Aplicaţii în energetică - linii monofilare

Cuprins

Obiective

- Realizarea de aplicaţii SCADA în energetică.
- Definirea simbolurilor grafice necesare dezvoltării de aplicaţi SCADA destinate sectorului energetic.
- Definirea elementelor unei linii monofilare.
- Aplicaţii SCADA pentru simularea unei linii monofilare.
- Aplicaţii SCADA pentru simularea unei linii monofilare si implementarea algoritmilor de condiţionalitate intre elementele liniei monofilare.
- Aplicaţii SCADA pentru simularea conectării secvenţiale a liniilor monofilare.
- Aplicaţii SCADA pentru managementul liniilor liniilor monofilare.

Organizarea sarcinilor de lucru

- Parcurgeţi cele trei capitole ale cursului.
- In cadrul fiecărui capitol urmăriţi exemplele ilustrative şi încercaţi sa le realizaţi în medul de dezvoltare "Citect".
- Fixaţi principalele idei ale cursului, prezentate în rezumat.
- Completaţi testul de autoevaluare.
- Timpul de lucru pentru parcurgerea testului de autoevaluare este de 15 minute.

1. Scheme monofilare - elemente introductive

Sectorul energetic este unul din sectoarele care necesita control şi monitorizare la diverse nivele. Vom realiza în continuare o serie de aplicaţii SCADA în domeniul energetic.

Crearea proiectului: "Scheme monofilare"

Vom dezvolta în continuare câteva aplicaţii din domeniul energetic în cadrul unui proiect numit **"Sch_monof"** .

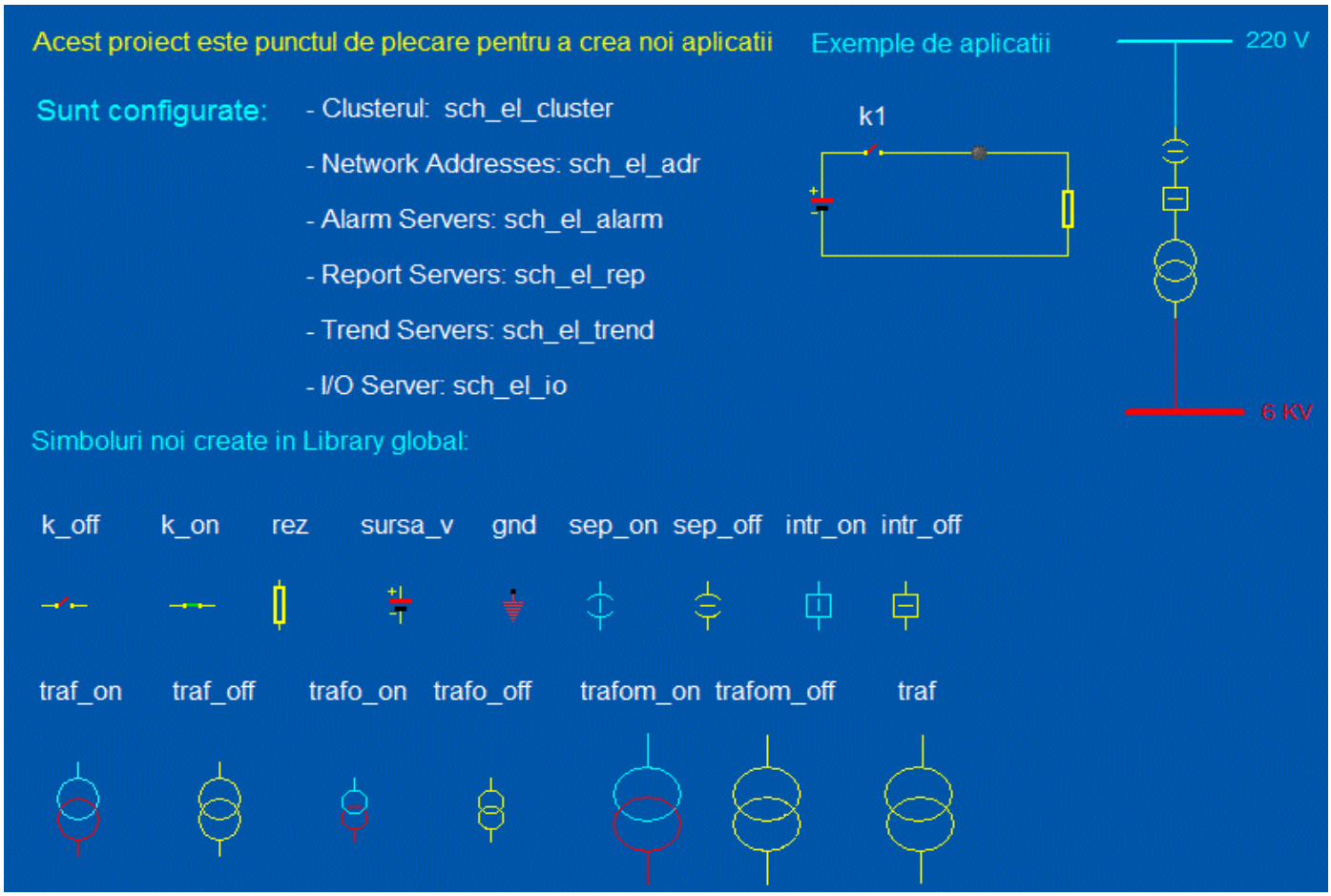
In cadrul acestui proiect vom realiza diverse scheme monofilare cărora le vom da funcţionalitate. Pentru a realiza scheme monofilare avem nevoie de o serie de simboluri cum ar fi simboluri pentru: separatoare, intreruptoare, transformatoare etc.

In proiectul **"Sch_el_start"** , sunt realizate deja o serie de astfel de simboluri, asa ca vom porni de la acest proiect.

Acest proiect poate fi descarcat aici : Download - "Sch_el_start"

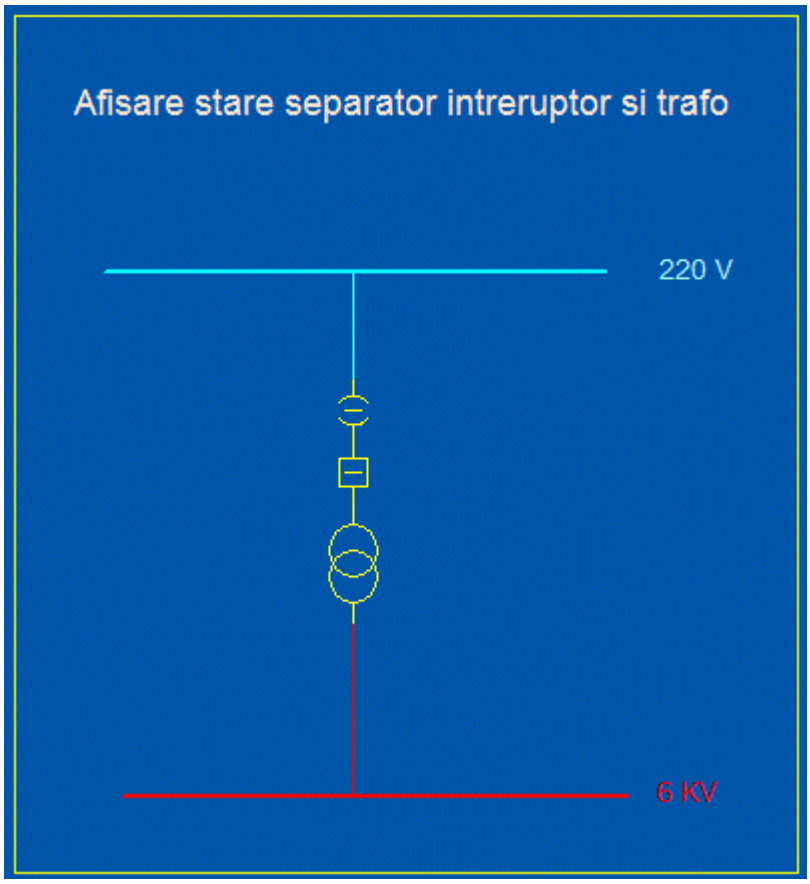
După ce s-a download-at acest fişier, din Citect Explorer->Restore se încarcă acest proiect şi i se atribuie numele **"Sch_monof"** .

Proiectul conţine pagina grafică start afişată mai jos.



Elementele unei linii monofilare

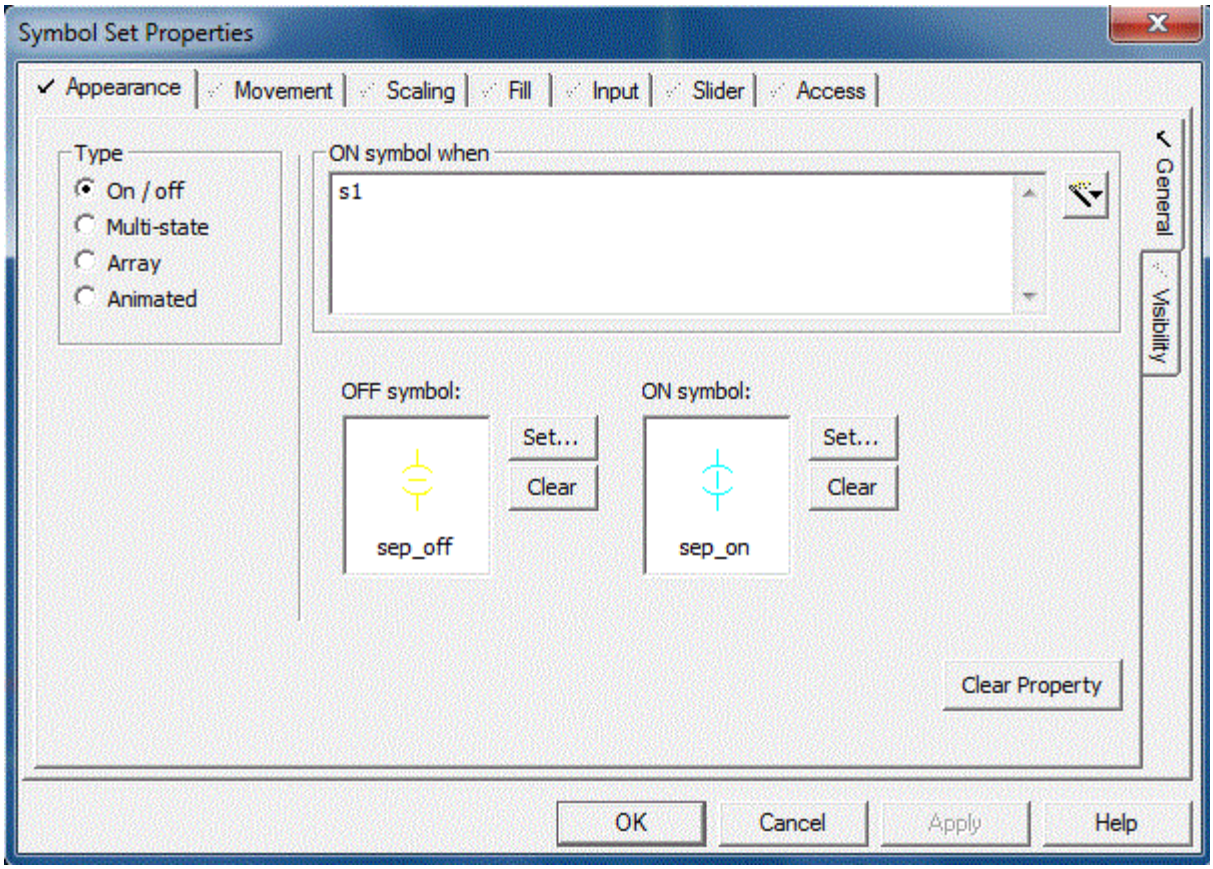
Se va crea o nouă pagină grafică numită **sch_monof_1** , în care vom plasa un separator, un intreruptor şi un transformator, realizând schema monofilară de mai jos. Toate simbolurile necesare acestei scheme monofilare au fost create anterior, odată cu pagina grafică **start** existentă în proiectul iniţial **"Sch_el_start"** .



Dorim să realizam o aplicaţie care să monitorizeze starea componentelor din schema monofilară de mai sus. Sa presupunem că avem un sistem de achiziţie şi control care este conectat la elementele din schema monofilară de sus. Starea fiecărui element este oglindită de tag-urile corespunzătoare adică:

Tag-uri aferente				
Nume	**Tip**	**Domeniu**	**Um**	**Comentariu**
s1	DIGITAL	-	-	Stare separator s1
intr1	DIGITAL	-	-	Stare intreruptor intr1

Pentru separator s-a utilizat un "Simbol Set" compus din simbolurile "sep_off", "sep_on". Starea separatorului fiind în concordantă cu starea tag-ului **s1**

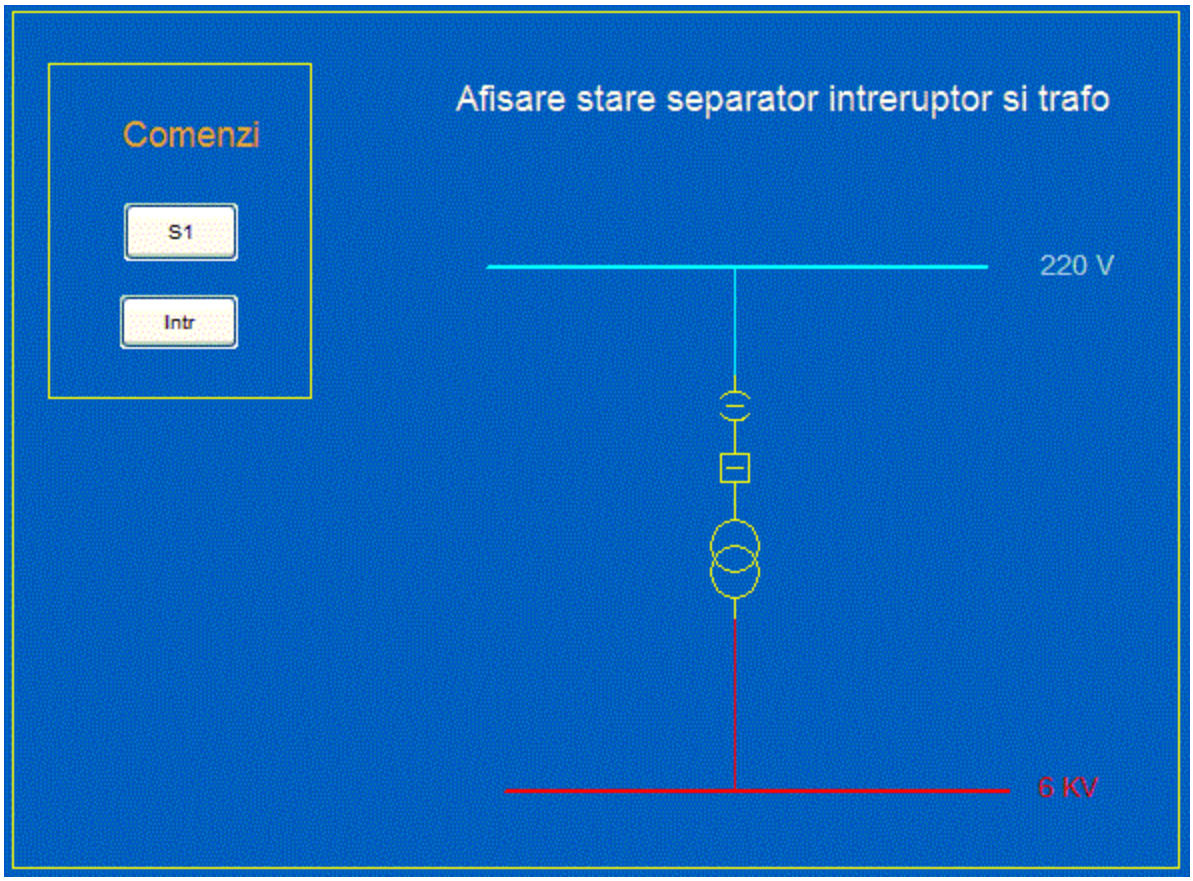


Similar, pentru intreruptor s-a utilizat un "Simbol Set" compus din simbolurile "intr_off", "intr_on", starea acestuia fiind în concordantă cu starea tag-ului **intr1** .

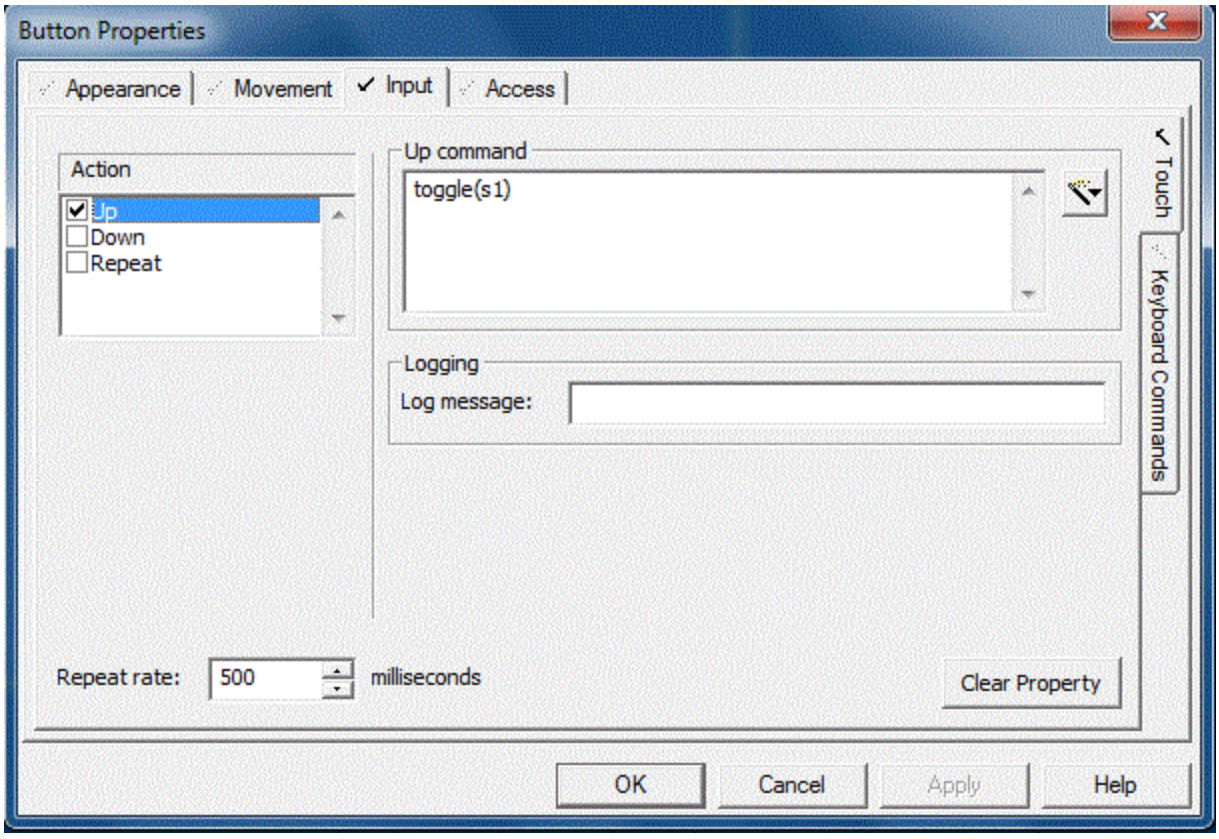
Pentru transformator s-a utilizat un "Simbol Set" compus din simbolurile "traf_off", "traf_on". De data aceasta, starea transformatorului este în concordantă cu expresia: **s1*intr1** .

Toate simbolurile utilizate anterior, se găsesc în "Library: global".

In lipsa sistemului de achiziţie şi control, nu ne putem baza pe tag-urile amintite anterior. Vom putea simula însa aceste tag-uri, prin utilizarea variabilelor locale. Vom utiliza deci variabilele locale în locul variabilelor Tag. Pe tot parcursul acestei lucrări vom utiliza numai variabile locale pentru a putea realiza aplicaţii în lipsa sistemului de achiziţie si comanda. Toate aceste variabile vor fi simulate local. Aplicaţia anterioara va fi completata cu doua butoane pentru a simula starea variabilelor **s1** respectiv **intr1** .

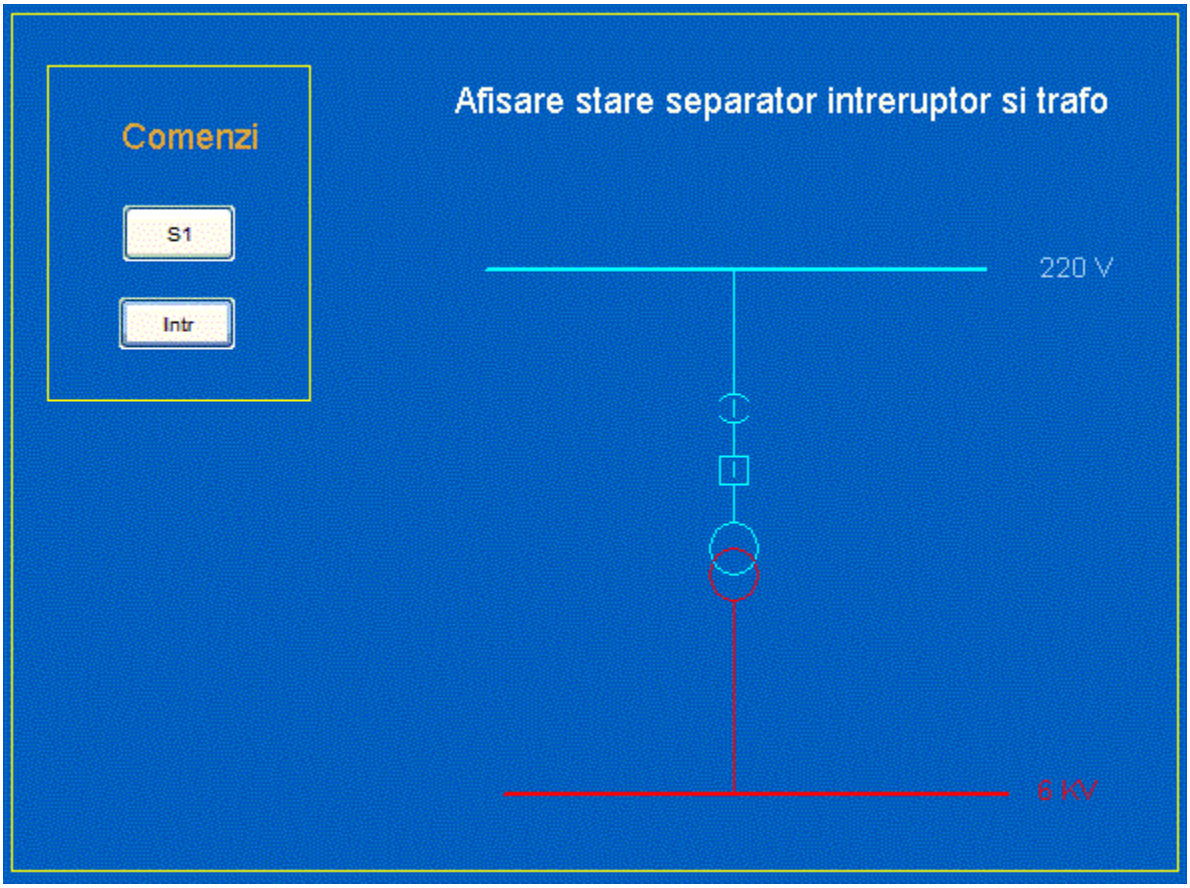


La apăsarea butoanelor, acestea trebuie sa schimbe starea variabilelor locale asociate astfel butonul S1 trebuie sa fie setat astfel:



Similar la apăsarea butonului "Intr", acesta trebuie sa execute funcţia: toggle(intr1).

In acest moment schema este funcţională în sensul ca la apăsarea butonului S1, separatorul îşi schimbă starea, la fel la apăsarea butonului Intr, intreruptorul intr1 îşi schimbă starea, iar ral realizarea condiţiei intr1=1 si s1=1 transformatorul indica faptul că este în funcţiune conform situaţiei de mai jos:

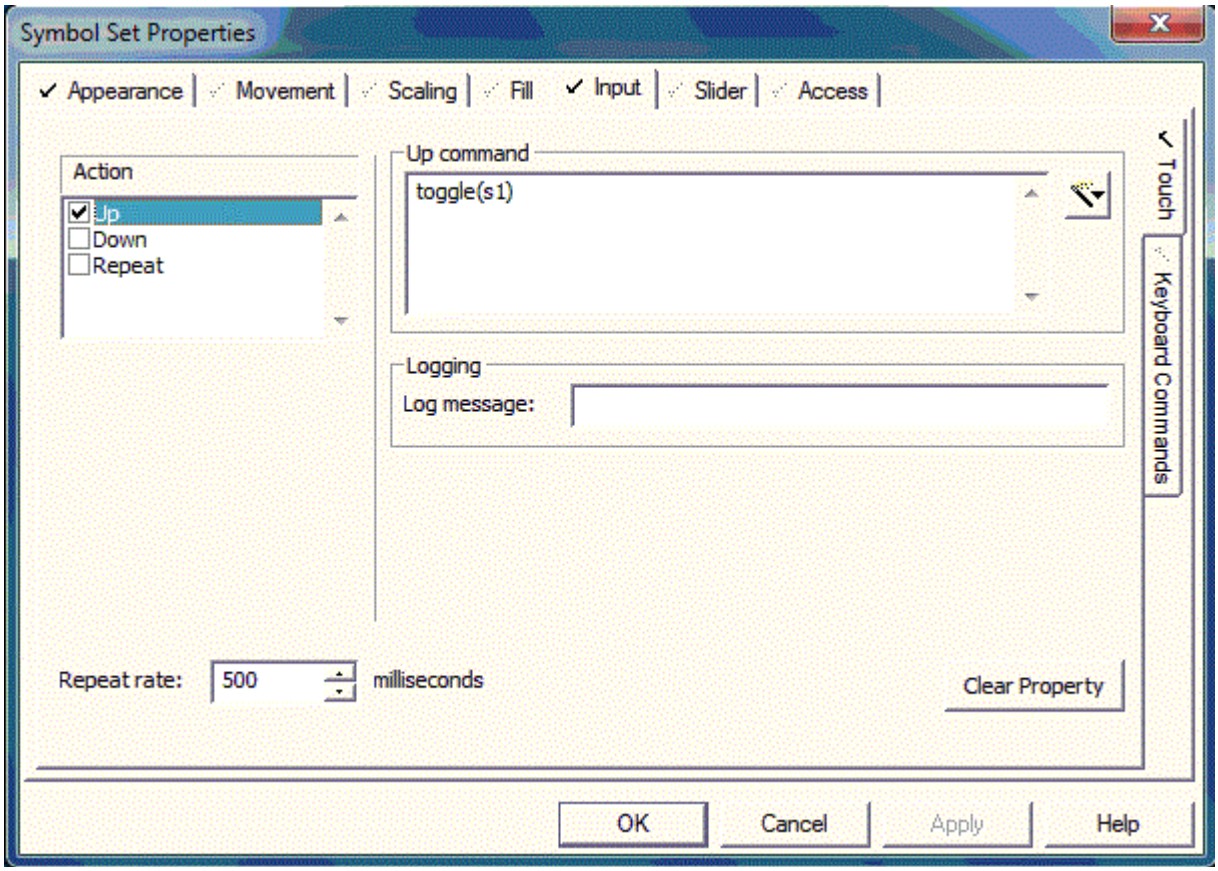


1. Funcţionarea unei lini monofilare

Vom introduce în continuare o serie de elemente care să facă funcţională linia monofilară schiţată anterior.

Să introducem deci noi facilităţi şi să adaugăm comenzi directe pentru închiderea respectiv deschiderea separatorului şi intreruptorului, realizând astfel o nouă pagină grafică **sch_monof_2** . In cazul în care sistemul de achiziţie şi comandă este prevazut atât cu elemente pentru citirea stărilor cat şi cu elemente de comandă, putem îmbogăţi funcţionalitatea schemei precedente cu facilitaţi de comandă pe lângă facilităţile anterioare de monitorizare. Vom da deci o dubla funcţionalitate celor două simboluri (s1, intr1) atât pentru afişarea stării cat şi pentru preluarea comenzii de închidere respectiv deschidere.

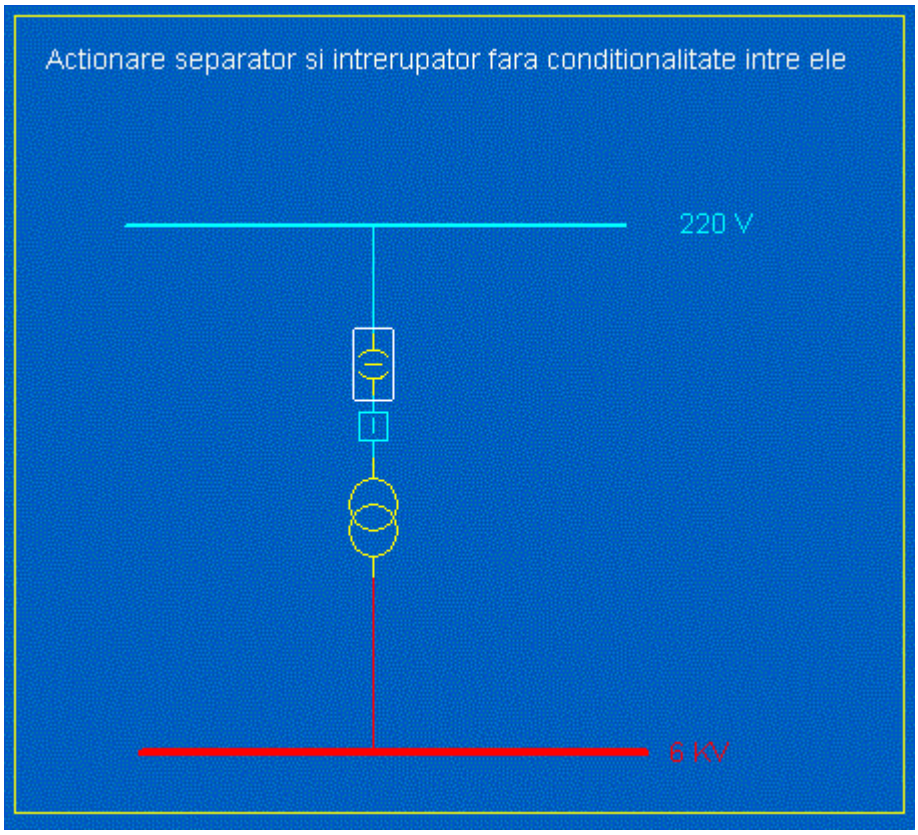
Se va atribui Set Simbolului s1 şi atribuţii de tip "Input" astfel:



Cu alte cuvinte, se înglobează funcţionalitatea butonul S1 în "Set Simbolul" S1.

Se procedează identic cu "Set Simbolul" Intr1.

In noua pagină grafică **sch_monof_2** se observă că cele două simboluri (s1, intr1) devin senzitive, putându-se lansa comenzi de închidere respectiv deschidere.



Am eliminat cele două butoane ce simulează starea celor doua elemente, comanda lor, făcându-se direct de la acestea.

Când cele două elemente sunt închise, simbolul transformatorului se modifică indicând funcţionarea acestuia. In aplicaţiile reale există o ordine în care se acţionează cele doua elemente, în sensul ca separatorul nu poate fi niciodată acţionat în sarcină. Această schema nu asigura această constrângere, nefiind condiţionalitate intre cele doua comenzi.

Condiţionarea separatorului de intreruptor

In următoarea pagină grafică **sch_monof_3** se va realiza condiţionalitatea intre cele doua comenzi. Ţinând cont ca separatorul nu poate fi niciodată acţionat in sarcină, acţionarea acestuia nu mai reprezintă o simplă operaţiune de schimbare a stării acestuia ci trebuie condiţionată de starea intreruptorului astfel:

```
IF (intr1=0)
THEN
toggle(s1)
END
```

S-a setat deci proprietatea "Input" a "Set Simbolului" s1.

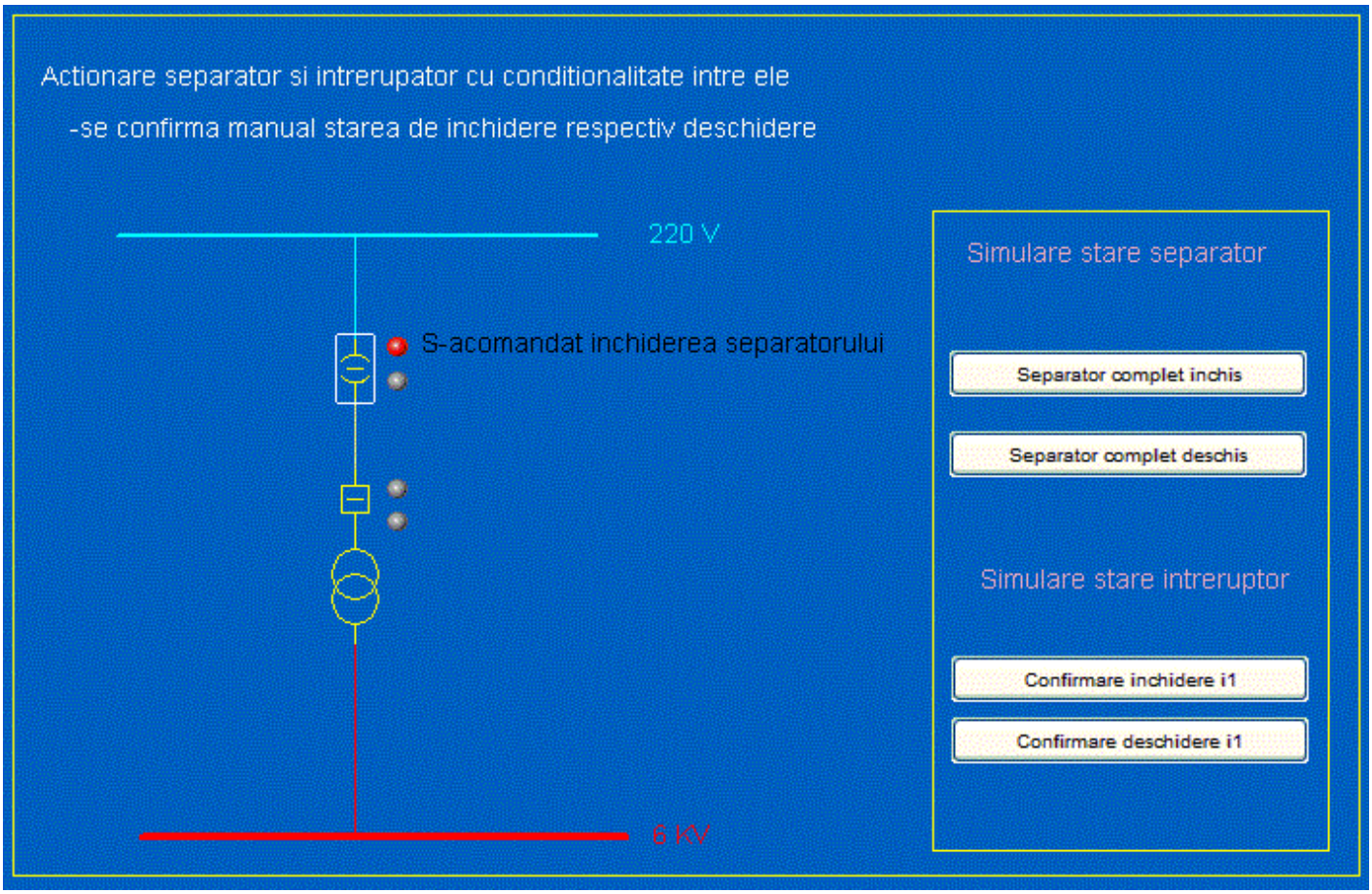
Confirmarea execuţiei comenzilor

In sistemele energetice, închiderea respectiv deschiderea unui separator sau a unui intreruptor nu este instantanee. Vom tine cont la următoarea pagină grafică **sch_monof_4** de acest fenomen. După comanda acţionarii unui element se scurge un interval de timp pană la efectuarea completă a comenzii. In aplicaţia curentă, după acţionarea unui element vom avea nevoie de un semnal care să confirme efectuarea comenzii.

Vom introduce pe lângă tag-urile existente, noi tag-uri pentru a implementa răspunsul efectuării comenzii.

Tag-uri aferente				
Nume	**Tip**	**Domeniu**	**Um**	**Comentariu**
s1	DIGITAL	-	-	Stare separator s1
intr1	DIGITAL	-	-	Stare intrerupator intr1
c_p_s1	DIGITAL	-	-	Comanda pornire separator 1
c_o_s1	DIGITAL	-	-	Comanda oprire separator 1
c_p_i1	DIGITAL	-	-	Comanda pornire intreruptor 1
c_o_i1	DIGITAL	-	-	Comanda oprire intrerupator 1

După acţionarea separatorului s1 de exemplu, nu se schimbă starea acestuia decât după apariţia confirmării. Va trebui să introducem un simbol care să indice comanda dată separatorului pentru a nu deruta utilizatorul, acesta având impresia ca nu se întâmplă nimic în urma acţionarii separatorului. După confirmarea acţionării se stinge simbolul aferent comenzii şi se schimbă starea separatorului. Aceste consideraţii sunt ilustrate în pagina grafica de mai jos.



In cazul în care se încearcă acţionarea separatorului în sarcină, nici măcar comanda de acţionare a separatorului nu este lansata. In acest caz, utilizatorul trebuie atenţionat cu un mesaj că nu poate acţiona separatorul. Mesajele vor fi trimise cu instrucţiunea:

Prompt("Mesaj");

Confirmările de realizare ale comenzilor sunt simulate de butoanele din partea dreapta a paginii grafice.
Butonul "Separator complet închis" are setată proprietatea "Imput" cu următoarea comandă:

```
c_p_s1=0
s1=1
```

Butonul "Separator complet deschis" are setata proprietatea "Imput" cu următoarea comandă:

```
c_o_s1=0
s1=0
```

Butonul "Confirmare închidere i1" are setata proprietatea "Imput" cu următoarea comandă:

```
c_p_i1=0
intr1=1
```

Butonul "Confirmare deschidere i1" are setata proprietatea "Imput" cu următoarea comandă:

```
c_o_i1=0
intr1=0
Prompt(" ");
```

Simbolul "s1" are setata proprietatea "Imput" cu următoarea comandă:

```
IF (intr1=0)
THEN
Prompt(" ");
 IF s1=0 THEN
   c_p_s1=1
   c_o_s1=0
  ELSE
   c_o_s1=1
   c_p_s1=0
 END
ELSE
Prompt("Nu se poate actiona separatorul!");
END
```

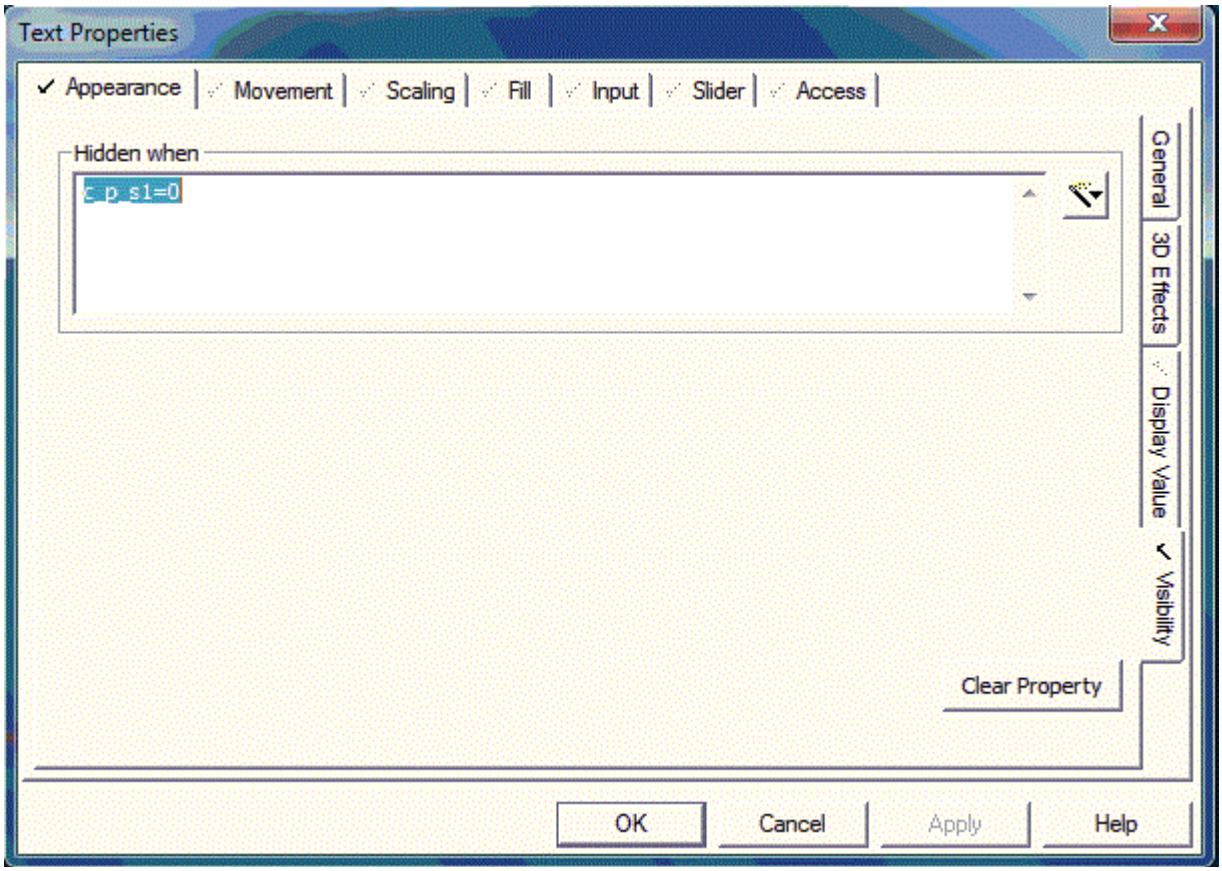
După cum se observă încercarea de a acţiona separatorul în sarcină are ca singur efect afişarea mesajului: "Nu se poate acţiona separatorul!"

Simbolul "intr1" are setata proprietatea "Imput" cu următoarea comandă:

```
IF intr1=0 THEN
   c_o_i1=0
   c_p_i1=1
  ELSE
   c_p_i1=0
   c_o_i1=1
 END
```

Cele patru simboluri de tip LED sunt simple "Set Simboluri" în concordantă cu tag-urile: c_o_s1, c_p_s1, c_o_i1, c_p_i1

In dreptul acestor simboluri apar şi textele corespunzătoare, texte care sunt vizibile atât timp cat tag-ul corespunzător este activ. Acest comportament s-a realizat prin setarea proprietăţii "Visibility"-> Hidden when

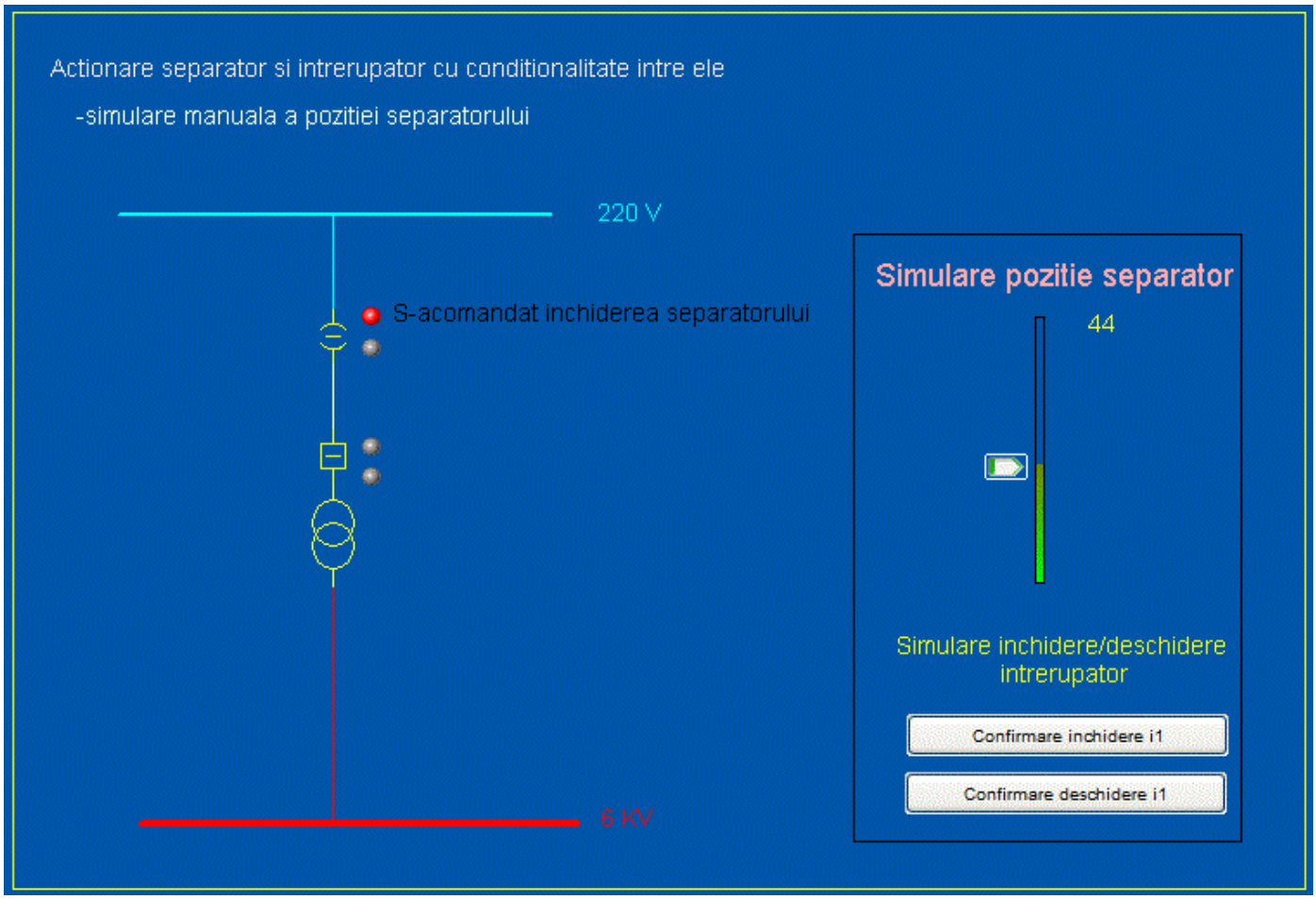


Simularea execuţiei comenzilor

Inchiderea respectiv deschiderea separatorului, ar putea fi simulată printr-o mărime analogică, având valori intre 0 si 100%. Pe schemă, această valoare ar putea fi simulată prin utilizarea unui control de tip "Slider" care ar fi în legătură cu un tag de tip analogic. Vom introduce un nou tag, Dealtfel în acest moment avem următoarele tag-uri:

Tag-uri aferente				
Nume	**Tip**	**Domeniu**	**Um**	**Comentariu**
s1	DIGITAL	-	-	Stare separator s1
intr1	DIGITAL	-	-	Stare intreruptor intr1
c_p_s1	DIGITAL	-	-	Comanda pornire separator 1
c_o_s1	DIGITAL	-	-	Comanda oprire separator 1
c_p_i1	DIGITAL	-	-	Comanda pornire intreruptor 1
c_o_i1	DIGITAL	-	-	Comanda oprire intreruptor 1
poz_s1	INT	100	%	Poziţie separator 1

Vom realiza deci următoarea pagină grafică **sch_monof_5** in care simularea închiderii respectiv deschiderii separatorului se va realiza manual de la un "Slider"

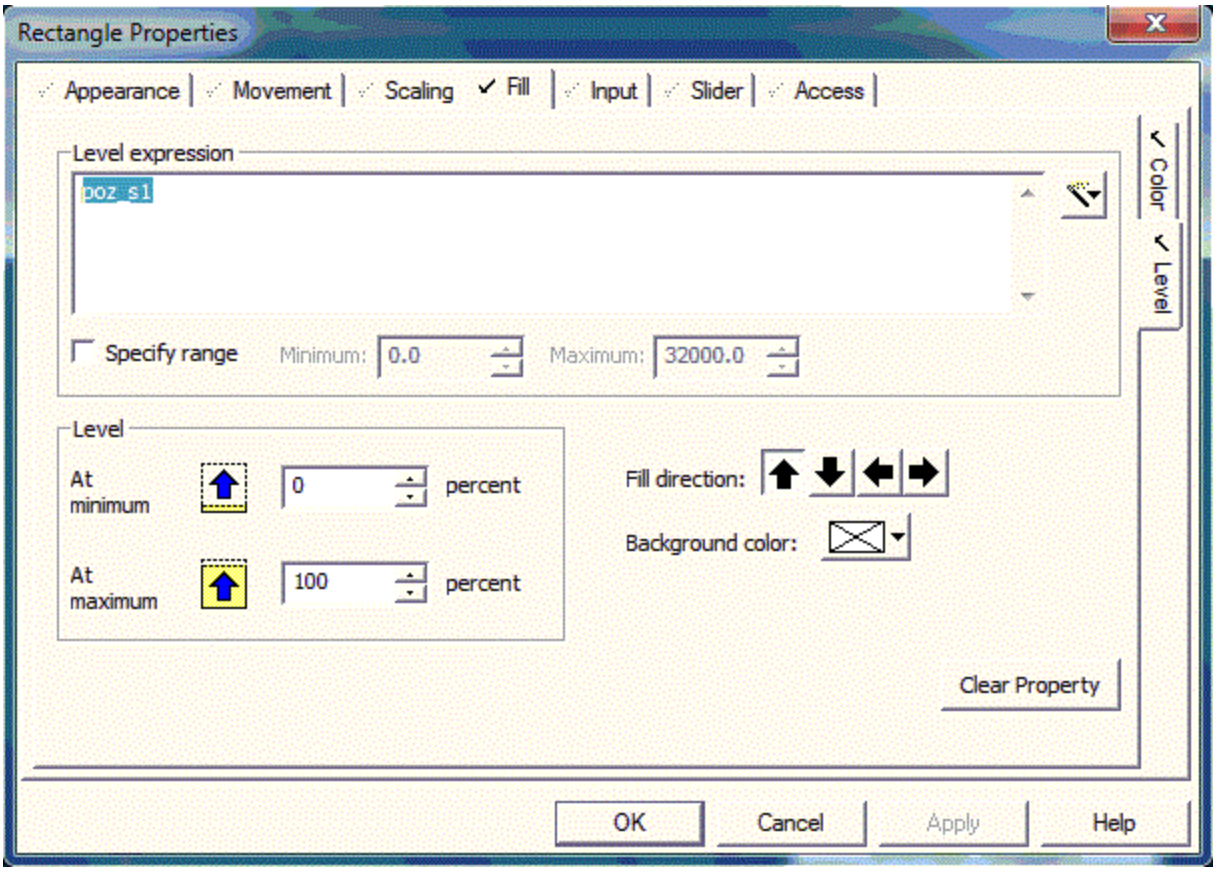


Simularea poziţiei separatorului se face utilizând un obiect "Symbol" de tipul Xp_slider -right normal care poate fi localizat în Library->XP_Sliders.

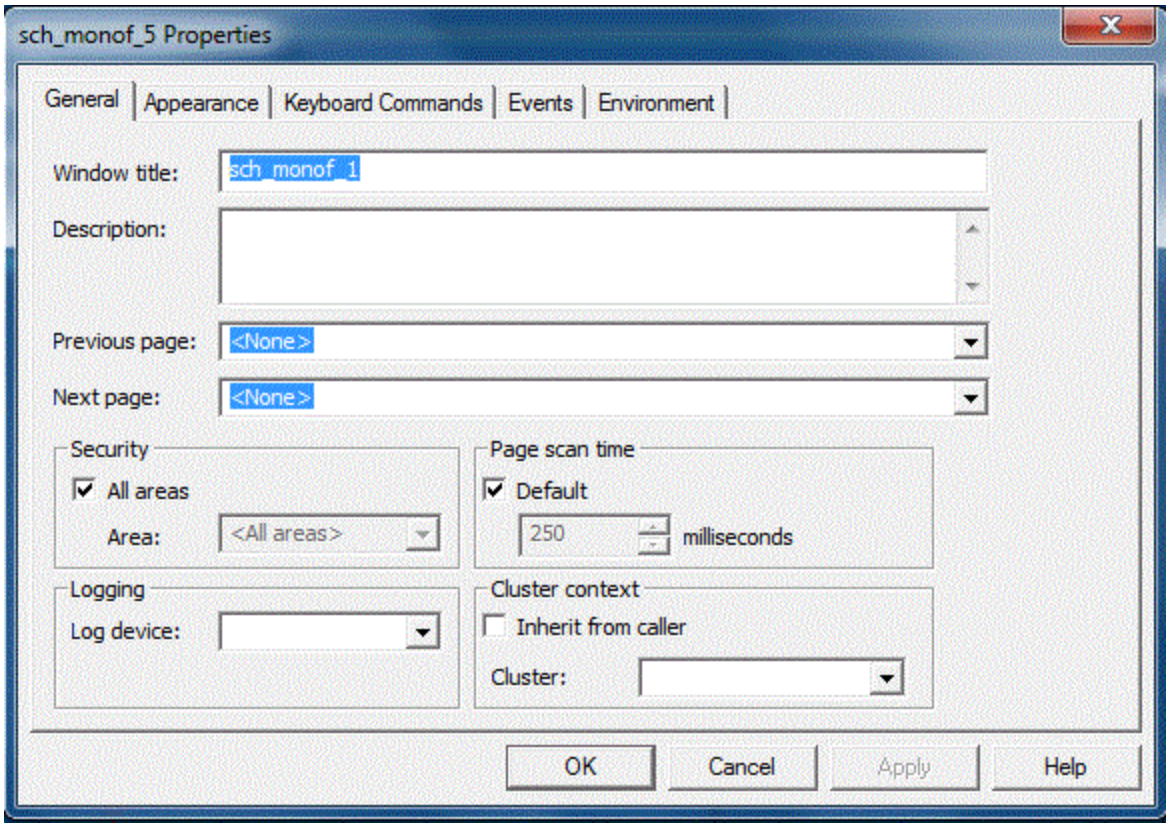
Setam proprietatea Slider->Vertical->Tag=poz_s1 si Offset at maximum = 150 pixeli. Setând această proprietate în acest mod, vom putea acţiona acest simbol şi sa-l deplasăm pe verticală 150 de pixeli. Mişcarea acestui obiect de la 0 la 150 pixeli va cauza modificarea tag-ului asociat poz_s1 de la 0 la val maxima definita adică 100.

Afişarea valorii poz_s1 sub forma numerică se face utilizând un obiest de tip "Number"

Reprezentarea sub forma unei bare verticale se face utilizând un obiect "Rectangle" in care s-a setat proprietatea: Fill->Level->Level expresion=poz_s1 iar proprietatea Apparence s-a setat cu Gradient fill.



După plasarea "Slider-ului", ar trebui sa urmărim în permanentă daca poz_s1 este egala cu 100 sau cu 0 pentru a confirma închiderea respectiv deschiderea completa a separatorului. Dacă analizam "Page->Properties",



observam ca "Page scan time" se face la fiecare 250 ms deci am putea plasa pe ecran o funcţie care s-ar lansa de 4 ori pe secunda.

Vom plasa deci pe pagină grafică funcţia **ecran_1( )** , şi vom edita în "Cicode Editor" următoarea funcţie:

```
FUNCTION ecran_1()

        IF (poz_s1=100) THEN
                c_p_s1=0
                s1=1
        END
        IF (poz_s1=0) THEN
                c_o_s1=0
                s1=0
        END

END
```

Următoarea pagină grafică **sch_monof_6** simularea închiderii respectiv deschiderii separatorului se va realiza automat fiind afişată pe controlul de tip "Slider"

Simularea automată se realizează prin rescrierea funcţiei **ecran_1( )** astfel încât poziţia sa se incrementeze după fiecare interval de scanare. Noua funcţie **ecran_2( )** are forma:

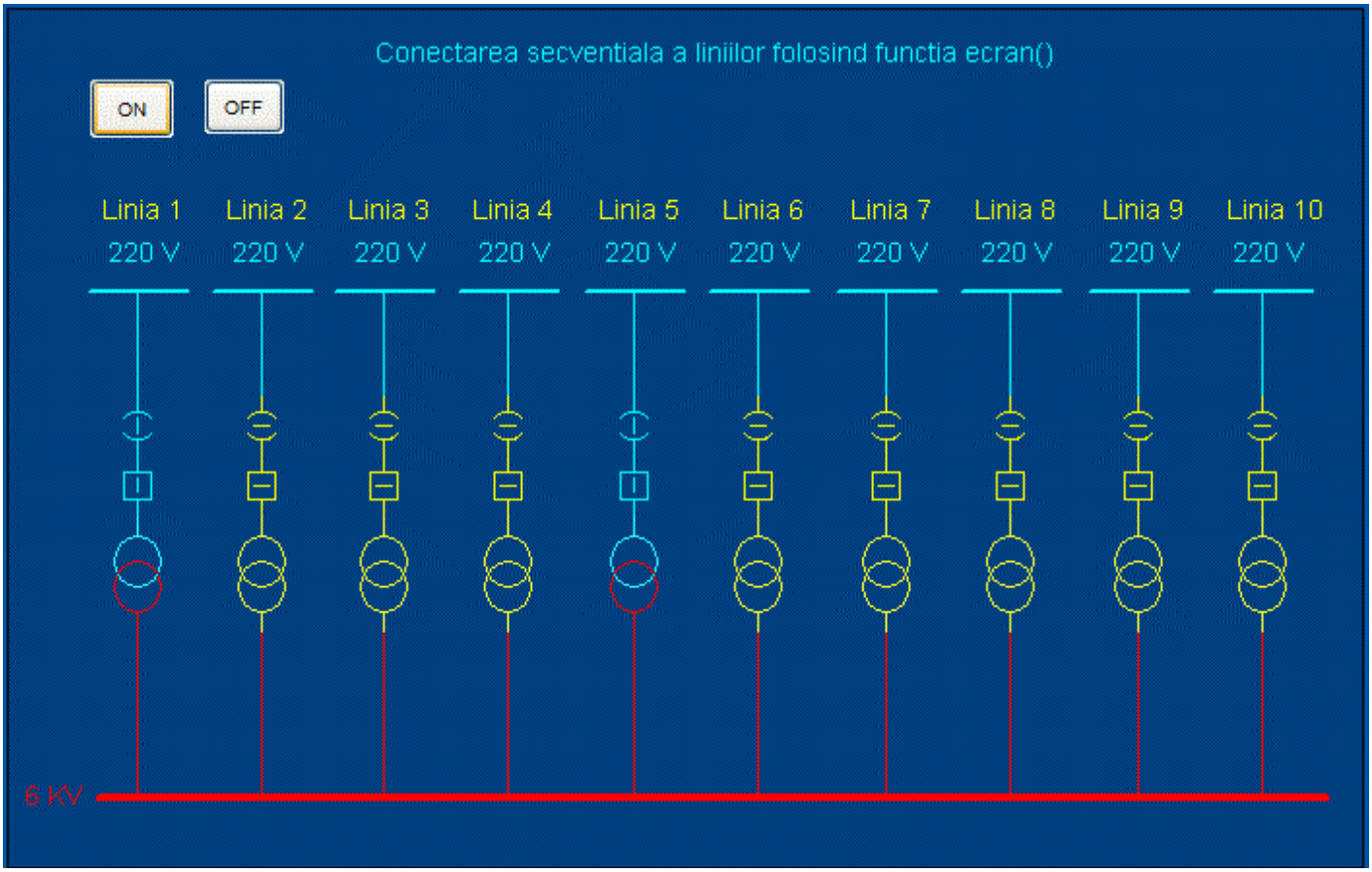
```
FUNCTION ecran_2()

	IF (c_p_s1=1) AND (poz_s1<100) THEN
		poz_s1=poz_s1+7
		IF (poz_s1>100) THEN
		  poz_s1=100
		END

	END
	IF (c_o_s1=1) AND (poz_s1>0) THEN
		poz_s1=poz_s1-7
		IF (poz_s1<0) THEN
		  poz_s1=0
		END


	END


	IF (poz_s1=100) THEN
		c_p_s1=0
		s1=1
	END
	IF (poz_s1=0) THEN
		c_o_s1=0
		s1=0
	END

END
```

3. Managementul liniilor monofilare

In multe cazuri în cadrul schemelor monofilare se întâlnesc mai multe elemente de acelaşi tip. In acest caz se recomandă definirea tag-urilor de tip Array si utilizarea instrucţiunilor repetitive.

Vom realiza în continuare diverse scheme monofilare în care avem mai multe linii de alimentare dotate fiecare cu transformator, intreruptor, separator etc.

Pentru început, să luăm o schema monofilară cu 10 linii separate de alimentare .



Sa reălizam o aplicaţie SCADA **sch_monof_7** care să comande închiderea respectiv deschiderea elementelor de comutaţie, într-o anumită ordine.

Pentru închiderea respectiv deschiderea elementelor de comutaţie, vom introduce două tag-uri de tip Array:

Tag-uri aferente					
Nume	**Tip**	**Domeniu**	**Um**	**Array Size**	**Comentariu**
sep	DIGITAL	-	-	11	Separatoarele sep[1]-sep[10]
intr	DIGITAL	-	-	11	Intreruptoarele intr[1]-intr[10]

Pe evenimentele click ale butoanelor "ON" respectiv "OFF" sunt afectate două variabile tag locale:

Tag-uri aferente				
Nume	**Tip**	**Domeniu**	**Um**	**Comentariu**
cmd_on	DIGITAL	-	-	Comanda inchiderea elementelor de comutatie
cmd_off	DIGITAL	-	-	Comanda deschiderea elementelor de comutatie
i	INT	-	-	Index

Pe evenimentul click al butonului "ON" vom pune: **toggle(cmd_on); i=0;**
Pe evenimentul click al butonului "OFF" vom pune: **toggle(cmd_off); i=0;**
Vom plasa funcţia **ecran()** pe pagina principală, funcţie lansată la fiecare scanare a ecranului.
Funcţie de variabilele de sus (cmd_on,com_off), la fiecare scanare a paginii, elementele de comutaţie se vor închide secvenţial, respectiv se vor deschide secvenţial. Deşi este o operaţie repetitivă, aceasta nu s-a mai implementat folosind instrucţiunea repetitiva **for**, repetitivitatea s-a realizat prin intermediul scanării implicite a paginii la intervalul stabilit anterior.

Ordinea de comutare sau deconectare a liniilor

Să presupunem acum că dorim activarea elementelor de comutaţie dintr-o schemă monofilară, într-o anumită ordine. Pentru acest lucru vom defini un vector de priorităţi de genul:

INT Prior[11]=0,1,5,7,4,2,9,3,6,8,10;

După cum se vede, s-au definit 11 elemente nu 10, pe motiv ca numerotarea LED-urilor începe de la 1 nu de la 0.
Conform modului de iniţializare al valorilor elementelor vectorului "Prior", ordinea de activare a elementele de comutaţie este:1,5,7,4,2,9,3,6,8,10
Vom rescrie funcţia **ecran_4( )** astfel:

```
INT Prior[11]=0,1,5,7,4,2,9,3,6,8,10;

FUNCTION ecran_4()
  IF (cmd_on=1) THEN
    cmd_off=0;
       i=i+1;
       IF (i=11) THEN
          cmd_on=0;
          i=0;
       END
    sep[Prior[i]]=1;
       intr[Prior[i]]=1;
  END

  IF (cmd_off=1) THEN
    cmd_on=0;
       i=i+1;
       IF (i=11) THEN
          cmd_off=0;
          i=0;
       END
       sep[Prior[11-i]]=0;
       intr[Prior[11-i]]=0;
  END
END
```

Dacă nu se utilizează funcţia **ecran( )**, vom scrie funcţia **sch_on( )** pentru butonul ON si funcţia **sch_off( )** pentru butonul OFF , funcţii în care vom folosi instrucţiuni repetitive astfel:

```
FUNCTION sch_on()
   INT i;
       FOR i=1 TO 10 DO
              sep[Prior[i]]=1;
              Sleep(1);
              intr[Prior[i]]=1;
              Sleep(1);
       END
END

FUNCTION sch_off()
   INT i;
       FOR i=1 TO 10 DO
              intr[Prior[11-i]]=0;
              Sleep(1)
              sep[Prior[11-i]]=0;
              Sleep(1);
```

```
        END
END
```

Dacă ţinem cont de faptul că la comutarea unei linii se comută prima data separatorul şi după aceea intreruptorul iar la deconectare, ordinea este inversă, funcţiile: **sch_on( )** respectiv **sch_off( )**

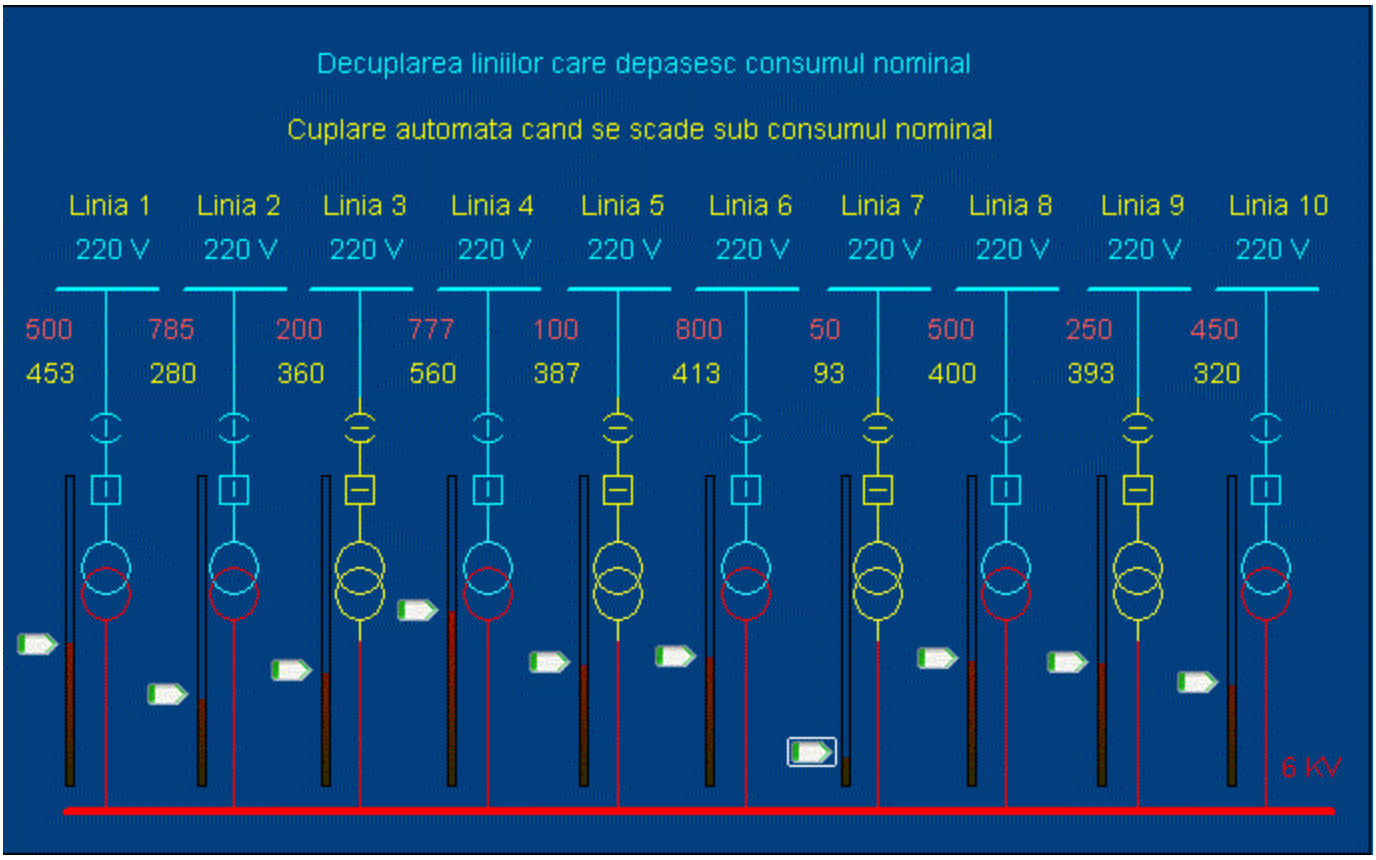
```
FUNCTION sch_on()
    INT i;
        FOR i=1 TO 10 DO
                IF intr[Prior[i]]=0 THEN
                        sep[Prior[i]]=1;
                END
                Sleep(1);
                intr[Prior[i]]=1;
                Sleep(1);
        END
END

FUNCTION sch_off()
    INT i;
        FOR i=1 TO 10 DO
                intr[Prior[11-i]]=0;
                Sleep(1)
                IF intr[Prior[11-i]]=0 THEN
                        sep[Prior[11-i]]=0;
                END
                Sleep(1);
        END
END
```

Protecţia liniilor la depăşirea parametrilor nominali

Vom simula în continuare protecţia maximală de curent în aplicaţia **sch_monof_10** . Fiecare linie de alimentare, are atribuita o valoare maxima a curentului admis. Pentru a memora valorile maximale de curent pentru fiecare linie, vom defini un nou vector tag de tip int **Imax** de dimensiune 11 (pentru fiecare linie cate o valoare).

Vom simula consumul efectiv pentru fiecare linie prin cate un simbol slider cărora le ataşam vectorul tag de tip int **i_ef** de dimensiune 11

Tag-uri aferente					
Nume	**Tip**	**Domeniu**	**Um**	**Array Size**	**Comentariu**
Imax	INT	-	-	11	Curent maxim pe liniile L1-L10
i_ef	INT	-	-	11	Curent efectiv pe liniile L1-L10



Dacă valoarea efectivă a curentului depăşeşte valoarea maximă prescrisă, trebuie deconectată linia, iar după ce valoarea efectivă a curentului scade sub valoarea maximală, linia se conectează automat. Acestefuncţii vor fi realizate de funcţia **ecran( )** funcţie care se lansează la fiecare scanare a ecranului.

```
FUNCTION ecran()
Imax[0]=0;
Imax[1]=500;
Imax[2]=785;
Imax[3]=200;
Imax[4]=777;
Imax[5]=100;
Imax[6]=800;
Imax[7]=50;
Imax[8]=500;
Imax[9]=250;
Imax[10]=450;
INT i;
FOR i=1 TO 11 DO
       IF i_ef[i]>Imax[i] THEN
              sep[i]=0;
              intr[i]=0;
       ELSE
              sep[i]=1;
              intr[i]=1;
```

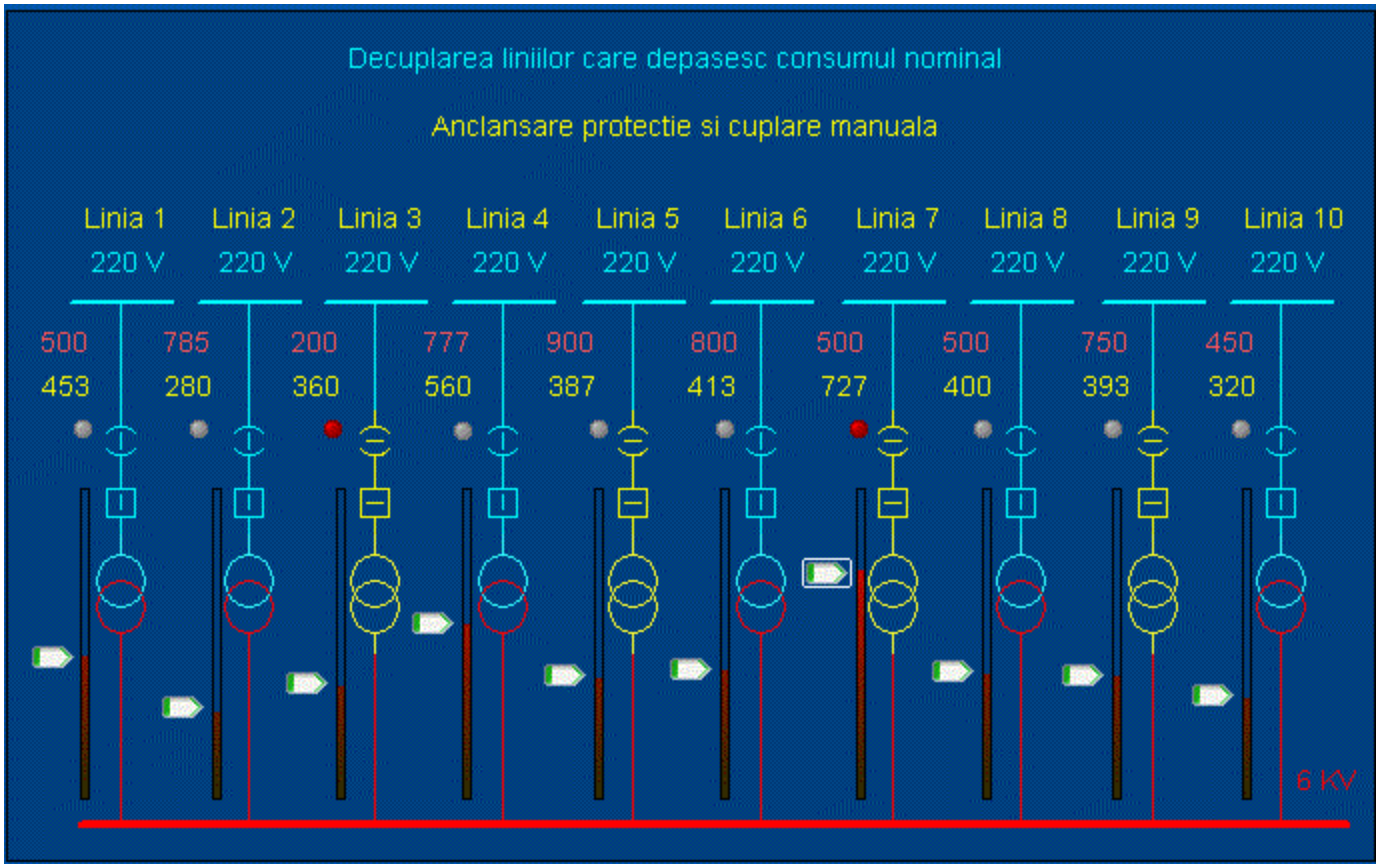
```
        END
END
END
```

Reconectarea liniei care a depăşit consumul, nu ar trebui să se facă automat, ar trebui sa se facă manual, numai după ce a fost redus consumul sub limita maximă. Ar trebui însa sesizată anclanşarea protecţiei, sesizare care va fi făcută prin intermediul unui indicator de tip LED. De asemenea LED-ul ar trebui să se invalideze în cazul ca nu mai există condiţii ca linia să intre din nou în protecţie.

Pentru controlul LED-urilor avem nevoie de un nou vector tag de tip int **ld_p** de dimensiune 11

Tag-uri aferente					
Nume	**Tip**	**Domeniu**	**Um**	**Array Size**	**Comentariu**
ld_p	DIGITAL	-	-	11	LED semnalizare protectie maximala de curent pe liniile L1-L10

In următoarea aplicaţie: **sch_monof_11** sunt puse în practică precizările de mai sus



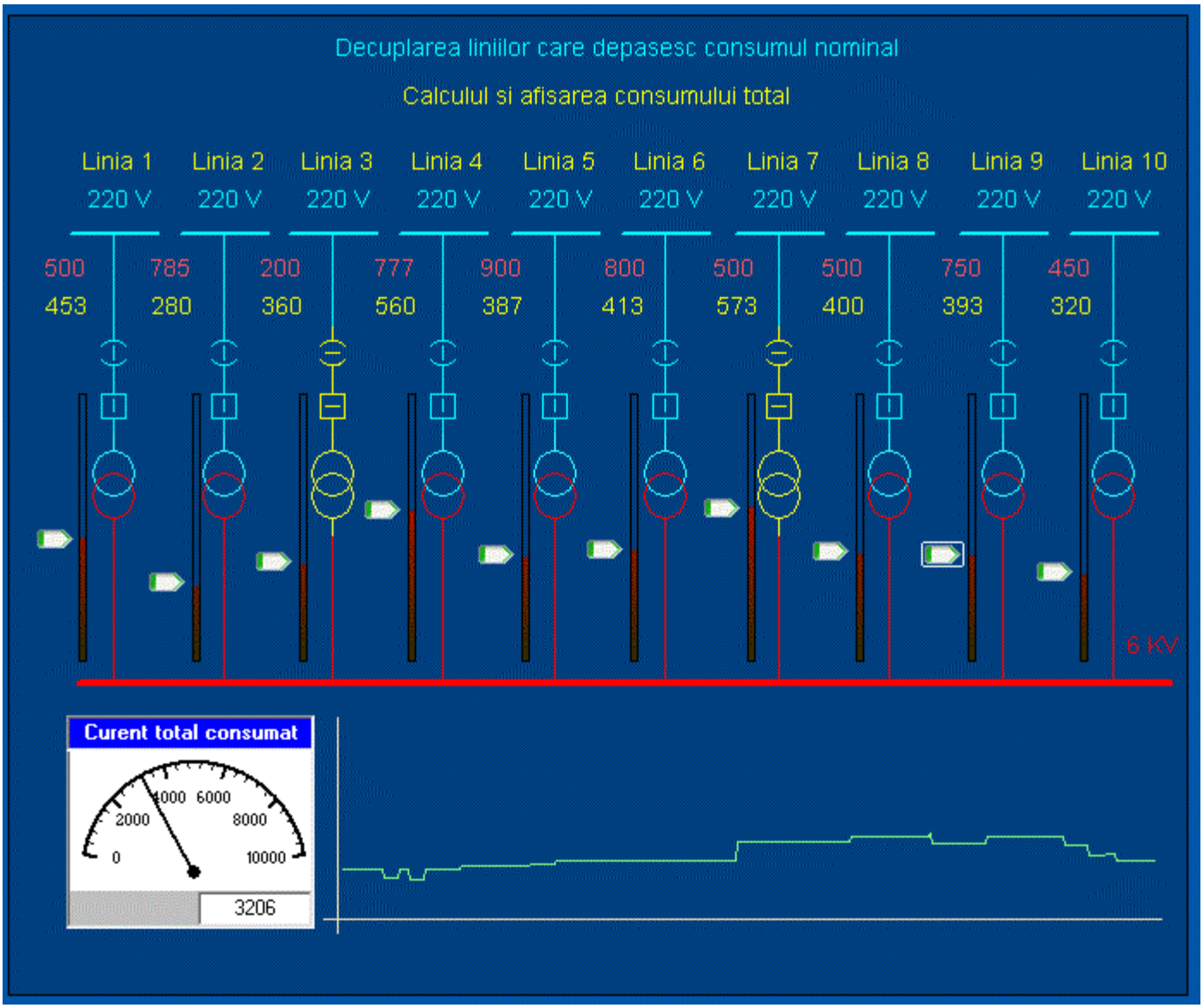
Pentru a implementa noul algoritm de funcţionare, vom rescrie funcţia ecran astfel:

```
FUNCTION ecran_11()
Imax[0]=0;
Imax[1]=500;
Imax[2]=785;
Imax[3]=200;
Imax[4]=777;
Imax[5]=900;
Imax[6]=800;
Imax[7]=500;
Imax[8]=500;
Imax[9]=750;
Imax[10]=450;

INT i;
i_tot=0;
FOR i=1 TO 11 DO

       IF i_ef[i]>Imax[i] THEN
              sep[i]=0;
              intr[i]=0;
              ld_p[i]=1
       ELSE
              ld_p[i]=0
              //sep[i]=1;
              //intr[i]=1;
       END
END
END
```

Pentru a monitoriza consumul total de curent vom plasa un obiect de tip X-Meter cat şi un obiect de tip Trending.

Va trebui să introducem un nou tag pentru a păstra consumul total efectiv şi obţinem astfel noua aplicaţie: **sch_monof_12** .

Tag-uri aferente				
Nume	**Tip**	**Domeniu**	**Um**	**Comentariu**
i_tot	INT	10000	Amps	Curent efectiv total



Funcţia ecran va fi cea care va prelua calculul curentului total: Trebuie ţinut cont de faptul că însumarea curenţilor se face numai pentru liniile conectate.

```
FUNCTION ecran_12()
Imax[0]=0;
Imax[1]=500;
Imax[2]=785;
Imax[3]=200;
Imax[4]=777;
Imax[5]=900;
Imax[6]=800;
Imax[7]=500;
Imax[8]=500;
Imax[9]=750;
```

```
Imax[10]=450;

INT i;
i_tot=0;
FOR i=1 TO 11 DO
   // se insumeaza numai liniile conectate
      IF sep[i]*intr[i]=1 THEN
            i_tot=i_tot+i_ef[i];
      END

IF i_ef[i]>Imax[i] THEN
      sep[i]=0;
      intr[i]=0;
      ld_p[i]=0
ELSE
      ld_p[i]=0
END
END
END
```

Întregul proiect poate fi descarcat aici : Download - "Sch_monof"

Test de autoevaluare

- -Marcaţi răspunsurile corecte la întrebările următoare.
- -ATENTIE: pot exista unul, niciunul sau mai multe răspunsuri corecte la aceeaşi întrebare.
- -Timp de lucru: 10 minute

1. Cine controlează starea elementelor plasate pe o schema monofilară ?

☐a. Simbolurile plasate
☐b. Tag-urile asociate
☐c. Variabilele locale utilizate
☐d. Funcţiile definite de utilizator

2. Pentru a introduce funcţionalităţi de comandă pentru elementele de pe schema, se setează proprietatea:

☐a. Access
☐b. Input
☐c. Slider

☐d. Movement

3. Care este condiţionalitatea intre acţionarea separatorului şi intreruptorului ?

☐a. Primul se acţionează întotdeauna intreruptorul
☐b. Primul se acţionează întotdeauna separatorul
☐c. Separatorul nu se acţionează in sarcină
☐d. Intreruptorul nu se acţionează in sarcină

4. Ce reprezintă confirmarea execuţiei comenzilor ?

☐a. Apariţia unui eveniment
☐b. Apariţia unui semnal de confirmare
☐c. Modificarea valorii unui tag
☐d. Scurgerea unui interval de timp

5. De ce e nevoie de conectare secvenţială a liniilor ?

☐a. Pentru a evita şocurile de sarcină
☐b. Pentru a diferenţia consumatorii in funcţie de importanta
☐c. Pentru implementarea nivelelor de prioritate
☐d. Pentru a putea urmări conectarea fiecărei linii în parte

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

Rezumat

Aplicaţii in energetică - linii monofilare

Pentru a realiza scheme monofilare avem nevoie de o serie de simboluri cum ar fi simboluri pentru: separatoare, intreruptoare, transformatoare etc.

O schemă monofilară conţine o serie de elemente de comutaţie cum ar fi separatoare, intreruptoare, etc. Există o ordine în care se acţionează cele doua elemente, în sensul ca separatorul nu poate fi niciodată acţionat in sarcină

In sistemele energetice, închiderea respectiv deschiderea unui separator sau a unui intreruptor nu este instantanee. In aplicaţiile SCADA se tine cont de acest fenomen. După comanda acţionarii unui element se scurge un interval de timp pana la efectuarea completa a comenzii. După acţionarea unui element e nevoie de un semnal care sa confirme efectuarea comenzii.

Pentru testarea aplicaţiilor SCADA fără conectare la sistemul real de achiziţie şi comandă, închiderea respectiv deschiderea elementelor de comutaţie, ar putea fi simulată printr-o mărime analogică, având valori intre 0 si 100%. Pe schemă, această valoare ar putea fi simulată prin utilizarea unui control de tip "Slider"

care ar fi în legătură cu un tag de tip analogic.

Starea fiecărui element plasat pe HMI este controlată de tag-ul corespunzător.

In multe cazuri în cadrul schemelor monofilare se întâlnesc mai multe elemente de acelaşi tip. In acest caz se recomandă definirea tag-urilor de tip Array si utilizarea instrucţiunilor repetitive.

In cazul sistemelor de alimentare cu energie, având mai multe linii, pentru a evita şocurile de sarcină, conectarea respectiv deconectarea acestora se face secvenţial într-o ordine prestabilita, în funcţie de importanta utilizatorului. Pentru a memora prioritatea consumatorilor, se recomandă utilizarea unui vector de priorităţi.

Tag-urile sunt actualizate de sistemul de achiziţie şi comandă. Acesta este prevazut atât cu elemente pentru citirea stărilor cat şi cu elemente de comandă

Rezultate aşteptate

După studierea acestui modul, ar trebui sa ştiţi:

- Să realizaţi pagini grafice reprezentând scheme monofilare
- Să daţi funcţionalitate schemelor monofilare
- Să implementaţi algoritmul de funcţionare al elementelor de comutaţie incluse în aplicaţii SCADA
- Să realizaţi aplicaţii SCADA care să cuprindă scheme monofilare complexe.

Termeni esenţiali

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
Limbaj Cicode	Limbaj de programare inclus in mediul de dezvoltare Citect SCADA
Sistem de achiziti date	Sistem fizic care interfateaza sistemele energetice cu aplicatiile SCADA. Acesta este prevazut atat cu elemente pentru citirea strilor elementelor componente ale sistemului energetic cat si cu elemente de comanda
Elemente de comutatie	Elemente componente ale retelelor de alimentare cu energie care permit inchiderea respectiv deschiderea unui circuit

-

Recomandări bibliografice

- [1] Traian Turc, Elemente de programare C++ utile in ingineria electrica, Ed.Matrixrom, Bucuresti,2010
- [2] Traian Turc, Programare avansata C++ pentru ingineria electrica, Ed.Matrixrom, Bucuresti,2010
- [3] Traian Turc, Programarea in limbaje de asamblare, uz intern, Univ."Petru Maior" ,Tg.Mures,2009
- [4] Traian Tur,Brevet de inventie 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 industriala - 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 instrumentatie 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

- -Marcaţi răspunsurile corecte la întrebările următoare.
- -ATENTIE: pot exista unul, niciunul sau mai multe răspunsuri corecte la aceeaşi întrebare.
- -Timp de lucru: 10 minute

1. Un tag care gestionează un simbol reprezentând un separator are tipul:

☐a. INT
☐b. DIGITAL
☐c. BYTE
☐d. BCD

2. Prescrierea curentului nominal pentru o linie se face utilizand:

☐a. Variabile locale
☐b. Vectori de valori
☐c. Tag-uri
☐d. Funcţii definite de utilizator

3. După decuplarea unei linii care a depăşit curentul nominal, recuplarea acesteia se face:

☐a. Automat după ce valoarea curentului scade după curentul nominal
☐b. După acţionarea manuală
☐c. După un algoritm prestabilit în program
☐d. După un interval de timp prestabilit

4. Calculul consumului total se face:

☐a. Utilizând un vector
☐b. Utilizând un tag
☐c. Utilizând un algoritm de calcul
☐d. Utilizând un obiect de tip Trending

5. *Un vector care păstrează curenţii nominali se defineşte:*

☐a. Din program
☐b. Prin introducerea unu tag de tip Array
☐c. Prin utilizarea unui obiect de tip multitreding
☐d. Prin utilizarea mai multor variabile

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