Automatele programabile din familia SIMATIC

1. Sistemul de automatizare SIMATIC

Sistemul de automatizare SIMATIC este reprezentat de componente coordonate cu metode unitare de configurarea, înregistrarea şi transmiterea datelor.

Automatele programabile din familia SIMATIC (S7) reprezintă baza sistemului de automatizare. Cele 3 tipuri aflate pe piață sunt: S7-200 un automat de mici dimensiuni folosit în automatizări ale unor procese mai simple (obiecte casnice de exemplu), S7-300 este un automat pentru automatizări medii, iar S7-400 reprezintă soluția pentru automatizări complexe. Un astfel de automat este format din unitatea centrală şi modulele de I/O. Unitatea centrală_are încărcat programul utilizator în timp ce modulele de I/O asigură comunicarea cu procesul care este controlat. Celelalte componente din soluția SIMATIC pentru automatizări, cum ar fi (C7, DP, C7, HMI, NET), completează automatul programabil reprezentând o consolă specializată de programare, module de I/O distribuite - pentru o comandă la distanță- modul pentru conectare în rețea etc.

Limbajul acestui concept de automatizare totală este STEP7 care este utilizat pentru configurarea componentelor SIMATIC, pentru a le atribui parametri şi nu în ultimul rând pentru a le programa. Unealta software centrală pentru control este reprezentată de SIMATIC Manager care păstrează toate datele unui proiect de automatizare într-un director cu o structură ierarhică şi permite reutilizarea softului de utilizator prin librării. Principalele activități realizate de STEP7 sunt:

- configurarea hardware-ului – ceea ce reprezintă aranjarea modulelor, atribuirea de adrese , precum şi setarea proprietăților acestora;

- configurarea parametrilor de comunicare precum şi a proprietăților acesteia;

- scrierea de programe utilizator pentru automatul programabil într-unul dintre cele 3 moduri folosite: Ladder Logic (LAD), Function Block Diagram (FBD) sau Statement List (STL), precum şi testarea online a acestora pe automat. Soluția SIMATIC pune la dispoziție şi pachete software opționale care pot extinde funcționarea uneltelor standard din STEP7.

2. Considerații privind alegerea soluției de automatizare

2.1 Alegerea hardware-ului

Există mai multe criterii pentru alegerea tipului de automat programabil. Pentru aplicații de nivel redus criteriul cel mai important este reprezentat de numărul de intrări şi ieşiri precum şi de dimensiunea programului utilizator. În cazul proceselor mai complexe trebuie avut în vedere dacă răspunsul în timp este suficient de rapid, dacă memoria este suficientă pentru volumul de date care urmează să fie înmagazinat.

O maşină unealtă va fi probabil comandată prin intermediul unui singur automat programabil. În acest caz numărul de intrări/ieşiri, dimensiunea memoriei, răspunsul în timp vor fi esențiale pentru alegerea înte una din variantele S7-200, S7-300 sau S7-400. În cazul proceselor răspândite în mai multe locații este mai util folosirea unor module de I/O distribuite decât a unor module dispuse pe automat. Aceasta nu numai că reduce lungimea unor cabluri de conectare cu procesul, ci poate şi indica viteza de răspuns a automatului. Soluția unei automatizări distribuite are şi alte avantaje: programele utilizator pentru diferitele părți ale procesului sunt mai scurte şi pot fi (în general) rulate independent de restul procesului. Totodată schimbul necesar de date între automat şi proces este în mod esențial mai facil dacă este utilizat modul de comunicare în rețea (SIMATIC NET).

2.2 Alegerea limbajului de programare

Alegerea limbajului de programare depinde de utilizator precum şi de complexitatea taskului care trebuie realizat. În cazul prelucrării semnalelor binare sunt mai uşor de folosit LAD şi FBD, în timp ce în cazul taskurilor care cer mânuirea variabilelor complexe şi adresarea indirectă poate fi utilizat STL. Personal recomand utilizarea STL datorită faptului că este familiar celor care programează la un nivel mai înalt, precum şi procesării unui volum mare de date.

2.3 Crearea unui proiect

Toate datele care oferă soluția de automatizare sunt colectate împreună într-un proiect. Acesta va fi creat utilizând STEP7. Datele sunt structurate ierarhic. Următorul nivel după proiect este reprezentat de "stații" care conțin una sau mai multe unități centrale cu un program utilizator. Pot fi utilizate comenzi de meniu care inserează obiecte noi, pot fi deschise aceste obiecte pentru a fi configurate.

2.4 Scrierea, analiza şi salvarea unui program

Programul utilizator conține toate instrucțiunile folosite de către programator pentru procesarea semnalelor pentru controlul maşinii sau procesului pentru a realiza taskul cerut. Împărțirea programului în blocuri de instrucțiuni poate fi orientată pe proces, caz în care fiecare funcție (bloc de funcții) corespunde unei părți a procesului sau maşinii, respectiv orientată funcțional, caz în care funcțiile corespund comunicării, modurilor de operare etc. Programele sunt editate şi testate. Testarea acestora se poate face după încărcarea lor pe automatul programabil sau pe softul de testare PLCSIM. După testarea softului acesta este încărcat într-o memorie EPROM şi este generată documentația proiectului utilizând DOCPRO. Proiectul poate fi salvat sub forma unui fişier comprimat.

3. Platforma hardware a unei automatizări SIMATIC. Scurtă prezentare

3.1 Componentele unei stații de automatizare SIMATIC

Considerăm o "stație" de automatizare SIMATIC un automat programabil din această familie împreună cu modulele de I/O. Componentele unei astfel de stații sunt:

- şina de montare – asigură conectarea modulelor individuale. Automatele S7-300 utilizează o şină simplă, lungimea sa fiind determinată de numărul modulelor. S7-400 folosesc o şină de aluminiu de lungime fixă prevăzută cu conectori pentru magistrală.
- sursa de alimentare – asigură alimentarea întregii stații de automatizare

- unitatea centrală – înmagazinează şi execută programul utilizator, atribuie parametri modulelor, realizează comunicația între dispozitivul de programare, module, stații adiționale prin intermediul magistralei
- module de interfață – conectează şinele de montare între ele
- module de I/O – face adaptarea semnalele din proces sau către proces
- module funcționale – realizează diverse funcții care nu pot fi "acoperite" de catre unitatea centrală (control)
- procesor pentru comunicare – folosit în momentul în care se doreşte conectarea în subrețele

Sunt folosite două tipuri de magistrale: una pentru transmiterea spre/ dinspre modulele de I/O, respectiv una pentru comunicarea rapidă cu un volum mai mare de date între unitatea centrală şi celelalte module. Modulele de I/O pot fi locale sau distribuite. Cele distribuite pot fi de tip master sau slave dacă sunt plasate undeva în câmpul procesului. Şi modulele distribuite sunt văzute ca şi cele locale având alocate adrese şi neexistând practic diferențe între cele 2 tipuri din punct de vedere al unității centrale.

3.2 Unități centrale SIMATIC

La ora actuală se folosesc 3 familii de unități centrale pentru automatizări SIMATIC:

1. S7-200:

- limbaj de programare STEP7 Micro
- folosit pentru automatizări mici, mai multe module de extindere, posibilități de conectare în rețea. Numărul de module de I/O poate fi ridicat astfel încât să satisfacă cerințele legate de proces. O interfață de tipul punct cu punct permite conectarea mai multor unități centrale împreună (până la 31), precum şi conectarea cu alte automate din familia SIMATIC.

2. S7-300:

- automatizări medii, aceleaşi caracteristici ca la S7-200 şi în plus posibilitate de instalare a sursei pe şina centrală. Sloturile sunt numerotate: 1 pentru sursă (chiar dacă lipseşte), 2 pentru unitate centrală, 3 – modul de interfață, 4-11 – module de I/O.

Este prevăzut cu module pentru lucru în mediu cu condiții ostile (temperaturi foarte ridicate sau scăzute, un nivel ridicat de vibrații, o rezistență la şocuri).

3. S7-400:

- automatizări complexe;
- îmbunătățeşte performanțele stațiilor din familia 300 având o mai mare capacitate de procesare a informației
- posibilitatea de conectare a două unități centrale la aceeaşi sursă, capacitate de multiprocesare.

Componentele din standardul SIMATIC S7-300/400 permit un sistem redundant de automatizare în cazul proceselor lente, astfel că o stație poate prelua controlul procesului în cazul în care o altă stație (master de exemplu) cade. În timpul acestei perioade toate semnalele din proces sunt "înghețate". Un sistem complet C7 presupune pe lângă unitatea centrală şi modulele de I/O şi un panou operator prin intermediul căruia operatorul poate interacționa cu sistemul de automatizare.

4. Limbajul de programare STEP7

Principala cerință pentru limbajul de programare a unui automat programabil este aceea de a fi uşor de înțeles şi utilizat în aplicații de conducere a proceselor. Ăcest lucru implică nevoia unui limbaj înalt pentru a furniza comenzi foarte apropiate de funcțiile cerute de către un inginer automatist, dar fără a fi complex şi a necesita un timp de învățare mare.

Există mai multe limbaje şi metode de programare care pot fi utilizate în cazul programării automatelor din familia SIMATIC după cum satisfac una dintre cerințele programatorilor. Trei dintre acestea (LAD, FBD şi STL) sunt incluse în pachetul STEP7 iar altele pot fi achiziționate ca pachete adiționale.

4.1 Programarea în LAD, FBD şi STL. Concepte de bază

Mediul STEP7 include un editor pentru cele trei moduri de programare. LAD şi FBD sunt limbaje grafice în timp ce STL se bazează pe listă de instrucțiuni.

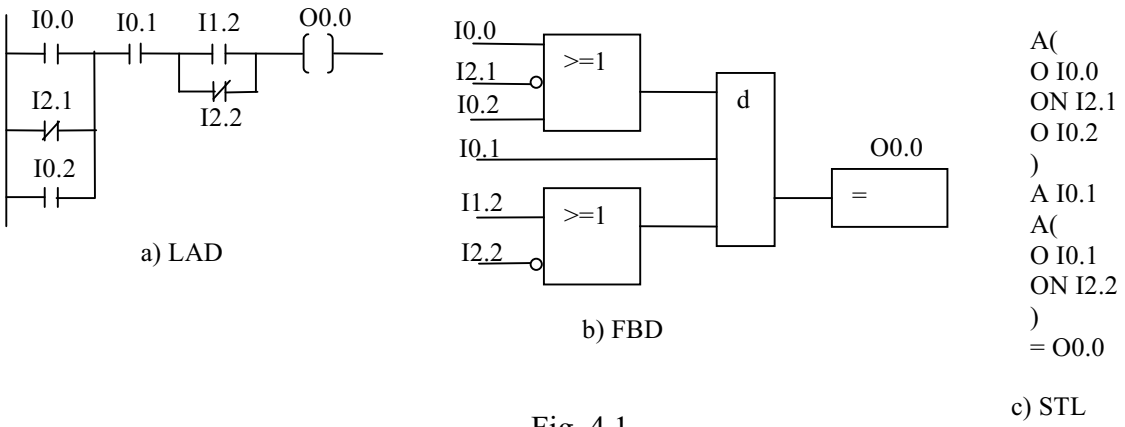

Fig. 4.1

După cum se vede în figura 4.1 în LAD putem realiza programe prin conectarea în serie sau paralel a diferitelor intrări , iar în FBD prin folosirea simbolurilor pentru funcțiile logice ŞI, SAU, NOT. LAD sau diagramele Ladder au reprezentat cea mai obişnuită metodă de descriere a circuitelor logice cu relee, fiind utilizate inițial pentru reprezentarea circuitelor electrice, iar ulterior s-au impus şi la automate programabile din dorința de a asigura utilizatorului aceleaşi facilități.

Spre deosebire de primele două metode, în STL este practic o programare de tipul limbajului de asamblare folosit în cazul microprocesoarelor. Programul în acest caz apare sub forma unei liste de instrucțiuni, fiecare linie definind funcția care urmează să fie realizată şi, dacă este cazul, o adresă de la care funcția să fie apelată.

Programul poate fi editat în oricare dintre modurile alese existând posibilitatea şi de a transfera rutinele dintr-un tip de programare în altul. Cele trei moduri nu presupun doar o procesare la nivel de bit a informației, existând posibilitatea de manipulare şi la nivel de octet, cuvânt etc. Pentru taskurile mai complexe există posibilitatea folosirii funcțiilor matematice, de conversie, deplasare, a salturilor. Programele sunt editate în forma unor blocuri. Blocurile organizaționale reprezintă interfața între sistemul de operare de pe unitatea centrală şi programul utilizator. În momentul apariției unui eveniment sistemul de operare de pe unitatea centrală apelează aceste blocuri care marchează începutul programului folosind diverse clase de prioritate sau nivele de execuție. Blocurile de tipul celor de funcții sau a funcțiilor pot fi apelate din cadrul blocurilor organizaționale şi apoi executate.

4.2 Programarea folosind LAD (diagramele Ladder)

În LAD programarea se face prin aranjarea elementelor grafice ale programului. Programarea este organizată pe rețele în cadrul cărora sunt poziționate contacte, bobine (analogie cu diagramele electrice) sau cutii. Majoritatea elementelor au nevoie de o identificare prin adresă (I0.3) sau etichetă (buton avans), nefiind permisă o realizare de tip paralel legată de altă rețea.

Contactele sunt folosite ca simboluri pentru intrări, iar prin aranjarea acestora în serie sau paralel pot fi realizate operații logice asupra stării semnalelor. În fig. 4.2 sunt prezentate tipurile de contacte. Cele normal deschise sunt testate pentru valoarea 1 a semnalului reprezentat de contactul respectiv, în timp ce cele normal închise sunt testate pentru valoarea 0 a semnalului de intrare.

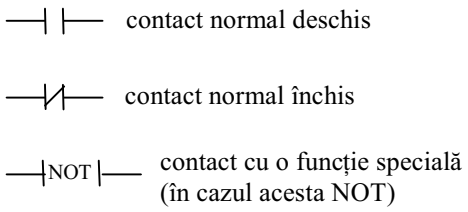

Fig. 4.2 Tipuri de contacte

În al treilea caz sunt prezentate contacte care nu sunt doar citite ci reprezintă biții de stare care, după ce le-a fost citită valoarea, sunt readuşi la valoarea inițială.

Bobinele sunt în general atribuite ieşirilor. În cazul celor simple bitul de la adresa bobinei este setat dacă rezultatul e 1, în timp ce în cazul bobinelor cu litere sau simboluri apare o funcționare adițională (salt într-un alt loc în program, controlul timerului, funcții de numărare etc.).

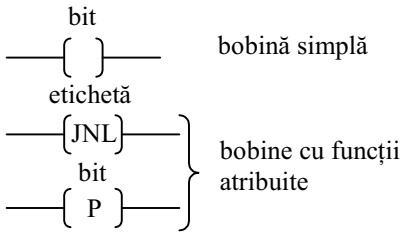

Fig. 4.3 Tipuri de bobine

Cutiile sunt folosite în cazul elementelor de program fără o funcționare binară. Acestea pot fi cu validare sau fără (EN, ENO) şi pot reprezenta mai multe tipuri de funcții (fig. 4.4).

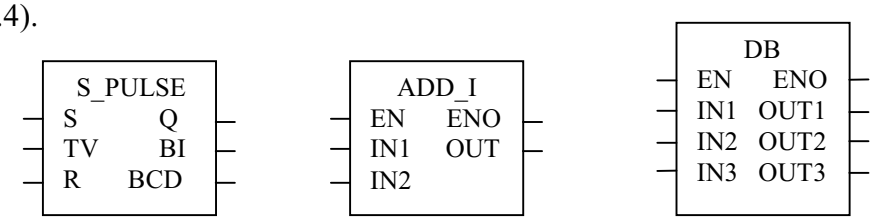

Fig. 4.4. Elemente de programare (cutii) în LAD

Parametri de tip validare (EN, ENO) reprezintă validarea intrărilor, respectiv a ieşirilor (în cazul în care funcția a fost realizată fără eroare). Datorită acestor parametriputem conecta în serie aceste funcții şi să ne asigurăm în acelaşi timp că o funcție este realizată doar în cazul în care sunt realizate şi cele dinaintea ei.

Conectarea în serie sau în paralel a elementelor prezentate se poate face ca în figură:

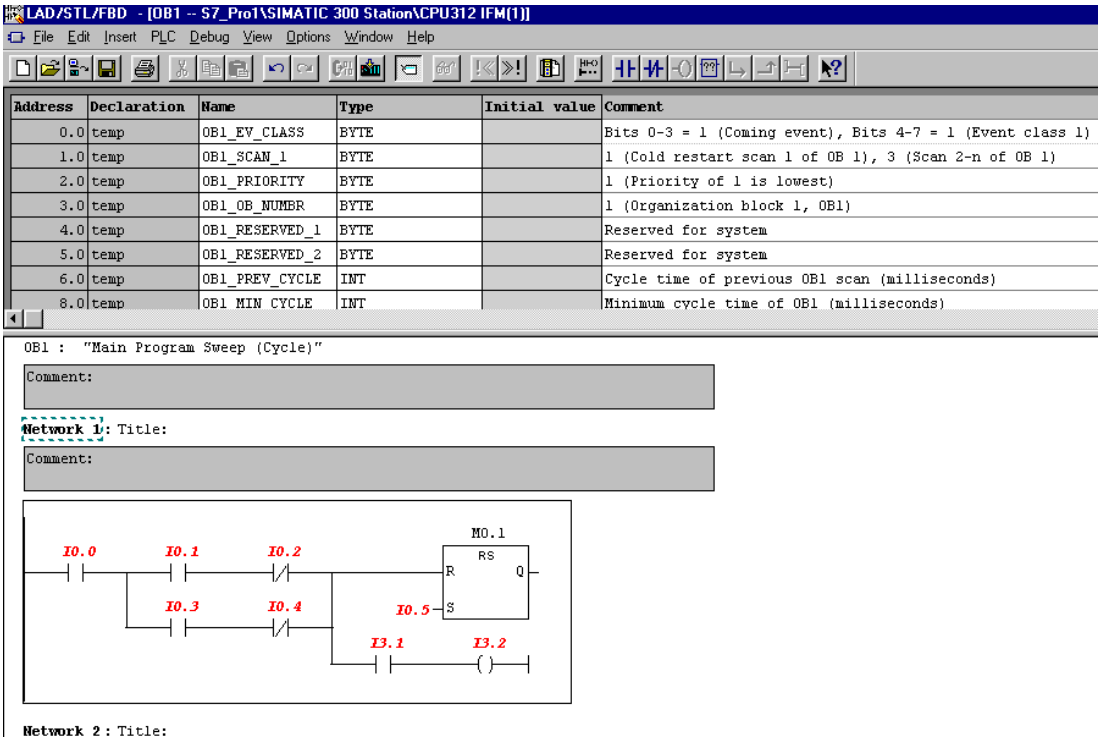

Figura 4.5

4.3 Programarea folosind FBD (function block diagram)

Este tot un mod grafic de programare prin conectarea mai multor cutii cu simboluri. Metoda de programare este apropiată de LAD fiind chiar identică în cazul cutiilor (funcții numerice – fig. 4.4). Funcțiile binare sunt reprezentate ca în fig. 4.6 a), în timp ce bobinele sunt înlocuite de cutii simple ca şi cele din fig. 4.6 b). Totodată şi folosirea parametrilor de tipul validare (EN/ENO) au aceeaşi utilitate ca în cazul diagramelor LAD. În fig. 4.7 am prezentat două scheme realizate in FBD.

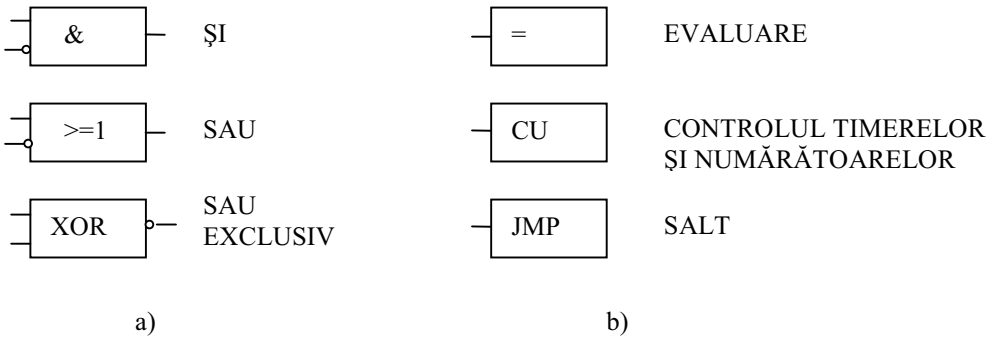

Fig. 4.6 Elemente de programare în FBD

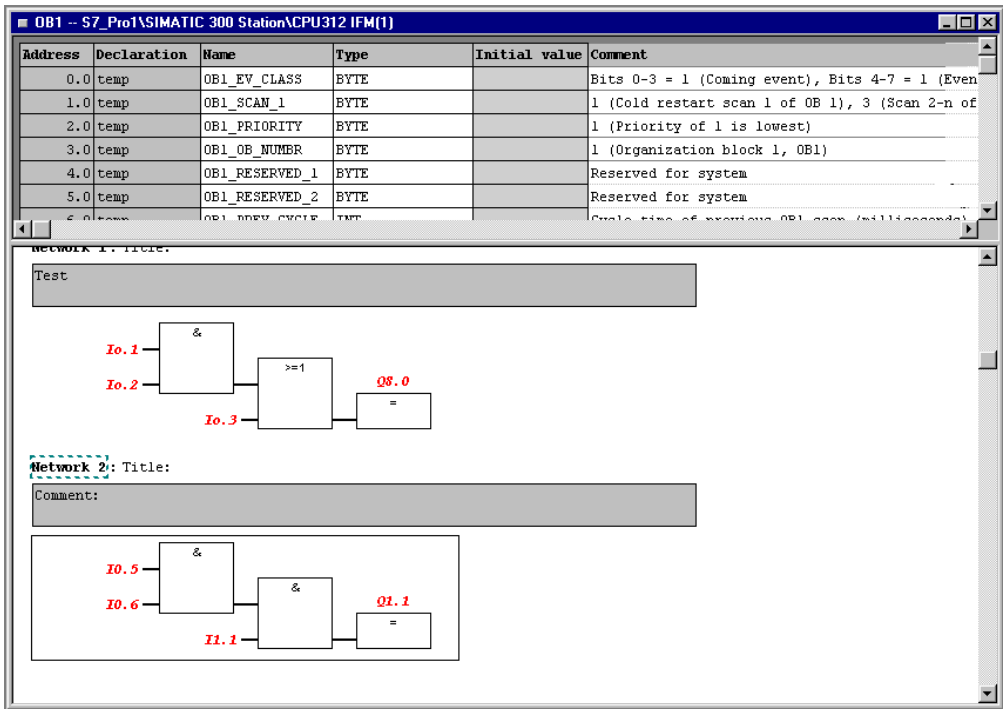

Figura 4.7

4.4 Programarea în STL

Presupune o listă de instrucțiuni (Statement List). Instrucțiunile la nivel de bit sunt cele cunoscute: A - Şi, O – SAU. În momentul în care se foloseşte un N după una din operațiile enumerate înseamnă ca variabila respectivă este negată. În cazul funcțiilor

numerice se lucrează cu acumulatorul. Atribuirea se face folosind "=". Structurile de control sunt următoarele:

- IF: IF condiție THEN instrucție; [ELSEIF condiție THEN instrucție;] [ELSE instrucție;] END_IF;
- CASE: CASE selecție OF listă de constante : instrucții; [ELSE instrucții;] END_CASE;
- FOR: FOR variabilă:=valoare_start TO valoare_final [BY increment] DO instrucții; END_FOR;
- WHILE: WHILE condiție executată DO instrucții; END_WHILE;
- REPEAT: REPEAT instrucții; UNTIL condiții_terminare; END_REPEAT;
- CONTINUE, EXIT, GOTO si RETURN sunt cele cunoscute.

În fig. 4.6 am prezentat un exemplu de program scris în STL. Funcțiile care pot fi folosite au apelare cunoscută şi sunt aritmetice sau cele oferite de STEP7 caracteristice familiei.

Alte posibilități de programare in STEP7 sunt reprezentate de programarea în CFC sau cu ajutorul diagramelor de stare, pachete care pot fi achiziționate separat.