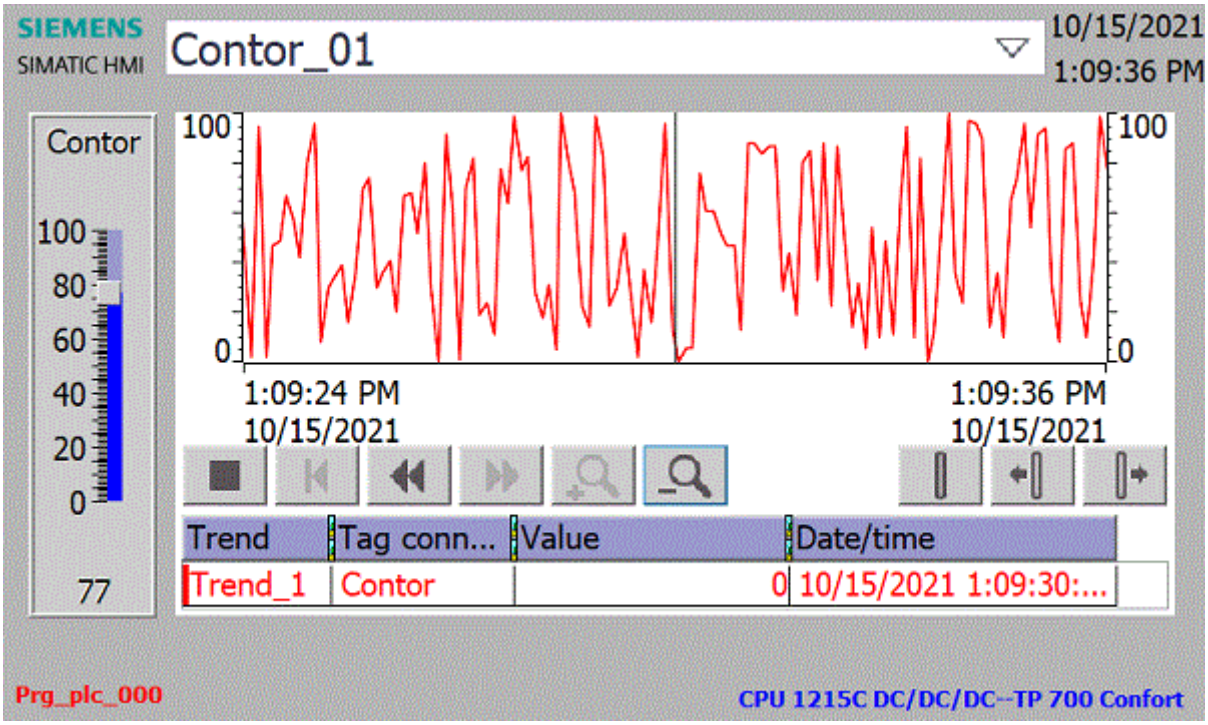
Am putea sa realizam un screen numit "Contor_01" in care sa afisam grafic valoarea contorului. Plasam pe acest screen un obiect "Trend view" si un obiect "Slider" carora le atribuim PLC tag-ul "Contor".



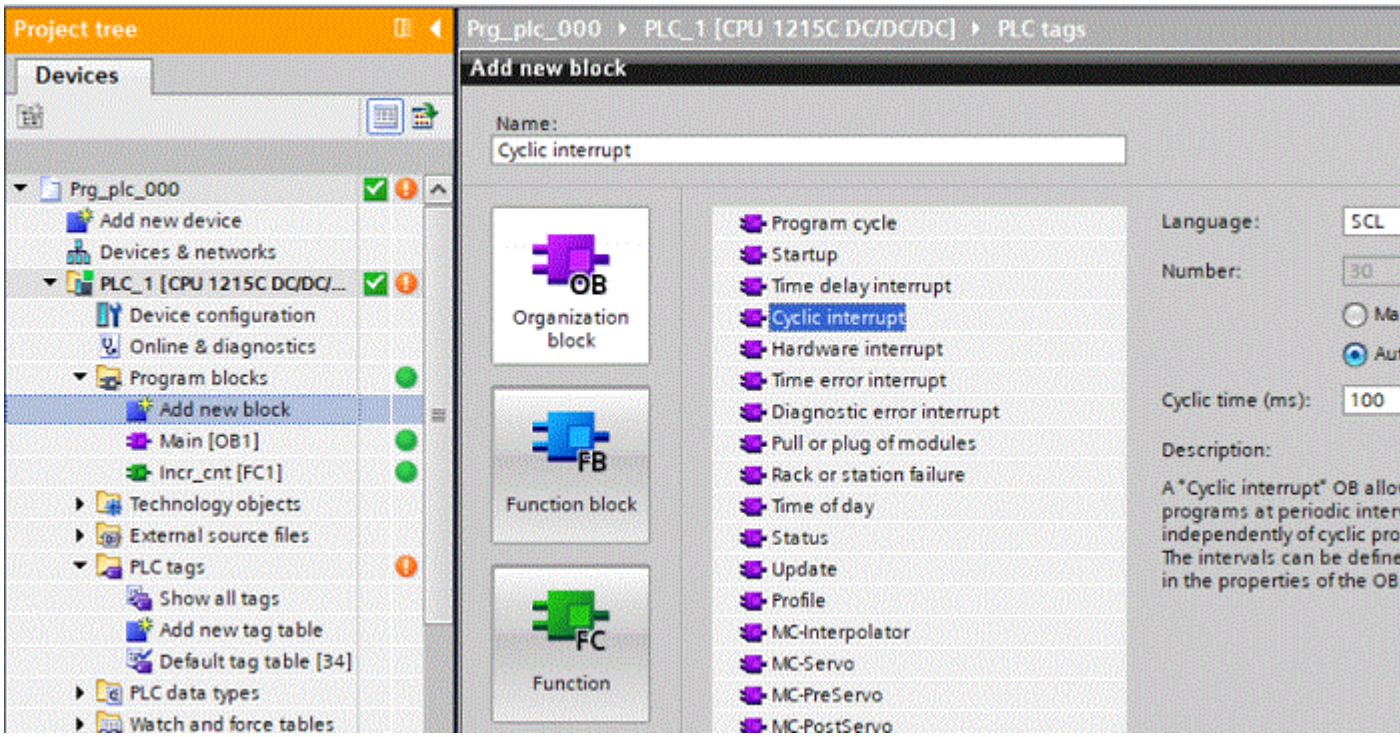
Dupa atribuirea PLC tag-ul "Contor" acestor obiecte, acest tag apare automat in HMI tags



Modificam Acquisition cyle la 100 ms, apasam "Start simulation" si obtinem:

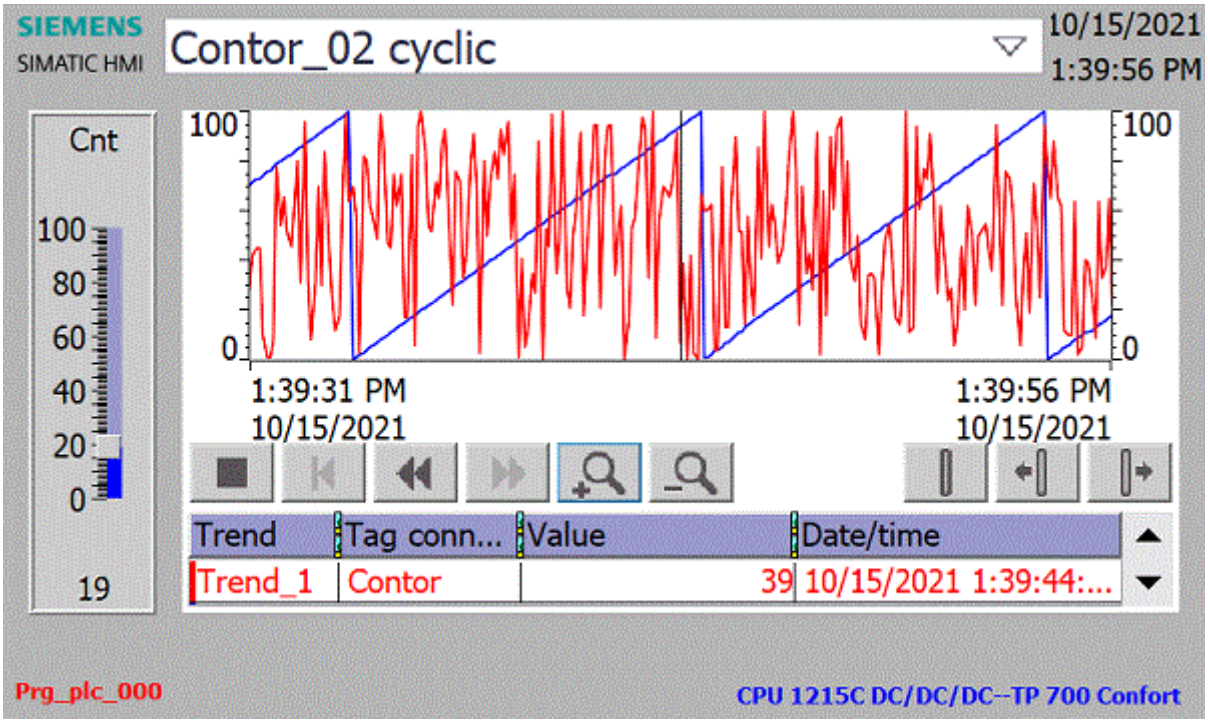


Pentru a realiza un contor care se incrementeaza numai la un anumit interval (de ex 100ms), vom crea un nou block "Cyclic interrupt" si un nou tag "Cyclic_cnt".



Continutul acestei functii fiind:

```
"Cyclic_cnt" := "Cyclic_cnt" + 1;
IF "Cyclic_cnt" > 100 THEN
    "Cyclic_cnt" := 0;
    ;
END_IF;
```

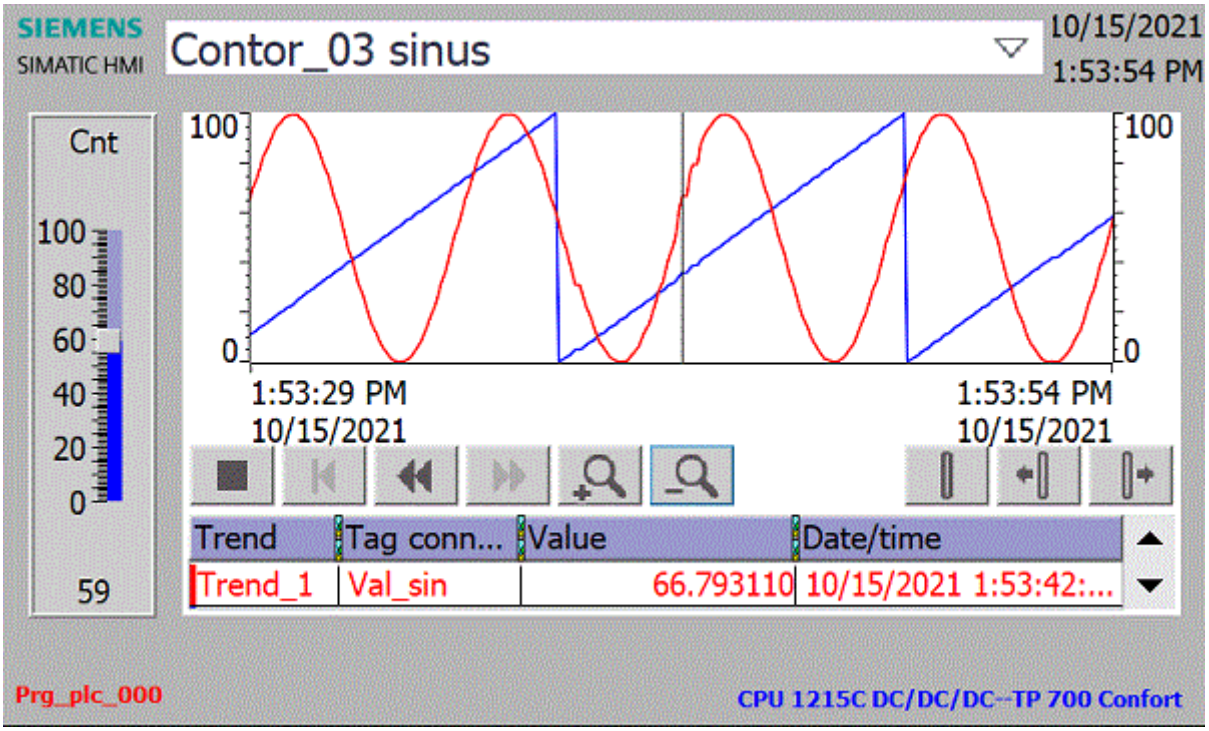


Daca introducem PLC tags: "Alfa" si "Val_sin" am putea sa afisam o forma sinusoidala. Completam functia "Cyclic interrupt" cu:

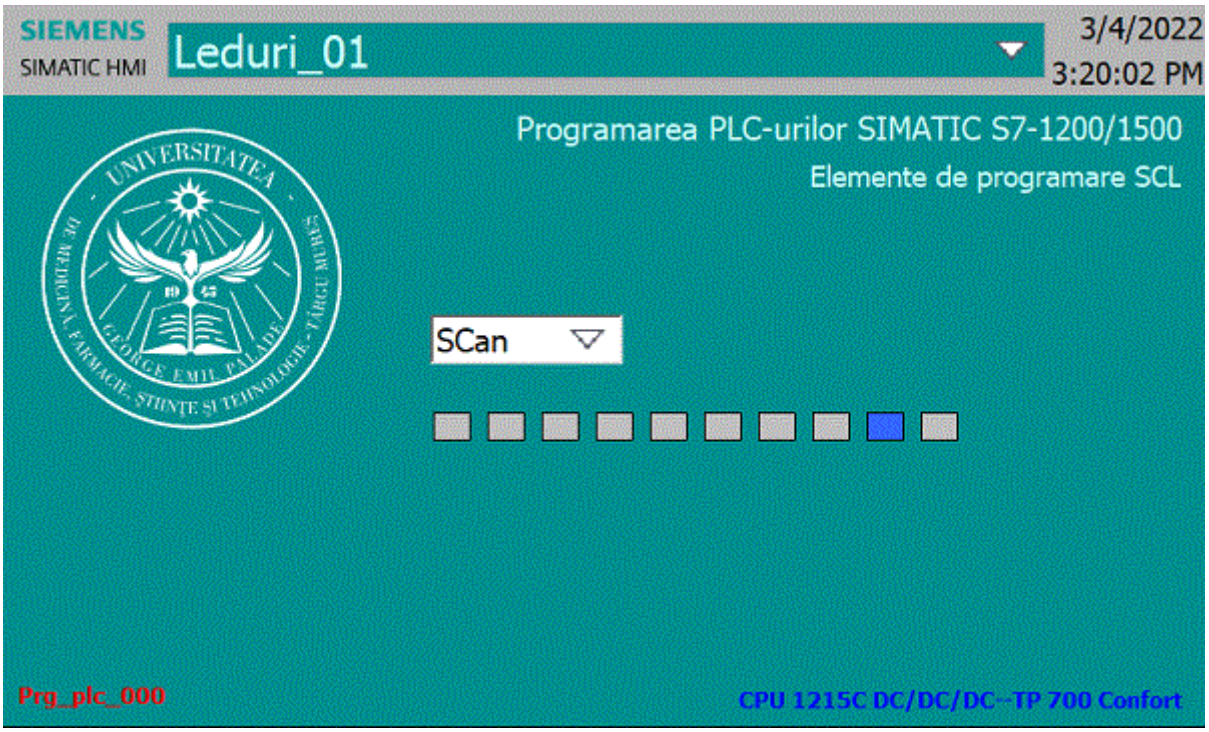
```
"Alfa" := "Alfa" + 0.1;
IF "Alfa" > 100 THEN
    "Alfa" := 0;
END_IF;

"Val_sin" := 50 * (1 - SIN("Alfa"));
```

Creem Screen-ul "Contor_03" in care atrbuim obiectului "Trend view" tag-urile "Contor" "Val_sin", rulam din nou aplicatia si obtinem:



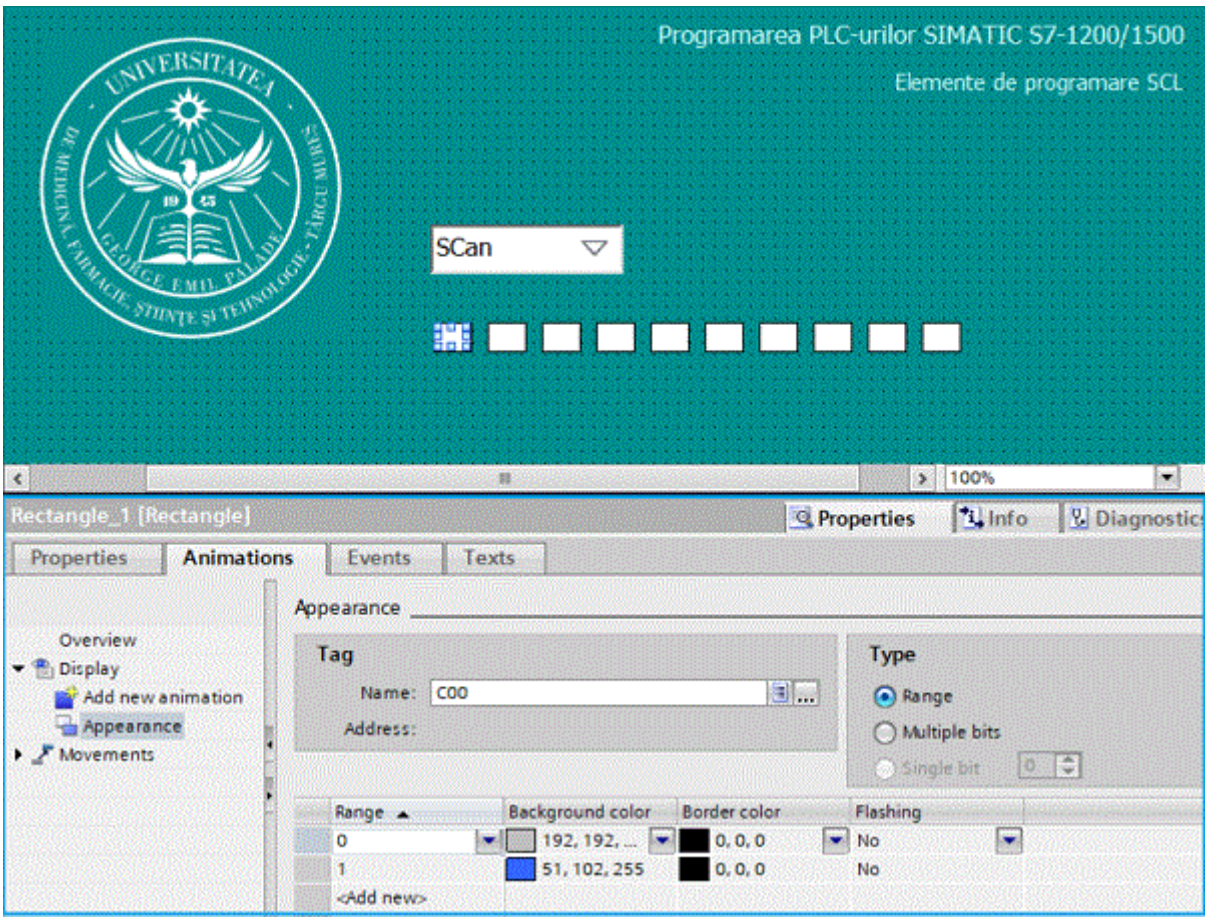
Vom folosi in continuare instructiunea CASE pentru a realiza diverse aplicatii in care vom folosi 10 LED-uri fizice plasate pe PLC cat si 10 LED-uri simulate,plasate pe Screen-ul "Leduri_01".



Vom defini deci 10 TAG-uri PLC de tip bool-ean, cu numele C00...C10 avand adresele %Q0.0...%Q0.7,%Q1.0,%Q1.1 si o variabila k de tip int.

5		C00	Default tag table	Bool	%Q0.0		
6		C01	Default tag table	Bool	%Q0.1		
7		C02	Default tag table	Bool	%Q0.2		
8		C03	Default tag table	Bool	%Q0.3		
9		C04	Default tag table	Bool	%Q0.4		
10		C05	Default tag table	Bool	%Q0.5		
11		C06	Default tag table	Bool	%Q0.6		
12		C07	Default tag table	Bool	%Q0.7		
13		C16	Default tag table	Bool	%Q1.0		
14		C17	Default tag table	Bool	%Q1.1		
15		k	Default tag table	Int	%MW6		

Pentru simularea LED-urilor, vom plasa pe Screen-ul "Leduri_01" 10 obiecte "Rectangle" cu numele "Rectangle_1"... "Rectangle_10". Fiecarui obiect "Rectangle" ii adaugam o animatie "Apperance" careia ii setam PLC Tag C01...C10

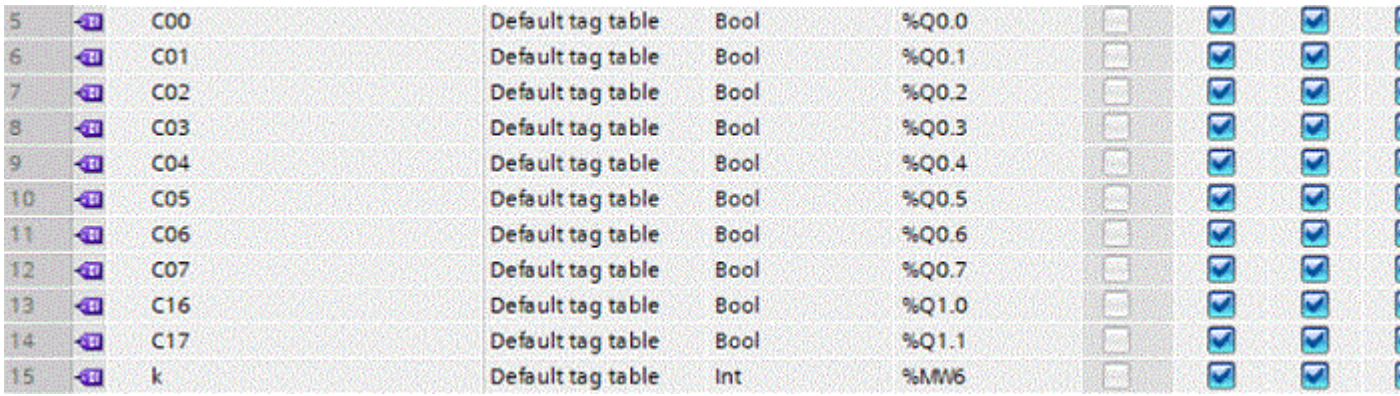


Aplictiile pe care le vom realiza in continuare, vor fi realizate in PLC prin intermediul functiilor SCL. In HMI nu vom mai realiza Script-uri, toata logica aplicatiei fiind inclusa in functiile scrise in SCL.

Prima aplicatie numita "Scan" va simula deplasarea unui singur LED de la staga la dreapta. Continutul functiei "Cyclic interrupt" va fi:

```
 "k" := "k" + 1;
IF "k" > 10 THEN
    "k" := 0;
END_IF;
CASE "k" OF
    0:
        "C17" := FALSE;
    1:
        "C17" := FALSE;
        "C00" := TRUE;
    2:
        "C00" := FALSE;
        "C01" := TRUE;
    3:
        "C01" := FALSE;
        "C02" := TRUE;
    4:
        "C02" := FALSE;
        "C03" := TRUE;
    5:
        "C03" := FALSE;
        "C04" := TRUE;
    6:
        "C04" := FALSE;
        "C05" := TRUE;
    7:
        "C05" := FALSE;
        "C06" := TRUE;
    8:
        "C06" := FALSE;
        "C07" := TRUE;
    9:
        "C07" := FALSE;
        "C16" := TRUE;
    10:
        "C16" := FALSE;
        "C17" := TRUE;
    ELSE
        "C00" := FALSE;
        "C17" := FALSE;
END_CASE;
```

Se lanseaza comanda "Download to device" dupa care se poate observa modul cum ruleaza acest program, fie prin vizualizarea PLC-tags si activarea obtiunii "Monitor all" fie prin afisarea HMI-uli "Leduri_01"

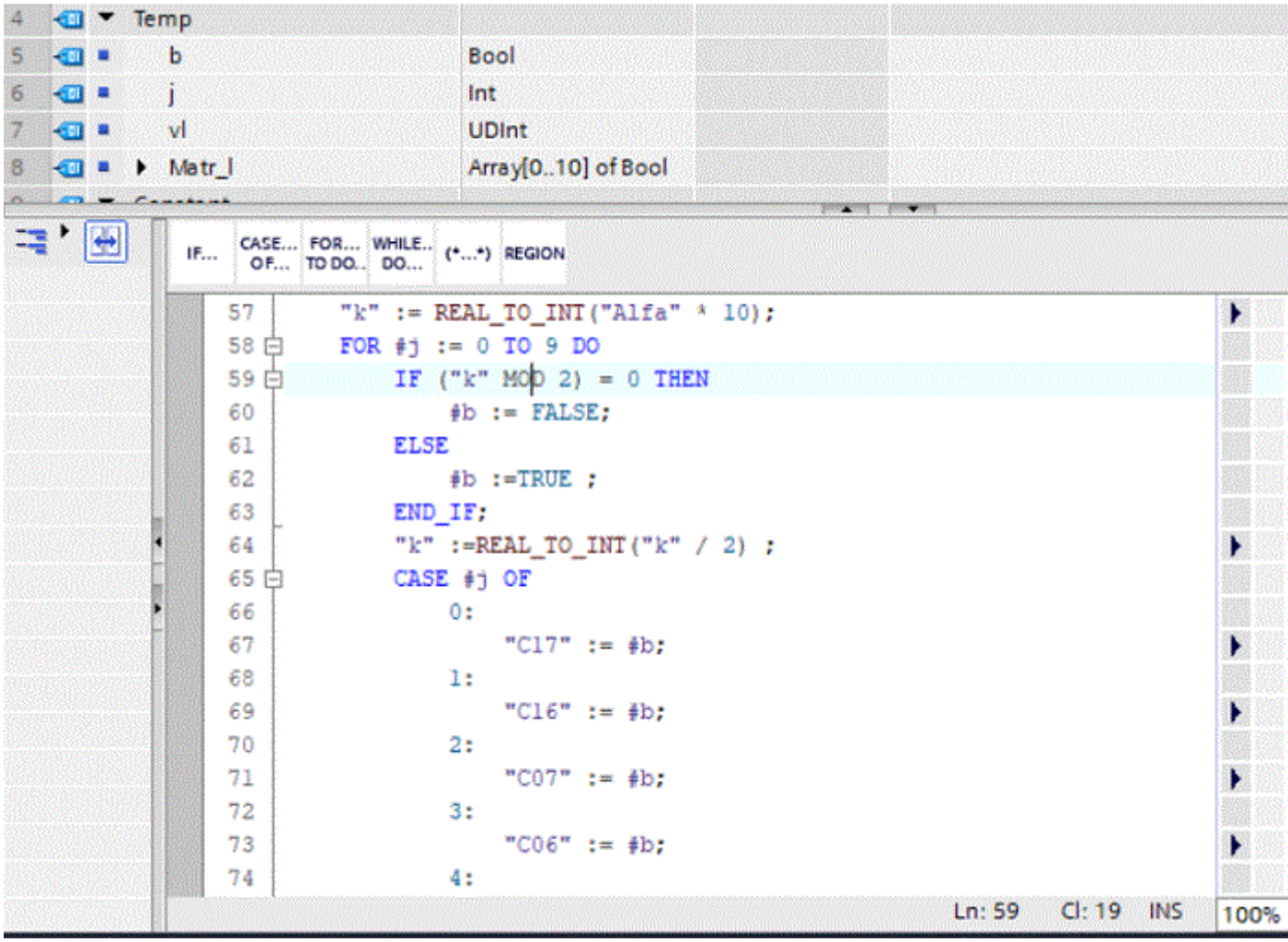


5	C00	Default tag table	Bool	%Q0.0	☐	☑	☑	
6	C01	Default tag table	Bool	%Q0.1	☐	☑	☑	
7	C02	Default tag table	Bool	%Q0.2	☐	☑	☑	
8	C03	Default tag table	Bool	%Q0.3	☐	☑	☑	
9	C04	Default tag table	Bool	%Q0.4	☐	☑	☑	
10	C05	Default tag table	Bool	%Q0.5	☐	☑	☑	
11	C06	Default tag table	Bool	%Q0.6	☐	☑	☑	
12	C07	Default tag table	Bool	%Q0.7	☐	☑	☑	
13	C16	Default tag table	Bool	%Q1.0	☐	☑	☑	
14	C17	Default tag table	Bool	%Q1.1	☐	☑	☑	
15	k	Default tag table	Int	%MW6	☐	☑	☑	

Instructiunea FOR :

FOR contor=contor_start **TO** contor_stop **DO**
declaratii
.
.
.
declaratii
END_FOR;

Vom folosi in continuare Screen-ul "Leduri_01" cu cele 10 LED-uri pentru a afisa PLC tag-ul k in format binar pe cele 10 LED-uri. Pentru a realiza conversia int->binar se vor face operatii repetate de impartire cu 2 si vom folosi deci instructiunea repetitiva FOR. Avem nevoie de variabilele temporare j si b pe care le vom defini in sectiunea "temp".

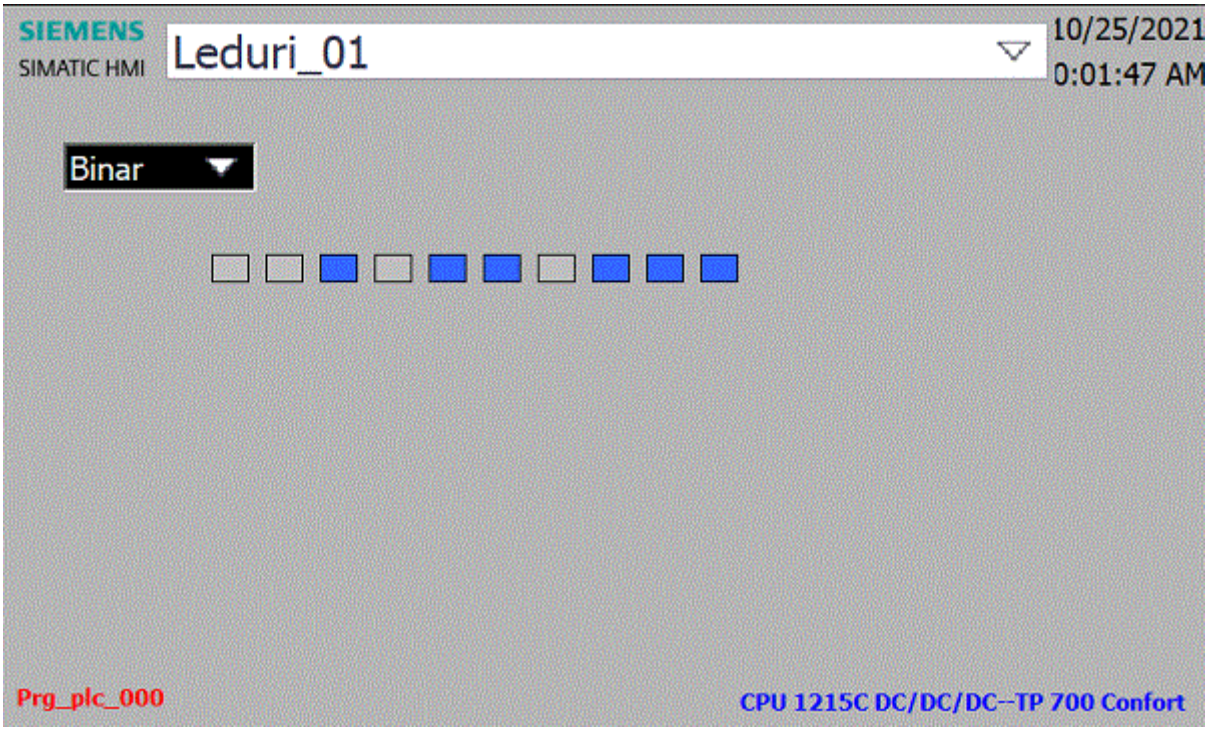


Apelarea tag-urilor in program se face folosind " " iar apelarea variabilelor temporare se face folosind simbolul # astfel ontinutul functiei "Cyclic interrupt" fiind:

```
"k" := REAL_TO_INT("Alfa" * 10);
FOR #j := 0 TO 9 DO
    IF ("k" MOD 2) = 0 THEN
        #b := FALSE;
    ELSE
        #b :=TRUE ;
    END_IF;
    "k" :=REAL_TO_INT("k" / 2) ;
    CASE #j OF
        0:
            "C17" := #b;
        1:
            "C16" := #b;
        2:
            "C07" := #b;
        3:
            "C06" := #b;
        4:
            "C05" := #b;
        5:
            "C04" := #b;
        6:
            "C03" := #b;
        7:
            "C02" := #b;
        8:
            "C01" := #b;
        9:
            "C00" := #b;
        ELSE
            ;
    END_CASE;

    ;
END_FOR;
```

Se lanseaza comanda "Download to device" dupa care se poate observa modul cum ruleaza acest program, fie prin vizualizarea PLC-tags si activarea obtiunii "Monitor all" fie prin afisarea HMI-uli "Leduri_01"



Instructiunea WHILE :

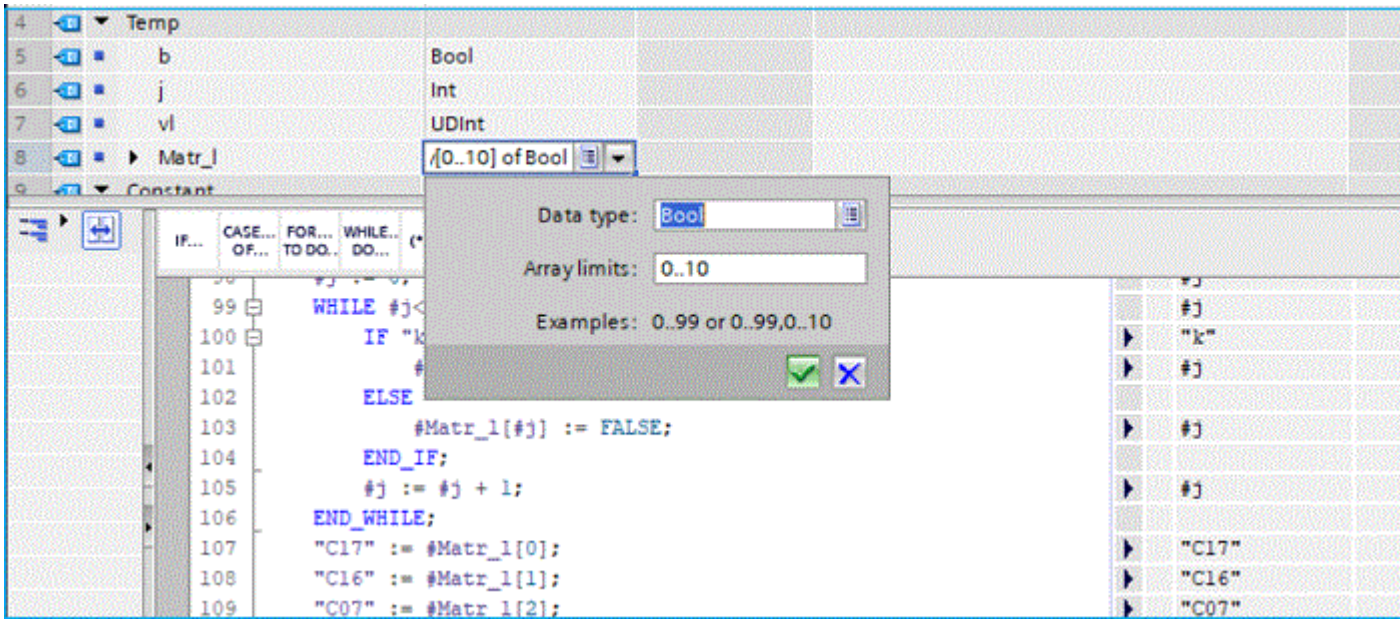
WHILE expresie relationala **DO**
declaratii;
.
.
.
declaratii;
END_WHILE;

Aplicatia anterioara poate fi rescrisa folosind instructiunea WHILE astfel:

```
"k" := "k" + 1;
IF "k" > 1000 THEN
    "k" := 0;
END_IF;
#j := 0;
WHILE #j
```

Definirea si utilizarea tablourilor

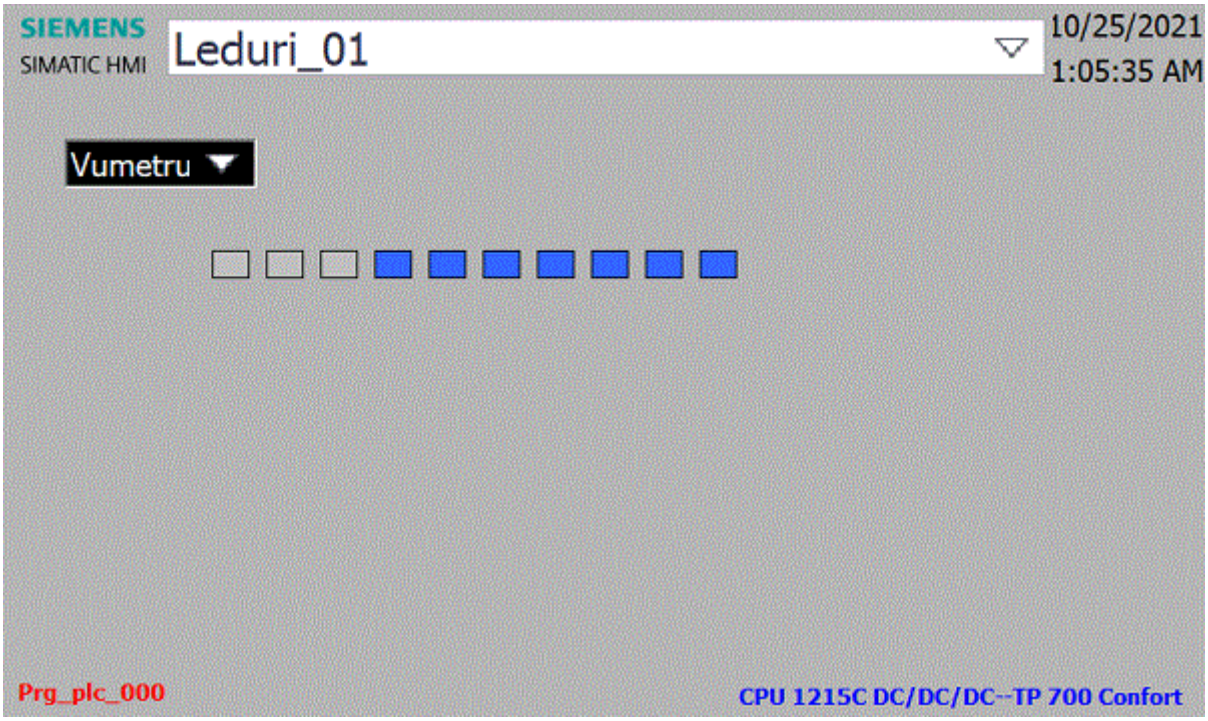
In SCL se pot defini tablouri numai ca variabile temporare. Mai jos este definita variabila "Matr_l" de tip Array of Bool.



Utilizand acest tablou, vom simula un "Vu-metru".

```
"k" :="k"+ 1;
IF"k" > 10 THEN
    "k" := 1;
END_IF;
#j := 0;
WHILE #j #j THEN
        #Matr_l[#j] := TRUE;
    ELSE
        #Matr_l[#j] := FALSE;
    END_IF;
    #j := #j + 1;
END_WHILE;
"C17" := #Matr_l[0];
"C16" := #Matr_l[1];
"C07" := #Matr_l[2];
"C06" := #Matr_l[3];
"C05" := #Matr_l[4];
"C04" := #Matr_l[5];
"C03" := #Matr_l[6];
"C02" := #Matr_l[7];
"C01" := #Matr_l[8];
"C00" := #Matr_l[9];
```

Se lanseaza comanda "Download to device" dupa care se poate observa modul cum ruleaza acest program, fie prin vizualizarea PLC-tags si activarea obtiunii "Monitor all" fie prin afisarea HMI-uli "Leduri_01"



3. Diverse aplicatii

Utilizarea marimilor digitale

In aceasta aplicatie vom utiliza:

- Comutatoarele B00 - B07 aflate pe intrarile digitale DIGITAL INPUT 0.0 - 0.7

8		B00	Default tag table	Bool	%I0.0
9		B01	Default tag table	Bool	%I0.1
10		B02	Default tag table	Bool	%I0.2
11		B03	Default tag table	Bool	%I0.3
12		B04	Default tag table	Bool	%I0.4
13		B05	Default tag table	Bool	%I0.5
14		B06	Default tag table	Bool	%I0.6
15		B07	Default tag table	Bool	%I0.7

- Comutatoarele B08 - B15 aflate pe intrarile digitale DIGITAL INPUT 1.0 - 1.7.

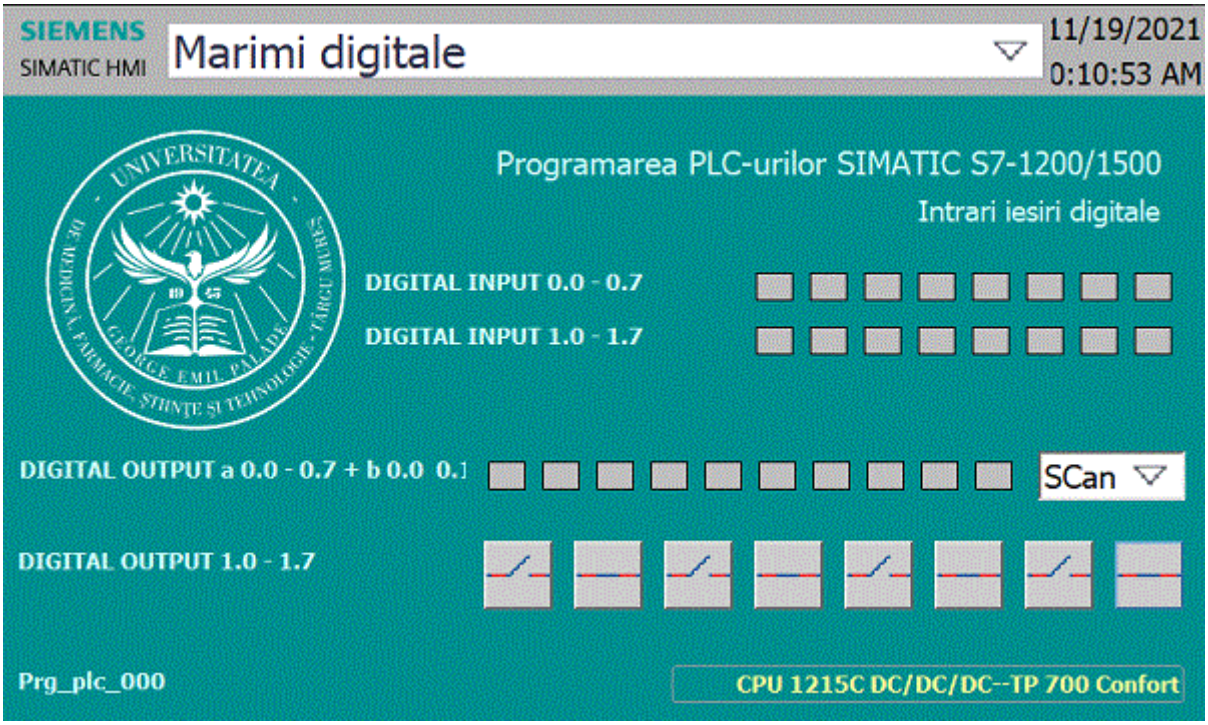
16	B08	Default tag table	Bool	%I8.0	
17	B09	Default tag table	Bool	%I8.1	
18	B10	Default tag table	Bool	%I8.2	
19	B11	Default tag table	Bool	%I8.3	
20	B12	Default tag table	Bool	%I8.4	
21	B13	Default tag table	Bool	%I8.5	
22	B14	Default tag table	Bool	%I8.6	
23	B15	Default tag table	Bool	%I8.7	

- LED-urile C00 - C07,C16,C17 aflate pe iesirile digitale DIGITAL OUTPUT a 0.0 - 0.7 + b 0.0 0.1

24	C00	Default tag table	Bool	%Q0.0	
25	C01	Default tag table	Bool	%Q0.1	
26	C02	Default tag table	Bool	%Q0.2	
27	C03	Default tag table	Bool	%Q0.3	
28	C04	Default tag table	Bool	%Q0.4	
29	C05	Default tag table	Bool	%Q0.5	
30	C06	Default tag table	Bool	%Q0.6	
31	C07	Default tag table	Bool	%Q0.7	

- LED-urile C08 - c15 aflate pe iesirile digitale DIGITAL OUTPUT 1.0 - 1.7.

32	C08	Default tag table	Bool	%Q8.0		☑
33	C09	Default tag table	Bool	%Q8.1		☑
34	C10	Default tag table	Bool	%Q8.2		☑
35	C11	Default tag table	Bool	%Q8.3		☑
36	C12	Default tag table	Bool	%Q8.4		☑
37	C13	Default tag table	Bool	%Q8.5		☑
38	C14	Default tag table	Bool	%Q8.6		☑
39	C15	Default tag table	Bool	%Q8.7		☑
40	C16	Default tag table	Bool	%Q1.0		☑
41	C17	Default tag table	Bool	%Q1.1		☑



Utilizarea marimilor analogice

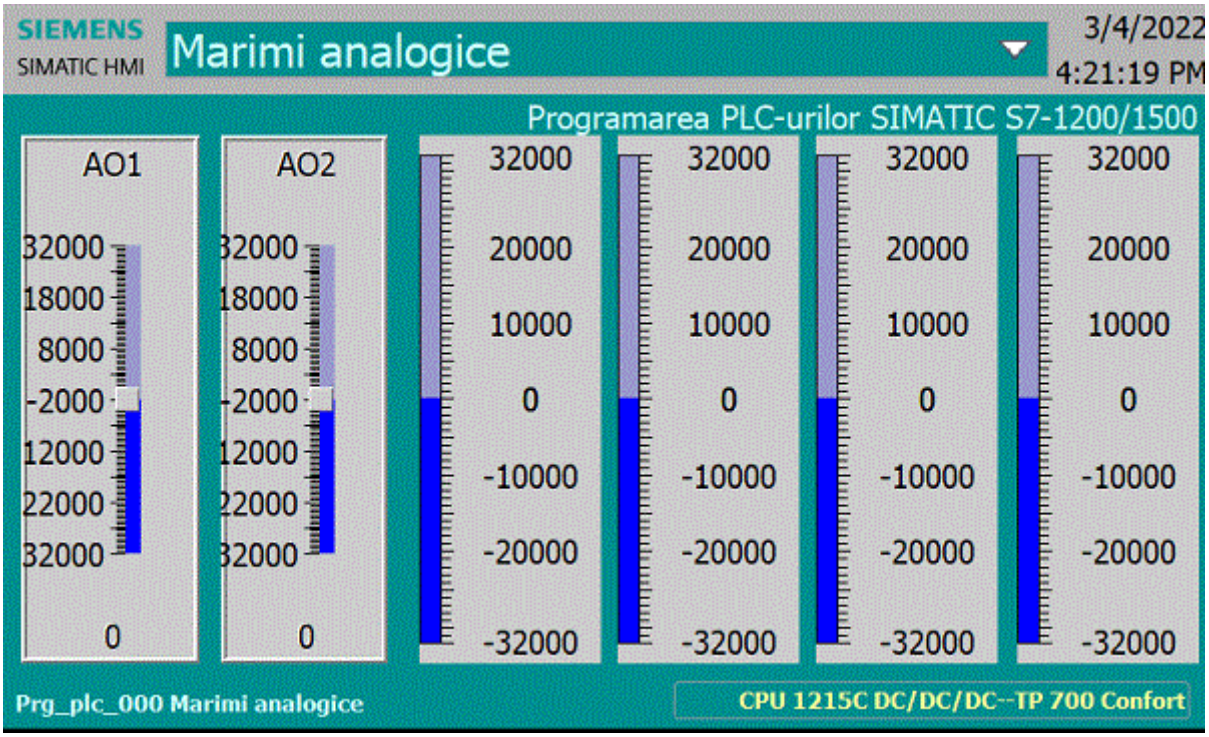
In aceasta aplicatie vom utiliza:

- Intrarile analogice AI0 - AI3 aflate pe intrarile analogice ANALOGICAL INPUT IW112 - IW118
- Iesirile analogice AO0 - AO1 aflate pe iesirile analogice ANALOGICAL OUTPUT QW64-QW66

1		AI0	Default tag table	Int	%IW112
2		AI1	Default tag table	Int	%IW114
3		AI2	Default tag table	Int	%IW116
4		AI3	Default tag table	Int	%IW118
6		AO0	Default tag table	Int	%QW64
7		AO1	Default tag table	Int	%QW66

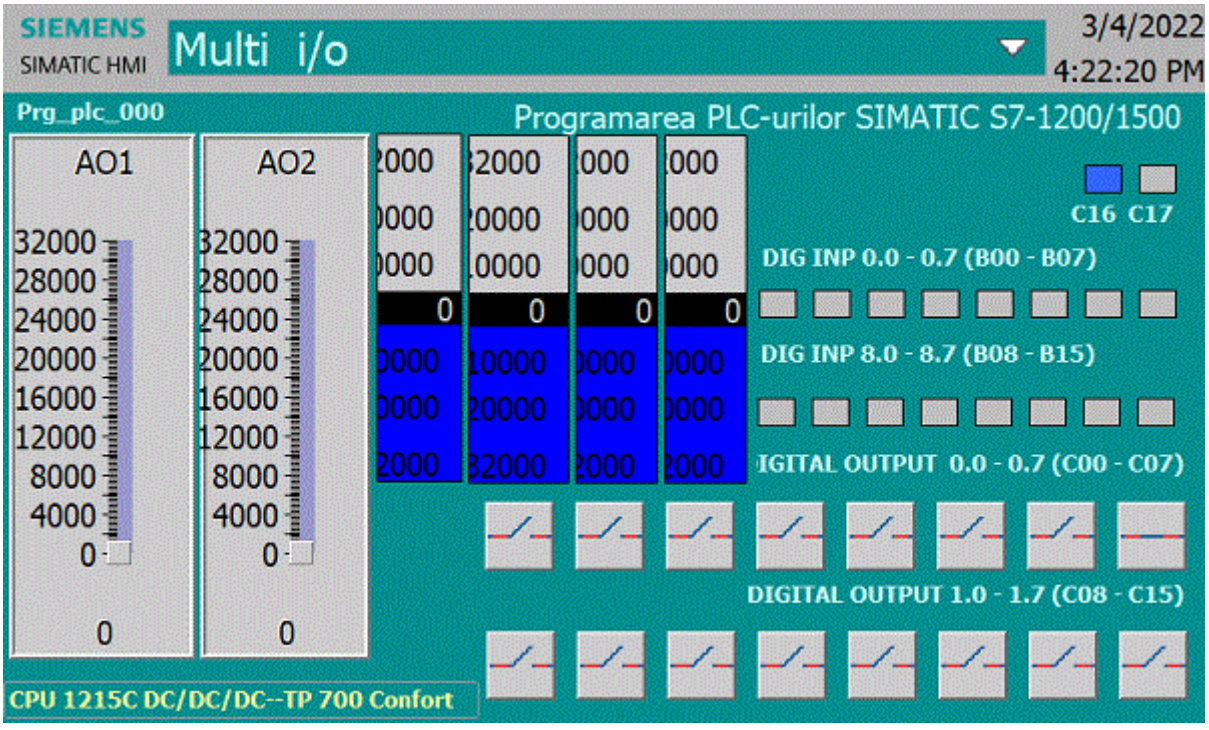
Pentru intrarile analogice in domeniul este 0 -10v, valorile corespunzatoare fiind in domeniul 0-32000

Pentru iesirile analogice in domeniul: -3200 +32000 valorile la iesire fiind in domeniul -10v +10v



Multi I/O

In aceasta aplicatie vom utiliza toate intrarile si iesirile:



Rezumat

- ## SCL (Structured Control Language)

Instructiunea IF :

Instructiunea **IF** se foloseste pentru a selecta executia unei instructiuni (sau a unui grup de instructiuni) functie de valoarea logica a unei expresii relationale

Formatul instructiunii:
Instructiunea **IF** are urmatoarele formate:

IF expresie relationala **THEN**
instructiune(instructiuni)
END_IF

sau

IF expresie relationala **THEN**
instructiune(instructiuni)
ELSE
instructiune(instructiuni)
END_IF

Instructiunea CASE :

CASE "variabila" **OF**
0:
declaratii;
1:
declaratii;
.
.
.
n:
declaratii;
END_CASE;

Instructiunea FOR :

FOR contor=contor_start **TO** contor_stop **DO**
declaratii
.
.
.
declaratii
END_FOR;

Instructiunea WHILE :

WHILE expresie relationala **DO**
declaratii;

.
.
.
declaratii;
END_WHILE;

- ## Rezultate asteptate

Dupa studierea acestui modul, ar trebui sa cunoasteti:

- Cum sa scrieti VB scripts
- Sa utilizati instructiuni decizionale si repetitive
- Sa realizati aplicatii SCADA in care sa folositi elemente de programare SCL

- ## Termeni esentiali

Termen	Descriere
SCADA	Supervisory Control And Data Aquisition
Tag	Nume generic pentru elementele din procesul monitorizat codificate prin intermediul variabilelor
HMI	Human Machine Interface -Interfata dintre aplicatie si utilizator
SCL (Structured Control Language)	Limbaj pentru programarea PLC-urilor de tipul SIMATIC S7-1200/1500
Instructiuni decizionale	Instructiuni care permit alegerea setului de instructiuni care urmeaza a fi executate in functie de o expresie relationala
Instructiuni repetitive	Instructiuni care permit rularea repetitiva functie de o expresie relationala, a unui setului de instructiuni
Expresie relationala	Expresie a carui rezultat este o valoare logica

- ## Recomandari bibliografice

- [1] T. Turc - Sisteme SCADA, Ed. Univ. "Petru Maior", ISBN: 978-606-581-110-2 , 2013
- [2] T. Turc - Aplicatii SCADA, Ed. Univ. " Petru Maior", ISBN: 978-606-581-109-6 , 2013
- [3] T. Turc - Programarea microprocesoarelor din familia X86:, Ed. Univ. "Petru Maior", ISBN: 978-606-581-026-6, 2011
- [4] T. Turc - Tehnologii WEB:, Ed. Univ. "Petru Maior", ISBN: 978-973-755-576-2, 2010
- [5] T. Turc - Informatica aplicata in ingineria electrica, ISBN: 978-973-169-700-0, Ed. univ. UMFST, Tg. Mures, 2021.
- [6] T. Turc - Programare avansata C++ pentru ingineria electrica, ISBN: 978-973-755-588-5, Ed. MatrixRom, 2009.
- [7] T. Turc - Elemente de programare C++ utile in ingineria electrica, ISBN: 978-973-755-576-2, Ed. MatrixRom, 2009
- [8] B. Barbat - Informatica industriala - Programarea în timp real – Institutul Central pentru Conducere si informatica 1984
- [9] I. Babuita – Conducerea automata a proceselor – Ed. Facla 1985

- **Link-uri utile**
  - https://support.industry.siemens.com/cs/ww/en/view/109755216 - SIMATIC WinCC V15.1 - Programming reference - 2021 -
  - https://support.industry.siemens.com/cs/us/en/view/109755202 - STEP 7 and SIMATIC WinCC V15.1 System Manual - 2021 -
  - https://support.industry.siemens.com/cs/ww/en/view/81318674 - Programming for SIMATIC S7-1200 and S7-1500 - 2021 -
  - https://support.industry.siemens.com/cs/document/39710145 - SIMATIC S7-1200 Easy Book - 2021 -
  - https://support.industry.siemens.com/cs/ww/en/view/68011496 - Creating and using user-defined web pages on S7-1200 / S7-1500 -
  - S7-1200_1500_Webserver_DOC_v4_en.pdf - Creating user-defined web pages for S7-1200 / S7-1500 - 2021 -

Test de evaluare

- -Marcati raspunsurile corecte la intrebarile urmatoare.
- -ATENTIE: pot exista unul, niciunul sau mai multe raspunsuri corecte la aceeasi intrebare.
- -Timp de lucru: 10 minute