

Programarea aplicațiilor SCADA

Cuprins

Programarea aplicațiilor SCADA	1
Obiective	1
Organizarea sarcinilor de lucru	1
1. Limbajul de programare Cicode	2
Set simbolul on/off	3
Funcții predefinite	4
Seturile simbol multi-state	6
Tablouri de elemente	8
2. Instrucțiuni decizionale	10
Aplicații care utilizează instrucțiuni decizionale	10
3. Funcții	13
Aplicații care utilizează funcții definite de utilizator	14
Test de autoevaluare	17
Rezumat	18
Rezultate așteptate	20
Termeni esențiali	20
Recomandări bibliografice	21
Link-uri utile	21
Test de evaluare	22

Obiective

- Prezentarea limbajului de programare Cicode, mod de utilizare, particularități
- Prezentarea instrucțiunilor decizionale și realizarea de aplicații cu aceste instrucțiuni
- Prezentarea modului de utilizare al funcțiilor și realizarea de aplicații care folosesc funcții
- Prezentarea și exemplificarea facilităților de programarea în realizarea aplicațiilor SCADA

Organizarea sarcinilor de lucru

- Parcurgeți cele trei capitole ale cursului.
- În cadrul fiecărui capitol urmăriți exemplele ilustrative și încercați să le realizați în medul de dezvoltare "Citect".
- Fixați principalele idei ale cursului, prezentate în rezumat.
- Completați testul de autoevaluare.
- Timpul de lucru pentru parcurgerea testului de autoevaluare este de 15 minute.

1.Limbajul de programare Cicode

Aplicațiile SCADA simple se pot crea în principiu fără a utiliza noțiuni de programare. Aplicațiile mai complexe nu pot fi realizate decât utilizând facilitățile oferite de limbajul de programare Cicode.

Cicode este un limbaj de programare integrat în mediul de dezvoltare CitectSCADA destinat dezvoltării de aplicații SCADA, permițând controlul software a elementelor utilizate pentru mimarea și controlul proceselor. Este un limbaj structurat similar cu "C" sau Visual Basic.

Vom realiza un nou proiect numit "**Sch_el**" urmând pașii descriși în cursul anterior anterior pentru a crea un nou proiect, sau se poate utiliza proiectul "**Sch_el_start**" proiect în care au fost parcurși toți pașii pentru crearea unui nou proiect.

În proiectul "**Sch_el_start**", sunt realizate deja o serie de astfel de simboluri, așa că vom porni de la acest proiect.

Acest proiect poate fi descărcat aici : [Download - "Sch_el_start"](#)

După ce s-a download-at acest fișier, din Citect Explorer->Restore se încarcă acest proiect și i se atribuie numele "**Sch_el**".

Proiectul conține pagina grafică intitulată: "**start**" afișată mai jos.

Acest proiect este punctul de plecare pentru a crea noi aplicații
Exemple de aplicații

Sunt configurate:

- Clusterul: sch_el_cluster
- Network Addresses: sch_el_adr
- Alarm Servers: sch_el_alarm
- Report Servers: sch_el_rep
- Trend Servers: sch_el_trend
- I/O Server: sch_el_io

Simboluri noi create in Library global:

k_off
k_on
rez
sursa_v
gnd
sep_on
sep_off
intr_on
intr_off



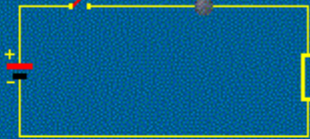






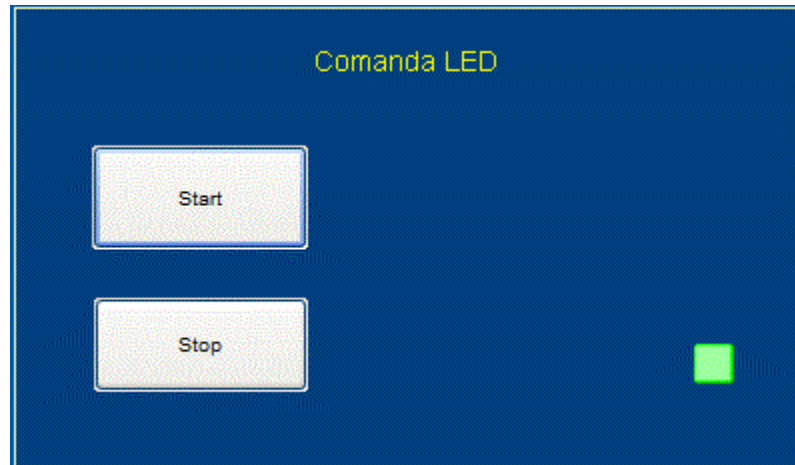


k1

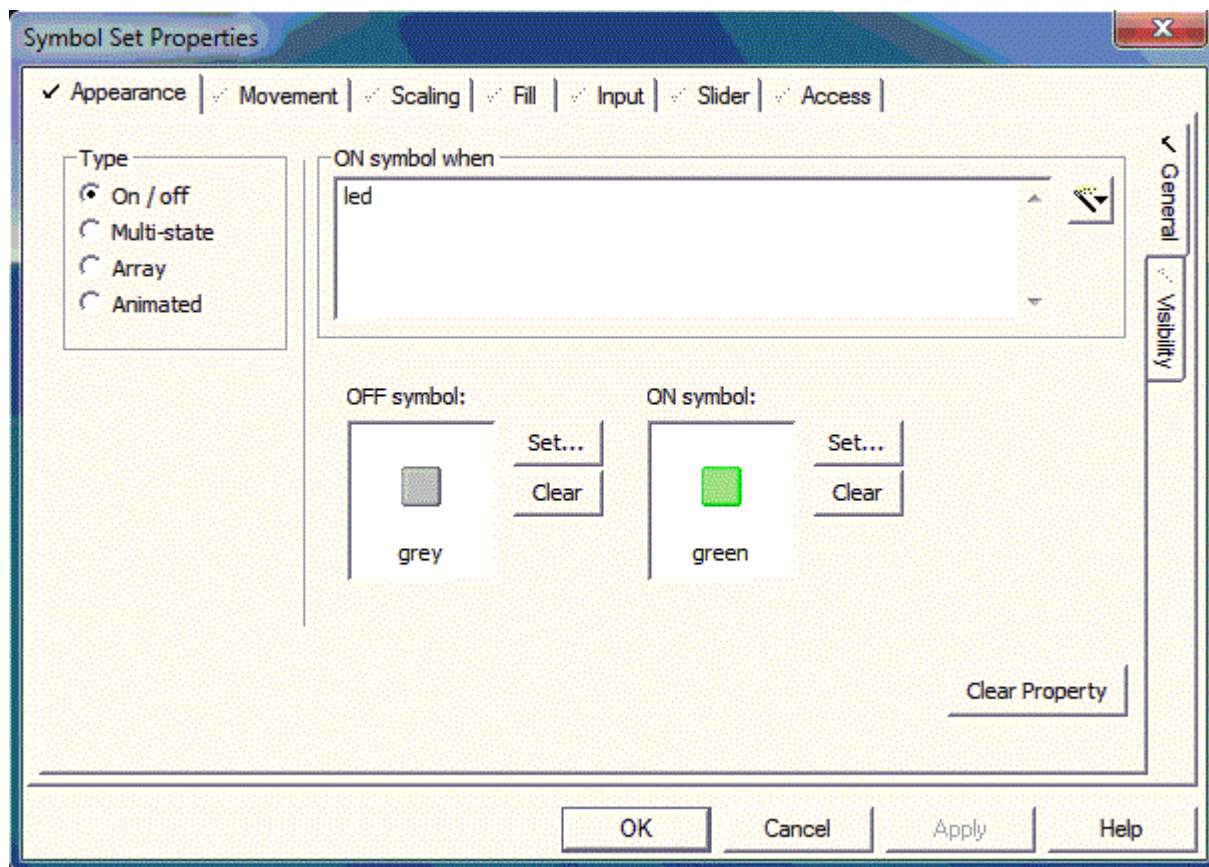


Set simbolul on/off

Vom încerca acum să creăm o pagină grafică **"led1"** pentru a simula comanda de alimentare cu energie a unui proces. Alimentarea va fi semnalizată în pagina grafică prin aprinderea unui LED. LED-ul va avea două stări. Starea stins de culoare gri și starea aprins, de culoare verde.



Pentru a putea afișa cele două stări vom folosi un Simbol-Set de tip LED.

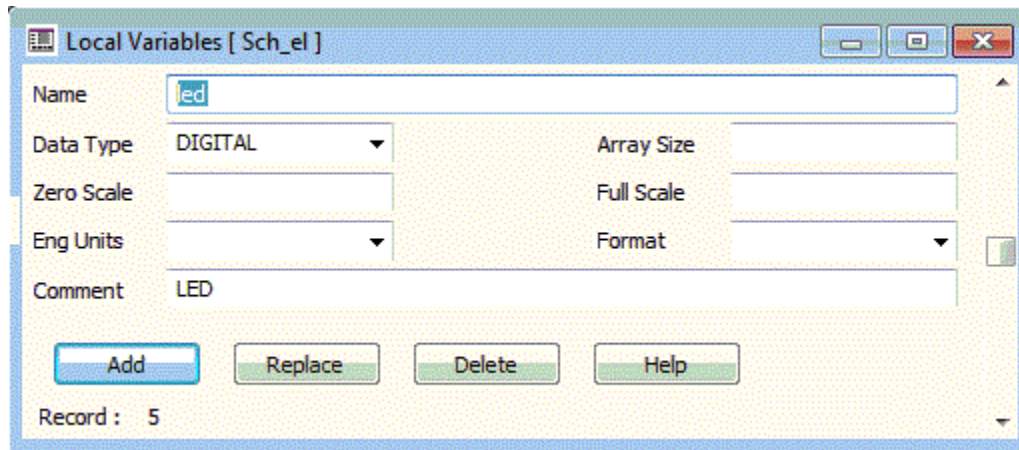


Pentru a afișa deci LED-ul s-a plasat un Simbol Set în care s-au plasat cele două simboluri pentru

Sisteme SCADA

LED stins și LED aprins

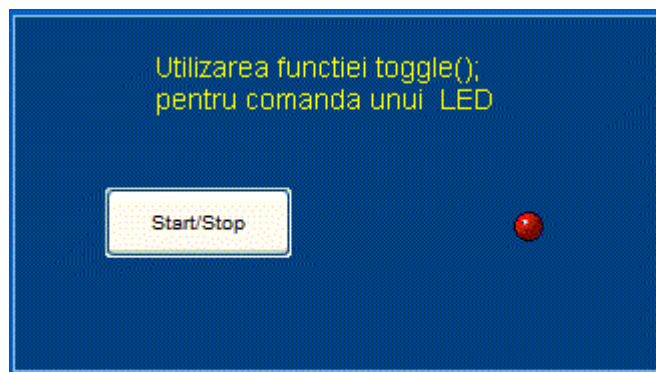
După cum se observa, pentru a controla comportamentul LED-ului, acestuia i s-a atribuit tag-ul DIGITAL numit "led".



Comanda se face prin apăsarea unui butonului cu titlul "Pornit" care setează tag-ul "led" la 1 iar oprirea prin apăsarea butonului cu titlul "Oprit" care setează tag-ul "led" la 0 .

Funcții predefinite

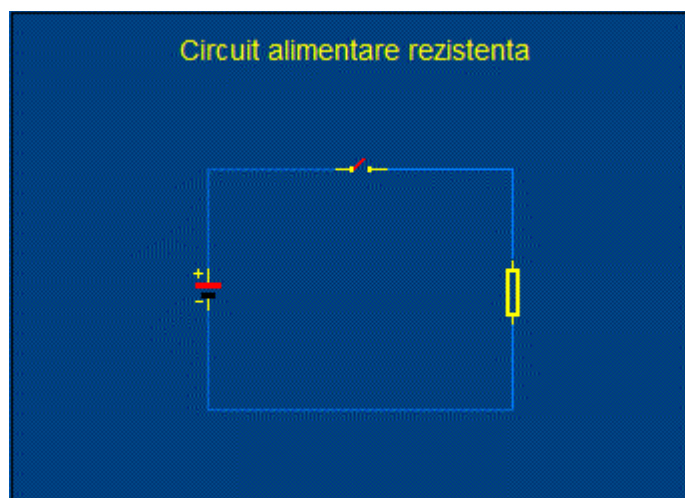
Vom crea o noua pagina grafica "**led2**" , in care vom introduce un singur buton care sa poarte titlul "Pornit/Oprit" și care sa însumeze funcțiilor celor două butoane din aplicația anterioară. Alimentarea cu energie se va semnaliza prin aprinderea unui LED. Pentru a realiza funcționarea butonului vom folosi funcția predefinită **toggle()** completând "Up command" al butonului "Pornit/Oprit" cu **Toggle(led)** .



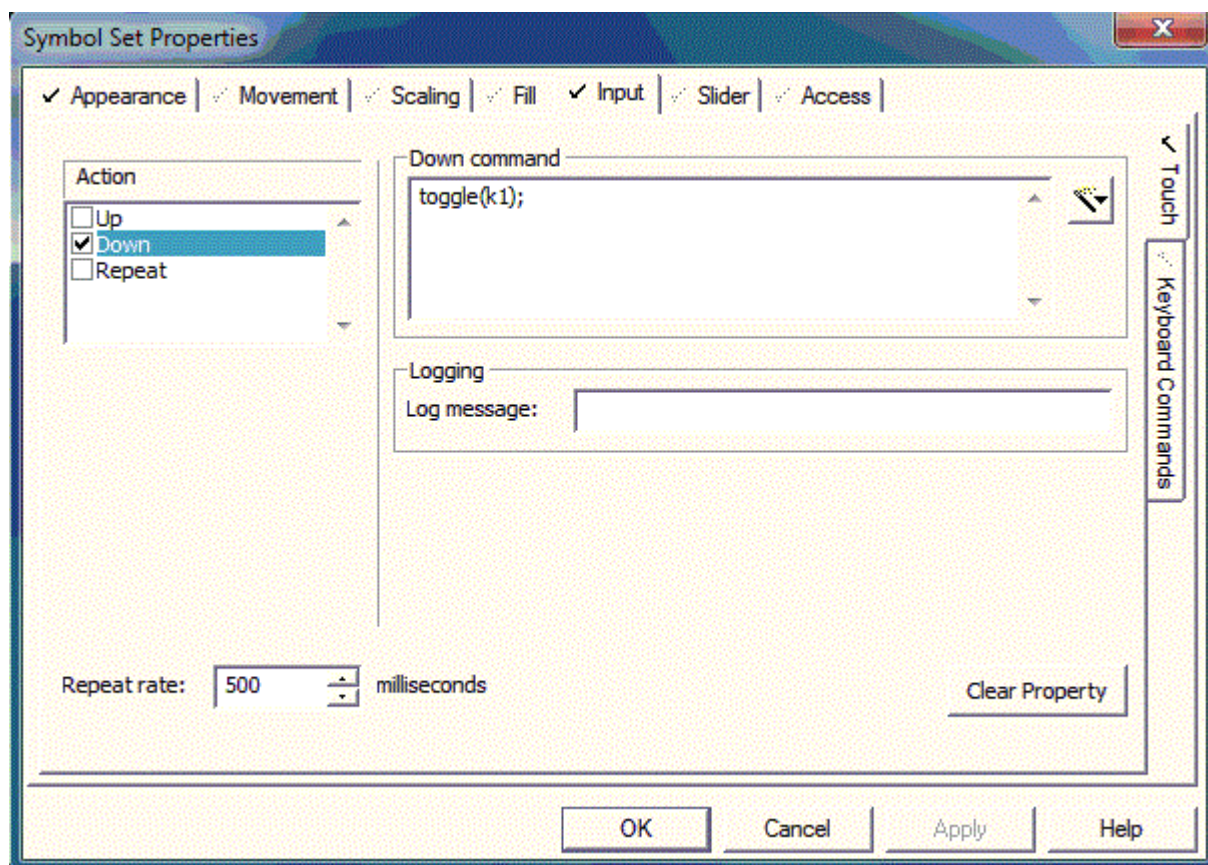
Am putea să simulăm circuite simple în care să avem comutatoare pornit oprit sub forma unui Set Simbol nu sub forma unui buton. Vom crea deci o nouă pagină grafică "**circ_01**" in care vom simula circuitul de alimentare a unei rezistente.

Tag-ul corespunzător comutatorului pentru alimentarea rezistentei, va fi un tag DIGITAL pe care îl vom numi "k1".

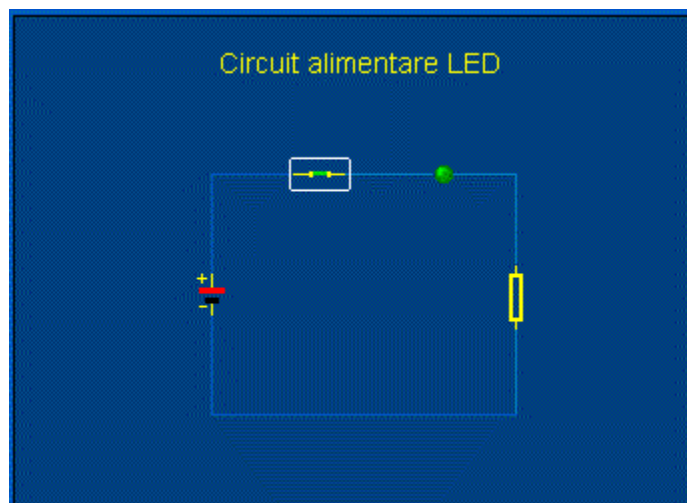
Tag-uri aferente				
Nume	Tip	Domeniu	Um	Comentariu
k1	DIGITAL	-	-	Comutator k1



Am utilizat Set simbolul "k_on", "k_off" definita in biblioteca "Global" Căruia i-am setat proprietatea "Input" cu **toggle(k1)** astfel:



Introducând încă un LED în circuit, vom realiza o nouă pagină grafică **"circ_02"** în care avem un nou circuit cu elemente interactive de comanda și afișare.

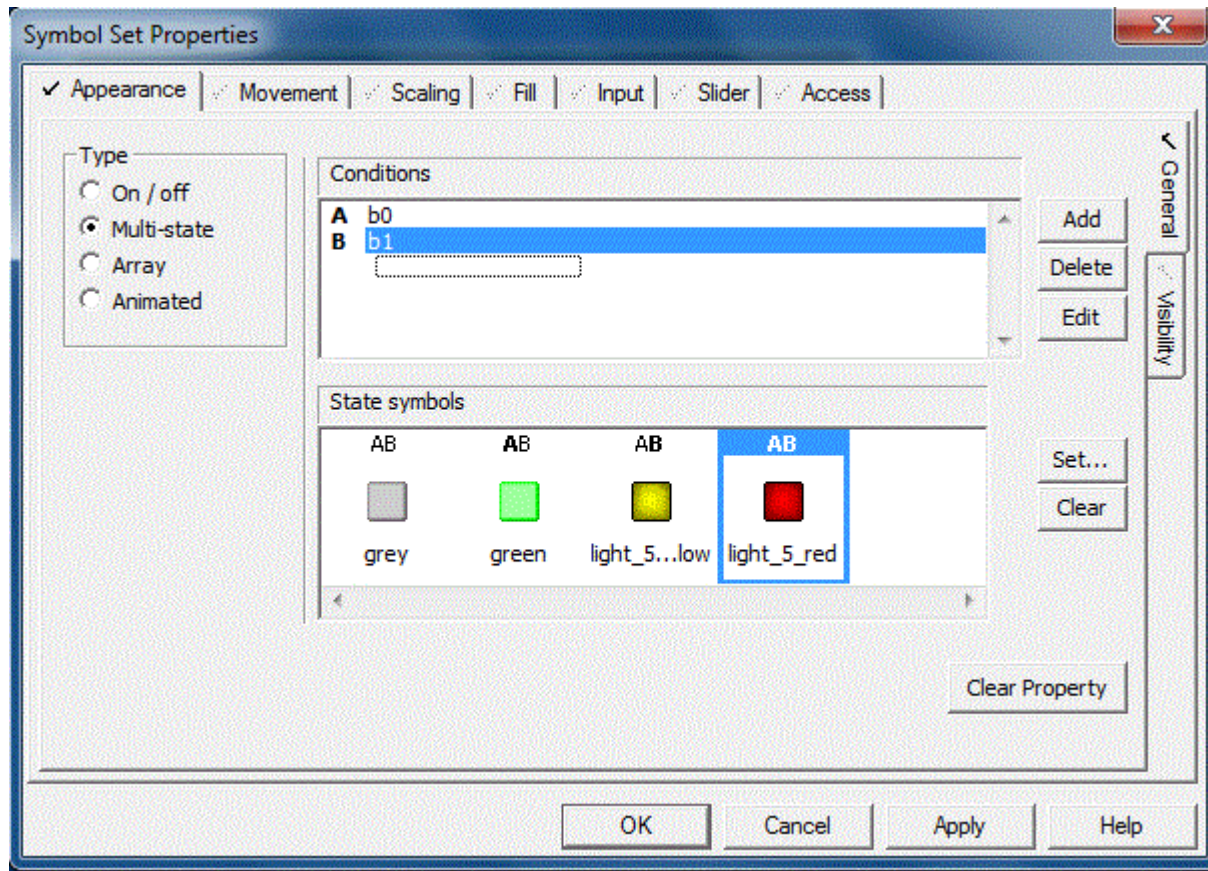


Seturile simbol multi-state

De multe ori, pentru mimarea cat mai fidela a diverselor procese, avem nevoie de simboluri cu mai multe stări. Vom utiliza în continuare un "Symbol Set Multi-state" pentru a afișa un LED cu patru stări

Sisteme SCADA

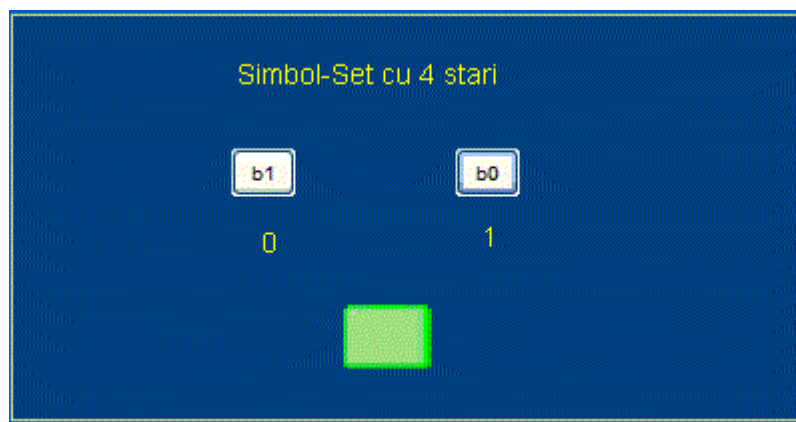
distincte reprezentate de culorile gri, verde, galben, rosu.



Vom realiza o aplicație **"led11"** care utilizează un "Symbol Set Multi-state". Pentru a realiza cele patru combinații, avem nevoie de doua variabile booleene pe care le vom controla cu doua butoane.

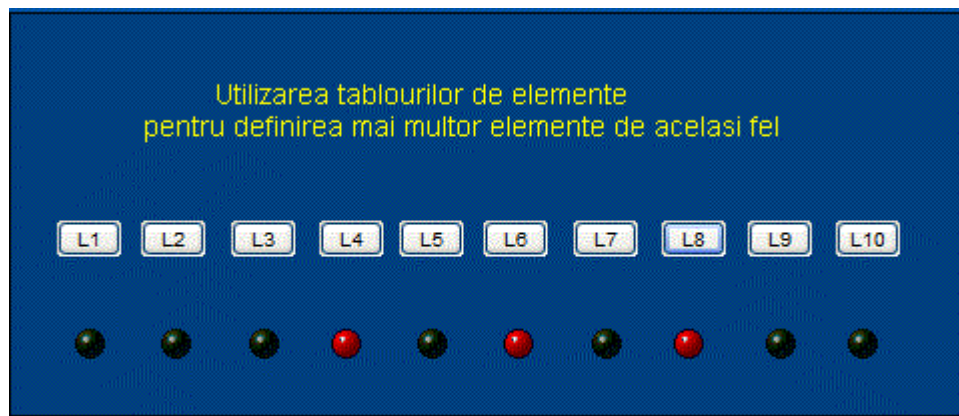
Tag-uri aferente				
Nume	Tip	Domeniu	Um	Comentariu
b0	DIGITAL	-	-	Variabila booleana b0
b1	DIGITAL	-	-	Variabila booleana b1

Butoanele b0 respectiv b1 schimba starea variabilelor b0 respectiv b1, utilizând funcțiile toggle(b0) respectiv toggle(b1)

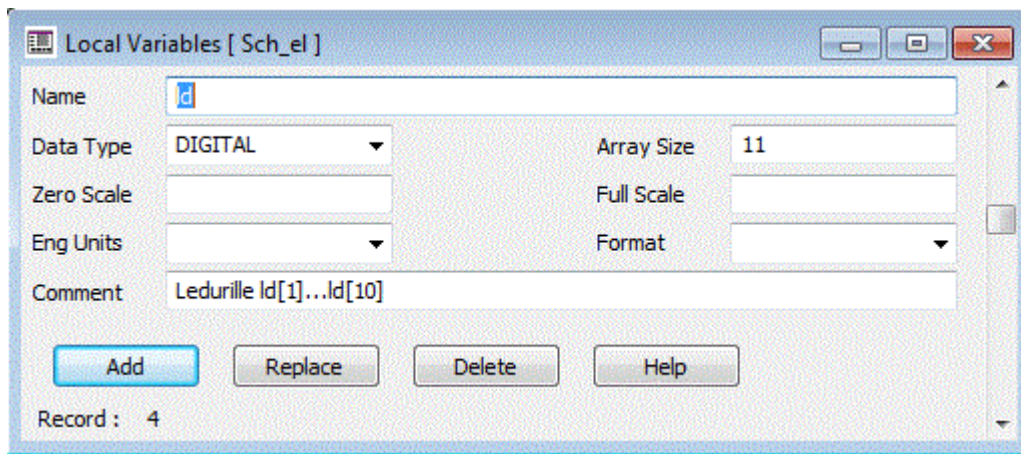


Tablouri de elemente

În multe situații avem de-a face cu mai multe elemente de același tip. Este mult mai convenabil să definim un tablou de elemente decât să definim fiecare element în parte. Să presupunem că vrem să afișăm starea a zece LED-uri pe ecran. Am putea defini câte un tag pentru fiecare LED sau să definim o singură variabilă tag de tip tablou. Să realizăm o pagină grafică "**led10**" de forma imaginii de jos în care plasăm 10 LED-uri și 10 butoane de la care să controlăm cele zece LED-uri.



În cadrul variabilelor locale vom introduce o variabilă de forma:



Pentru fiecare LED vom seta în cadrul facilitații "Simbol Set Properties" tag-ul `ld[i]` în care `i` ia valori de la 1 la 10.

Pentru fiecare buton vom seta "Button Properties" -> Input cu: `toggle(ld[i])`; în care, la fel `i` ia valori de la 1 la 10.

Tablourile pot fi utilizate și în alte cazuri decât definirea tag-urilor. Pentru a utiliza un tablou, acesta trebuie inițial definit.

Definirea unui tablou:

Un tablou de elemente se definește astfel:

tip nume[dim1,{dim2},{dim2}]{=valoare};

Putem defini deci tablouri de 1,2 sau 3 dimensiuni. Vom defini un tablou de o singura dimensiune, deci un vector, astfel:

INT Num[10]

Am definit mai sus un tablou de numere întregi cu numele "Num", având maxim 10 elemente.

Se pot atribui implicit valori astfel:

INT Num[10]=0,1,58,71,134,2,999,3,6,8;

Pentru a accesa elementul `i` se utilizează forma **Num[i]**

Vom putea realiza alte moduri de implementare a comenzii Pornit/ Oprit, folosind limbajul propriu de programare "Cicode".

Utilizând limbajul de programare Cicode se pot controla prin program toate variabilele locale și cele în timp real, precum și toate facilitățile oferite de CitectSCADA cum ar fi tag-uri, variabile locale, trends, grafice, rapoarte.

Cicode se poate utiliza și pentru a interfața aplicația cu alte resurse cum ar fi porturi de comunicație, sistemul de operare, baze de date, etc.

2. Instrucțiuni decizionale

Instrucțiunea **if** se folosește pentru a selecta execuția unei instrucțiuni (sau a unui grup de instrucțiuni) funcție de valoarea logică a unei expresii relaționale

Formatul instrucțiunii:

Instrucțiunea **if** are următoarele formate:

If expresie relațională THEN
 instrucțiune(instrucțiuni);
END

sau

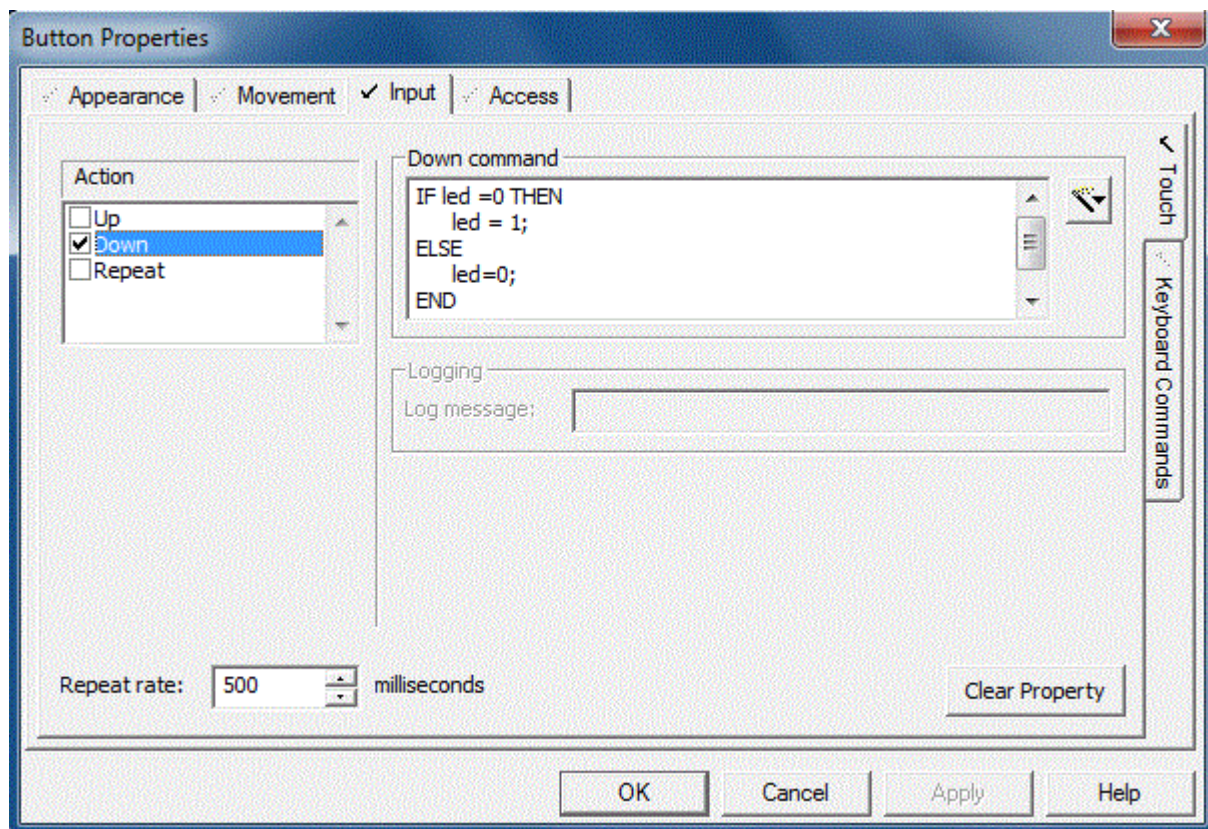
If expresie relațională THEN
 instrucțiune(instrucțiuni);
ELSE
 instrucțiune(instrucțiuni);
END

Aplicații care utilizează instrucțiuni decizionale

Vom încerca să utilizăm în continuare "Cicode" pentru a controla funcționarea butonului Pornit/Oprit. În primul caz am completat "Up command" cu **Toggle(led)**. De data aceasta vom înlocui funcția **Toggle(led)** cu următorul program :

```
IF led =0 THEN
    led = 1;
ELSE
    led=0;
END
```

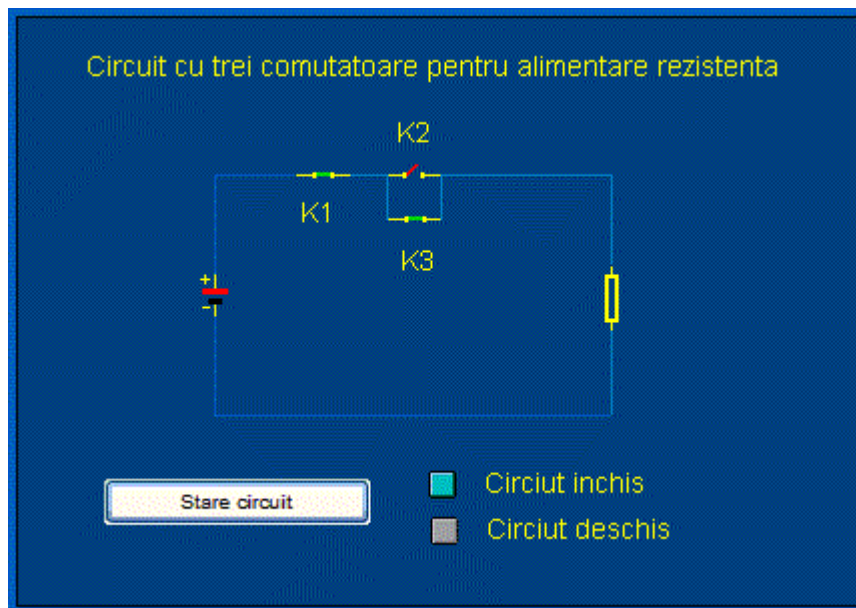
Liniile de cod de sus, vor fi atribuite: Button Properties->Input.



Am realizat astfel pagina grafica "**led3**" in care, folosind instrucțiunea IF, un singur buton preia funcția de Start/Stop.



Următoarea aplicație "**circ_3**" analizează starea unui circuit electric și afișează starea acestuia.



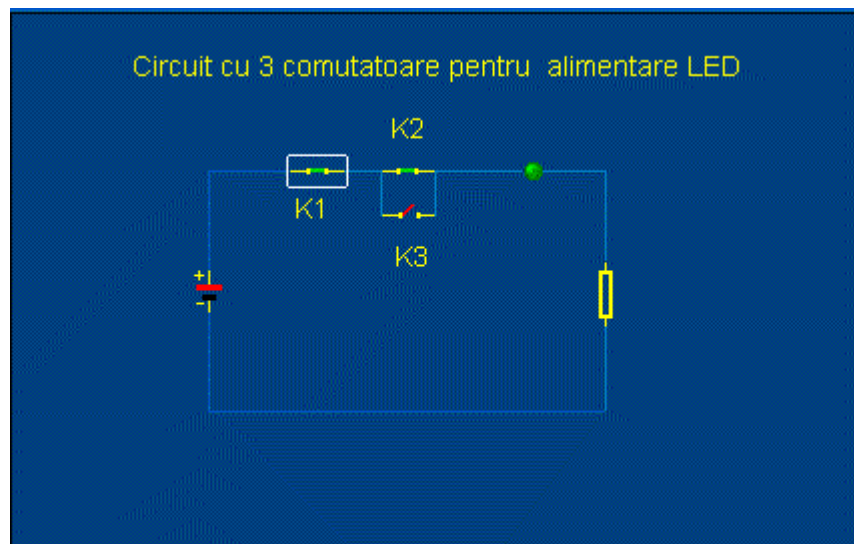
Pentru realizarea aplicației, avem nevoie de următoarele tag-uri:

Tag-uri aferente				
Nume	Tip	Domeniu	Um	Comentariu
k1	DIGITAL	-	-	Intrerupator k1
k2	DIGITAL	-	-	Intrerupator k2
k3	DIGITAL	-	-	Intrerupator k3
led	DIGITAL	-	-	LED

La apăsarea butonului "Stare circuit" se aprinde LED-ul corespunzător. Primul LED este controlat de expresia `led=1` iar LED-ul de jos de expresia `led=0`. Va trebui deci ca la apăsarea butonului "Stare circuit" să se ruleze un mic program care setează în mod corespunzător tag-ul `led` în funcție de starea întrerupătoarelor `k1`, `k2`, `k3`.

```
IF k1 AND ( k2 OR k3) THEN
    led=1;
ELSE
    led=0;
END
```

O rezolvare mai elegantă "**circ_4**", ar fi obținută prin introducerea unui LED în serie cu rezistența. Controlul acestuia nu va mai fi făcut de tag-ul `led` ci de expresia: **`k1 AND ( k2 OR k3)`**



3. Funcții

Funcțiile sunt des utilizate pentru realizarea aplicațiilor SCADA. Majoritatea butoanelor, prin intermediul evenimentelor lansează secvențe de cod împachetate în cadrul unor funcții. Există o varietate mare de funcții astfel avem funcții cu sau fără parametri, funcții care returnează sau nu valori.

Formatul pentru definirea unei funcții fără parametri și fără returnare de valori:

Pentru definirea unei astfel de funcții se folosește următorul format:

```
FUNCTION nume_funcție()
  declarații;
  .
  .
  .
  declarații;
END
```

Formatul pentru definirea unei funcții cu parametri și fără returnare de valori:

Pentru definirea unei astfel de funcții se folosește următorul format:

```
FUNCTION nume_funcție(Argumente)
  declarații;
  .
  .
  .
  declarații;
END
```

Formatul pentru definirea unei funcții cu parametri și cu returnare de valori:

Pentru definirea unei astfel de funcții se folosește următorul format:

Tip valoare returnata FUNCTION nume_funcție(Argumente)

declarații;

.

.

.

declarații;

RETURN valoare

END

Aplicații care utilizează funcții definite de utilizator

În aplicațiile anterioare, am apelat funcția **Toggle()**, funcție aflată în biblioteca limbajului Cicode, sau am scris direct în câmpul "Up command" câteva linii de program care înlocuiesc funcția **Toggle()**.

De data aceasta vom încerca să scriem o funcție proprie pe care să o invocăm la apăsarea butonului "Pornit/ Oprit"

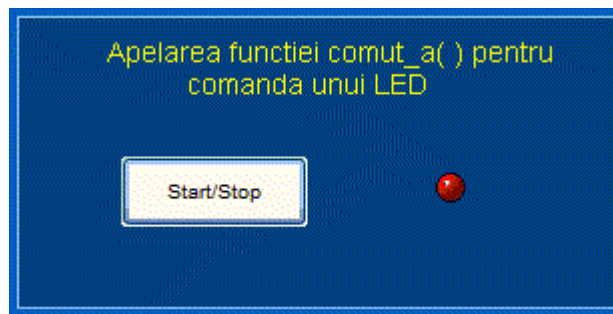
Să denumim funcția "comut_a()" și să deschidem CicodeEditor pentru a o scrie. CicodeEditor editor se lansează din "Citect Explorer"->Cicode Files->Create new Cicode page.

Vom defini funcția "comut_a()" astfel:

```
FUNCTION comut_a()  
  
    IF led =0 THEN  
        led = 1;  
    ELSE  
        led=0;  
    END  
END
```

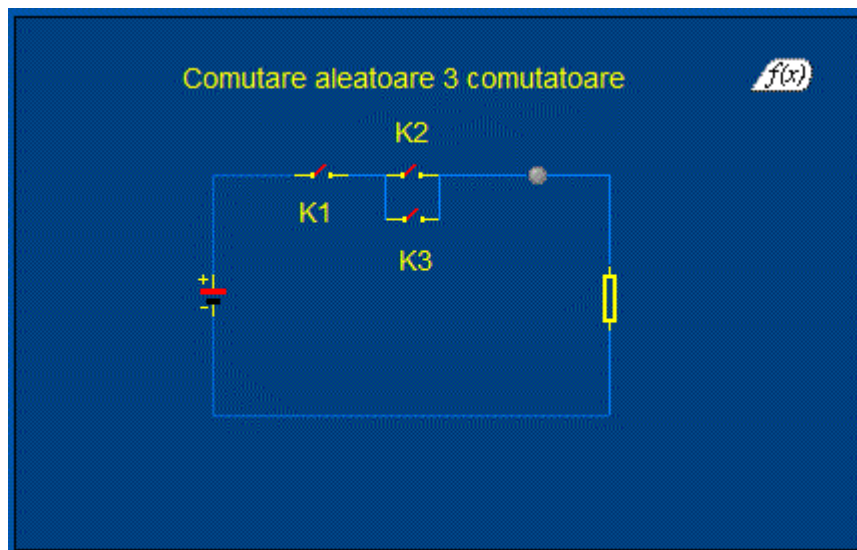
În acest moment, pentru butonul "Start/Stop" - setăm proprietatea " Input" -Action up, "Up command" cu **comut_a()**

Utilizăm deci butonul "Start/Stop" și realizăm o nouă pagină grafică având numele "**led12**" similară cu pagina grafică de jos.

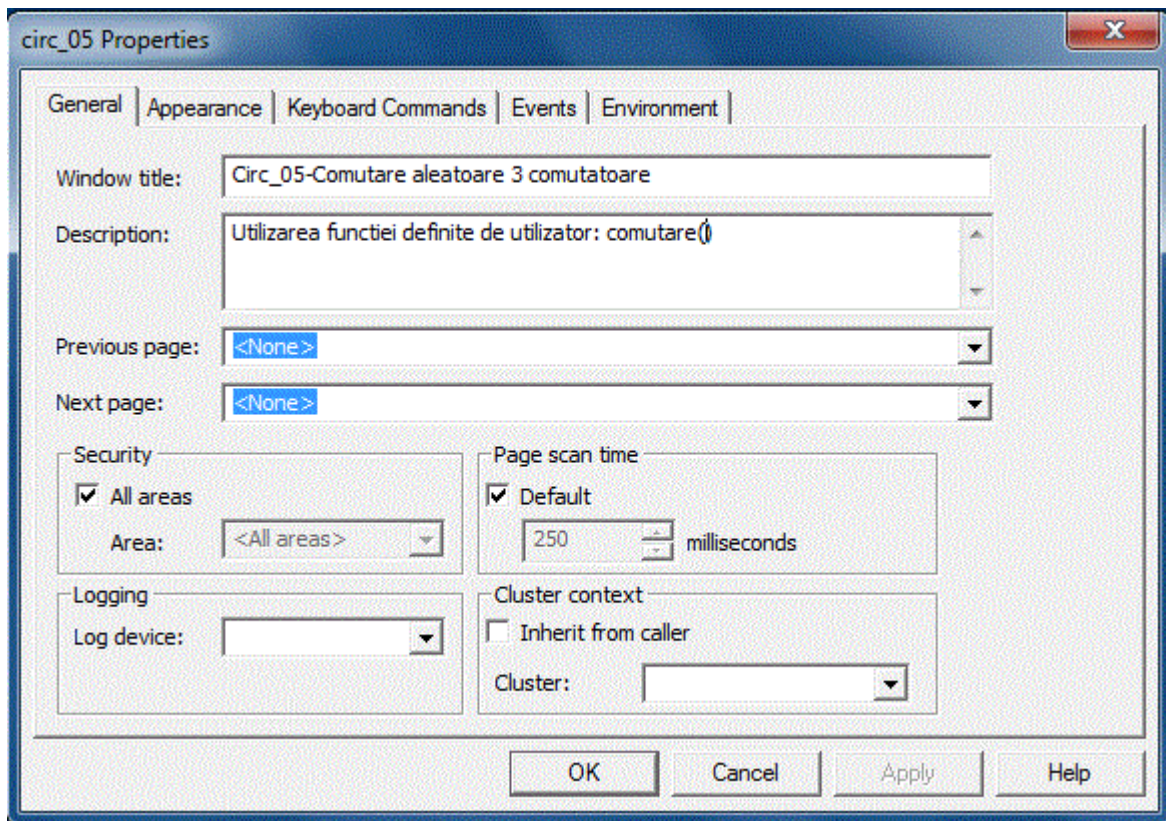


Pentru LED s-a ales un "Symbol Set" de tip "On/Off" controlat de tag-ul "led"

Funcția **comut_a()** se lansează de evenimentul click al butonului. In general funcțiile scrise de utilizatori, se lansează de diverse evenimente. Avem posibilitatea să plăsam o funcție chiar și pe pagina grafică. Vom relua aplicația "circ_4" și vom realiza aplicația "circ_5" in care vom plasa o funcție care comută aleator cele trei comutatoare.



Funcția plasată am denumit-o "comutare()". Declanșarea acesteia se face la fiecare scanare a paginii. In mod implicit, scanarea se face la 250 ms dar se poate stabili un alt timing.



```

FUNCTION comutare()
    k1=Rand(2);
    k2=Rand(2);
    k3=Rand(2);
END

```

După cum se observă, s-a folosit funcția predefinită **Rand(2)** funcție care întoarce un număr între 0 și 1.

După rularea aplicației, se observa ca în fiecare secundă se produc 4 comutări. Pentru a mari timpul între doua comutări, am putea sa umblam la timpul de scanare, dar asta ar înrăutăți funcționarea paginii grafice.

Vom încerca sa rescriem funcția comutare() astfel încât sa putem controla timpul între doua comutări. Vom contoriza scanările paginii grafice și vom executa atribuirea de valori pentru k1, k2, k3 numai după un anumit număr de scanări. Vom introduce un tag, pe care il vom folosi pe post de contor. Avem deci nevoie de tag-urile:

Tag-uri aferente				
Nume	Tip	Domeniu	Um	Comentariu
k1	DIGITAL	-	-	Intrerupator k1
k2	DIGITAL	-	-	Intrerupator k2
k3	DIGITAL	-	-	Intrerupator k3

i	INTEGER	-	-	Contor
---	---------	---	---	--------

Funcția comutare() devine:

```
FUNCTION comutare()  
    i=i+1;  
    IF i>3 THEN  
        k1=Rand(2);  
        k2=Rand(2);  
        k3=Rand(2);  
        i=0;  
    END  
END
```

In acest caz comutarea se face numai la interval de 4 scanări deci la interval de $4 \cdot 250\text{ms} = 1\text{s}$

Test de autoevaluare

- -Marcați răspunsurile corecte la întrebările următoare.
- -ATENȚIE: pot exista unul, niciunul sau mai multe raspunsuri corecte la aceeasi intrebare.
- -Timp de lucru: 10 minute

1. Cicode este:

- ☐ a. Un mediu de dezvoltare aplicații SCADA
- ☐ b. Un mediu de programare independent
- ☐ c. Un mediu de programare inclus in mediul de dezvoltare Citect SCADA
- ☐ d. Un mediu de programare visual pentru aplicații SCADA

2. Pentru a afișa 4 simboluri de tip "Symbol Set Multi-state" avem nevoie de:

- ☐ a. 1 variabila de tip Array
- ☐ b. 2 variabile booleene
- ☐ c. 1 variabila de tip INTEGER
- ☐ d. 1 variabila vector

3. Atribuirea de valori elementelor unui tablou se poate face:

- ☐ a. Într-o singura declarație
- ☐ b. Atribuind valori în parte fiecărui element
- ☐ c. Prin utilizarea unei funcții predefinite
- ☐ d. Printr-o metoda a variabilei de tip tablou

4. Instrucțiunea IF presupune:

- ☐ a. Cel puțin două expresii relaționale
- ☐ b. Cel puțin o expresie relațională
- ☐ c. Cel puțin o ramura de execuție
- ☐ d. Cel puțin o variabila de tip "tag"

5. O funcție returnează:

- ☐ a. Nici un parametru
- ☐ b. Un singur parametru
- ☐ c. Mai multi parametri
- ☐ d. Numai un parametru de tip tag

Grila de evaluare: 1-c; 2-b; 3-a, b; 4-b, c; 5-a, b.

Rezumat

Limbajul de programare Cicode

Cicode este un limbaj de programare integrat in mediul de dezvoltare CitectSCADA destinat dezvoltării de aplicații SCADA, permițând controlul software a elementelor utilizate pentru mimarea și controlul proceselor.

Limbajul Cicode permite utilizarea:

- Set simbolului on/off
- Funcțiilor predefinite
- Set simbolurile multi-state
- Tablourilor de elemente
- Instrucțiunilor decizionale
- Instrucțiunilor repetitive
- Funcțiilor definite de utilizator

Definirea unui tablou:

```
tip nume[dim1,{dim2},{dim2}]{=vaoare};
```

exemplu:

```
INT Num[10]
```

```
INT Num[10]=0,1,58,71,134,2,999,3,6,8;
```

Instrucțiunea **if**

Formatul instrucțiunii:

```
If expresie relațională THEN  
    instrucțiune(instrucțiuni);  
END
```

sau

```
If expresie relațională THEN  
    instrucțiune(instrucțiuni);  
ELSE  
    instrucțiune(instrucțiuni);  
END
```

Funcții

- Formatul pentru definirea unei funcții fără parametri și fără returnare de valori

```
FUNCTION nume_funcție()  
    declarații;  
.  
.  
    declarații;  
END
```

- Formatul pentru definirea unei funcții cu parametri și fără returnare de valori

```
FUNCTION nume_funcție(Argumente)  
    declarații;  
.  
.  
    declarații;  
END
```

-Formatul pentru definirea unei funcții cu parametri și cu returnare de valori

```
Tip valoare returnata FUNCTION nume_funcție(Argumente)  
    declarații;  
.  
.  
    declarații;  
    RETURN valoare  
END
```

Rezultate așteptate

După studierea acestui modul, ar trebui să cunoașteți:

- Cum să scrieți aplicații utilizând limbajul Cicode
- Să utilizați instrucțiuni decizionale
- Să definiți funcții proprii
- Să realizați aplicații SCADA în care să folosiți elemente de programare

Termeni esențiali

Termen	Descriere
SCADA	Supervisory Control And Data Acquisition
Tag	Nume generic pentru elementele din procesul monitorizat codificate prin intermediul variabilelor
HMI	Human Machine Interface -Interfața dintre aplicație și utilizator
Limbaj Cicode	Limbaj de programare inclus în mediul de dezvoltare Citect SCADA
Funcții predefinite	Funcții incluse definite și incluse în mediul de programare Cicode
Instrucțiuni decizionale	Instrucțiuni care permit alegerea setului de instrucțiuni care urmează a fi executate în funcție de o expresie relațională
Expresie relațională	Expresie a cărei rezultat este o valoare logică

Recomandări bibliografice

- [1] Traian Turc, Elemente de programare C++ utile în ingineria electrica, Ed.Matrixrom, București,2010
- [2] Traian Turc, Programare avansata C++ pentru ingineria electrica, Ed.Matrixrom, București,2010
- [3] Traian Turc, Programarea in limbaje de asamblare, uz intern, Univ."Petru Maior" ,Tg. Mures,2009
- [4] Traian Tur,Brevet de invenție nr:11863 "Sistem pentru automatizarea si monitorizarea proceselor industriale", OSIM, 2003
- [5] Jeff Kent, C++ fara mistere,Ed.Rosetti Educational 2004 .
- [6] Boldur Barbat - Informatica industrială - Programarea în timp real – Institutul Central pentru Conducere si informatica 1984
- [7] Ioan Babuita – Conducerea automata a proceselor – Ed. Facla 1985
- [8] Ghercioiu-National în struments - Orizonturi în instrumentație 1995

Link-uri utile

- 1. <http://www.free-scada.org/> - Free SCADA - 2009.
- 2. <http://www.7t.dk/igss/default.asp> - IGSS SCADA System - 2009
- 3. <http://www.7t.dk/igss/default.asp?showid=374> - IGSS Online SCADA Training - 2009
- 4. <http://www.7t.dk/free-scada-software/index.html>- IGSS Free SCADA Software -2009
- 5. <http://www.citect.com/> - CITECT SCADA -2009
- 6. http://www.citect.com/index.php?option=com_content&view=article&id=1457&Itemid=1314 - Download CITECT demo - 2009
- 7. <http://www.indusoft.com/index.asp> - INDUSOFT SCADA - 2009
- 8 <http://www.gefanuc.com/products/2819> - Proficy HMI/SCADA - CIMPLICITY - 2009.
- 9. <http://www.genlogic.com/> - Dynamic Graphics, Data Visualization, Human-Machine Interface (HMI) - 2010
- 10 <http://www.genlogic.com/demos.html> - On-Line Java and AJAX Demos - 2010
- 11 <http://www.free-scada.org/> - - 2009
- 12 <http://www.free-scada.org/> - - 2009

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

1. Pentru a simula funcționarea unui LED, se utilizează:

- ☐ a. Un Simbol-Set de tip "On/off"
- ☐ b. Simbol-Set de tip "Animated"
- ☐ c. Simbol de tip "lights"
- ☐ d. O forma geometrica "circle"

2. Care dintre funcțiile de jos sun predefinite ?

- ☐ a. Funcția toggle()
- ☐ b. Funcția screen()
- ☐ c. Funcția Rand()
- ☐ d. Funcția comut()

3. Care dintre enunțurile de mai jos sunt adevărate ?

- ☐ a. Instrucțiunea IF are mai multe formate
- ☐ b. Instrucțiunea IF include o expresie relațională
- ☐ c. Instrucțiunea IF este o instrucțiune decizională
- ☐ d. Instrucțiunea IF este inclusă în mediul de programare Cicode

4. O funcție poate fi apelată de:

- ☐ a. Evenimentele unui buton
- ☐ b. Scanarea ecranului
- ☐ c. O variabilă tag
- ☐ d. Un Symbol Set

5. Funcția predefinită "Random (): "

- ☐ a. Are nevoie de un parametru
- ☐ b. Nu are nevoie de nici un parametru
- ☐ c. Are nevoie de doi parametri
- ☐ d. Are nevoie de un parametru de tip tag

Grila de evaluare: 1-a; 2-a, c; 3-a, b, c, d; 4-a, b; 5-a.