# Програмируеми логически контролери

## I. Теоретична част

### 1. Общи положения

Програмируемите логически контролери (PLC) са компактни и гъвкави (универсални) устройства, които се използуват за управление на машини (обекти) и/или технологични процеси.

Официално приетата дефиниция за PLC е следната:

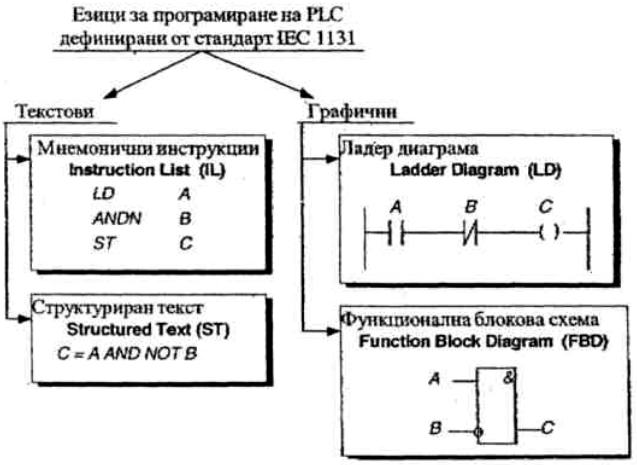
Програмируеми (промишлени) контролери се наричат **програмно управлявани електронни системи**, проектирани за използуване в качеството на промишлено оборудване за логическо управление на различни машини, съоръжения и технологични процеси, през цифрови или аналогови входове и изходи.

PLC са програмно управлявани устройства. Програмата извършва управлението на контролера, в смисъл, че при активирането на входен сигнал от логическо (on/off) или аналогово входно устройство, се изработва съответстващ отговор - активиране на изход от съответствуващо изходно устройство (логическо или аналогово) в зависимост от съществуващите логически (програмни) връзки между отделните устройства.

### 2 Езици за програмиране на PLC

Стандартът IEC 1131 дефинира синтаксиса, семантиката и различните приложения на езиците за програмиране в системите за индустриална автоматизация (включително и езиците за програмиране на PLC), независимо от базовата структура на хардуера и вградената операционна система използвана от производителя.

С това се създава възможност за използване на веднъж научения език с широк спектър от PLC системи, базирани на стандарта.




Използваната класификация на езиците за програмиране е по-обобщена от показаната, а дефинираните езици за програмиране са само четири, с което на различните производители са оставени достатъчно възможности за бъдещо развитие.

Дефинираните езици за програмиране на PLC са базирани на програмните езици използвани от най-големите производители на системи за индустриална автоматизация, с което се намалява значително времето за усвояване на PLC система.

В съвременният си вариант езиците за програмиране са взаимно свързани, т.е. дават възможности за автоматично транслиране от един вид в друг.

## 2.1. Ладер диаграми (Ladder Diagrams)

Ладер диаграмите са графичен език за програмиране на PLC, базиран на електрическите релейно-контактни схеми. Началото на "Ладер" програмирането е поставено в САЩ от първите производители на PLC.

"Ладер програмата" се състои от определена последователност на логически ("контактни") инструкции, чрез които се идентифицира състоянието на всеки един от елементите (контактите) на електромеханичната (релейно-контактна) система използвани за управлението на машината (и/или процеса). Действителното състояние на контактите от електромеханичната система се замества с логическо. В качеството на отделен елемент ("контакт") от системата за управление могат да се използват всички битове от адресируемите даннови области на конкретния PLC, в съответствие с използваната система за адресация.

Анализът на състоянията на отделните битове ("контакти") се извършва посредством две основни инструкции: "изпълни при отворен" ***(examine if open*** - ]/[), което означава че контактът е нормално затворен и "изпълни при затворен" ***(examine if close*** - ] [ ), което означава, че контактът е нормално отворен. Когато при изпълнение на съответната "контактна" инструкция контролерът анализира състоянието на конкретния бит (контакт), той изработва съответно логическо състояние.

На фиг. 1 е показано съответствието между състоянията на контакти в релейно контактна система за управление, използуваните в PLC " контактни" инструкции и изработените логически състояния.

| | Състояние на контакта | Тип на използваната инструкция | Графично обозначение | Логическо състояние |
|---|---|---|---|---|
| Нормално отворен контакт | изкл неактивен | **examine if open** изпълни при отворено | И | **1** |
| | | **examine if close** изпълни при затворено | ] [ | **0** |
| | вкл. активен | **examine if open** изпълни при отворено | И | **0** |
| | | **examine if close** изпълни при затворено | ] [ | **1** |
| Нормално затворен контакт | вкл неактивен | **examine if open** изпълни при отворено | И | **0** |
| | | **examine if close** изпълни при затворено | ] [ | **1** |
| | изкл. активен | **examine if open** изпълни при отворено | И | **1** |
| | | **examine if close** изпълни при затворено | ] [ | **0** |

Фиг. 1.

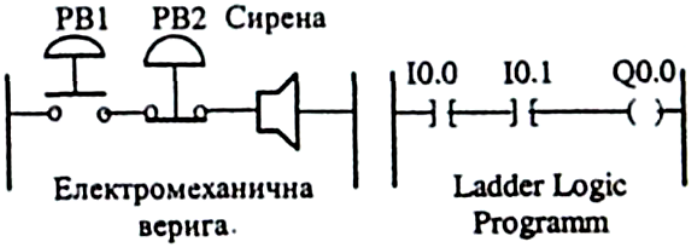
В релейно-контактните системи, управлението на всеки изпълнителен механизъм се извършва посредством отделна електрическа верига завършваща с "бобина". Контактите на всяко от устройствата участвуващи в управлението на "бобината" се свързват (паралелно и/или последователно) в съответствие с логическата функция на управлението. Когато състоянията на отделните контакти позволят протичането на ток през бобината, изпълнителният механизъм се активира.

По аналогия с релейно-контактните схеми, всяко от устройствата участващи в управлението на конкретния изпълнителен механизъм (или междинно реле) се представя в комбинация от последователни и/или паралелни логически връзки оформени, като отделно логическо уравнение -

"стъпало" в ладер диаграмата, съответстващо на веригата за управление.

Началото на всяко "стъпало"- логическо уравнение в ладер програмата, започва с условна "линия на захранването" (power line), както в релейно-контактните аналози. Прието е логическата комбинация между състоянията на отделните "контакти" в логическото уравнение (стъпалото) да се нарича "тестова зона", а участвуващите инструкции - "входни инструкции". Последният елемент на всяко "стъпало" е условна "бобина" (изходна инструкция), която се активира в резултат на текущото изчисление на логическата комбинация от състоянията на входовете (контактите) в "стъпалото". Изходът се активира, ако резултатът от сканирането на входните инструкции (изчисляването на логическата комбинация) е "истина" (логическа единица) т.е., бит **RLO (Result of logic operation) = 1** след изпълнение на последната входна инструкция

Чрез следващия пример се илюстрира съответствието между електромеханичната верига за управление на изпълнителен механизъм и стъпало в ладер диаграмата:



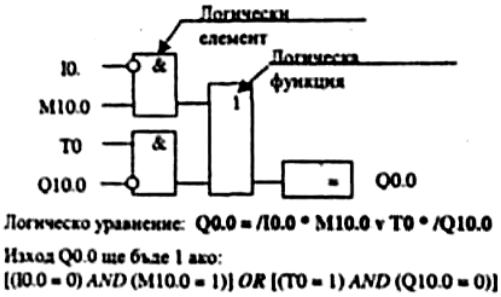

Показаната електромеханична верига включва два последователно свързани бутона PB1 и PB2. управляващи сирена Бутонът PB1 е нормално отворен, а бутонът PB2 е нормално затворен. Същата електрическа верига е показана и като ладер програма състояща се от два последователно свързани контакта към входовете **IO.0** и **IO.1,** които управляват изхода **Q0.0** За анализиране състоянието и на двата контакта са използувани инструкциите - "изпълни при затворено" (examine if closed).

### 2.2. Език на Функционалните блокови схеми (Function Block Diagram- FDB)

Езикът на функционалните блокови схеми е графичен език за програмиране на PLC, използуващ логически елементи **(Logie boxes)** от Булевата алгебра за представяне на управляващите функции.

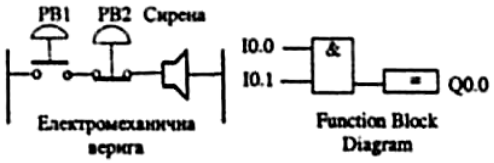
Езикът се базира на възможността за описване работата на конкретен обект за управление, а от там и функцията на управляващото устройство, посредством система от логически уравнения. Всяко логическо уравнение се кодира самостоятелно и се оформя като отделен логически стринг **(Logical Network/Logical Segment)**. Уравнението се кодира посредством последователност от логически инструкции, всяка от които проверява (тества) състоянието на адресираната променлива за 1 (True) или 0 (False) и образува съответен резултат. Полученият резултат се запомня директно или се използва за извършване на указана двоична логическа операция (Boolean logic operation) и се запомня след извършването й. Резултатът от извършената операция се означава с **RLO (Result of Logic Operation)** и се съхранява в едноименния бит на CPU. Резултатът от изпълнението на логическите инструкции за всички възможни комбинации на адресираните променливи се определя съгласно съответните таблици на истинност, познати от Булевата алгебра За графичното представяне на логическите връзки между отделните променливи се използват познатите също от булевата алгебра логически елементи **(Logic boxes) - AND; OR; XOR** и техните комбинации.

Примерна FDD програма, кодираща елементарно логическо уравнение е следната:




Графичното представяне на уравнението включва два логически елемента, изпълняващи функция AND, един - OR и един изходен елемент от тип "присвояване" (Assignment). Отделният логически елемент се кодира чрез последователност от инструкции, които се изпълняват по реда на тяхното написване, т.е. логическият елемент се сканира отгоре надолу. Резултатът от извършените логически операции над състоянията на всички променливи, участвуващи в логическия елемент, се запомня в стека за двоични логически операции и се използва за извършване на указаната в дясно поставения логически елемент двоична логическа операция т.е., участвуващите в логическото уравнение логически елементи се сканират отляво надясно.

Отделната електрическа верига на произволна релейно-контактна схема може да бъде преобразувана в логическа функция и кодирана като FDB програма Например, електромеханичната верига разгледана при ладер програмирането се представя като FDB програма по следния начин:




3. Програмируем логически контролер **SIEMENS - SIMATIC** S7 200

Програмируемият логически контролер **SIMATIC S7 200** е типичен представител на съвременните конвенционални **PLC**, разработен от американски производител. Контролерът е от клас "micro PLC" и може да се използува за решаването на различен тип задачи за управление. В конструктивно отношение контролерът е от тип "компактно устройство", а различните му модификации се различават по брой входове и изходи, обем на паметта за потребителска програма, обем на достъпната област за данни, наличието на аналогови входове и т.н.

Контролерът, който се използува в лабораторното упражнение е тип S7 200 - CPU 224. Този контролер се захранва с 24 V DC, има 14 цифрови входа и 10 цифрови изхода. За цифровите входове логическата 0 е от 0V до 5V DC, а логическата 1 е от 15V до 30V DC. Цифровите изходи са с нива OV DC и 24V DC. В данновата област състоянията на цифровите входове се записват като **Ix.y,** където **x** определя младшия или старшия байт, а **y** определя съответния бит от този байт. Аналогично се означават състоянията на цифровите изходи с **Qx.y.**

**3.1.** Програмираме на **SIMATIC** S7 200

Програмирането на контролера SIMATIC S7 200 се извършва с помощта на специализирана програма STEP 7 MicroWin 32, работеща върху персонален компютър под управление на WINDOWS. Използваният език за програмиране STEP 7 Micro е език от ниско ниво, включващ две взаимозаменяеми версии: език на мнемонични инструкции ( **ST**atement

**List – STL)** и език на релейно-контактните схеми, допълнен с редица функционални символи (Ladder Diagram).

## Контактни инструкции

**Контактните инструкции** описват състоянието на отделни "контакти" – битове от съответната потребителска област и могат да се кодират директно в "ладер диаграмата", чрез използване на правилата за преобразуване за логическа в релейно-контактна схема:

- Нормално отвореният контакт (Normally Open contact/ ***examine if close*** – NO) се затваря (включва), когато стойността на адресирания бит е 1. При STL програмиране нормално отвореният контакт се представя с инструкциите: зареждане – **Load,** логическо "И"- **AND,** логическо "ИЛИ" – **OR.**
- Нормално затвореният контакт (Normally Closed contact/ ***examine if open*** – NC) е затворен (включен), когато стойността на адресирания бит е 0. При STL програмиране нормално отвореният контакт се представя с инструкциите: зареждане на инверсната стойност – **Load Not,** логическо "И" на инверсната стойност - **AND NOT,** логическо "ИЛИ" на инверсната стойност - **OR NOT.**
- Бобина (Coil) – в качеството на бобина се използват инструкции за управление на битов операнд (bit control instuctions), допълнителни логически функции (таймерни, броячни и математически операции) и инструкции за управление на програмата.

Ще разгледаме шест базови инструкции, необходими при програмирането на релейно-контактни схеми.

## Инструкции за зареждане Load и Load Not

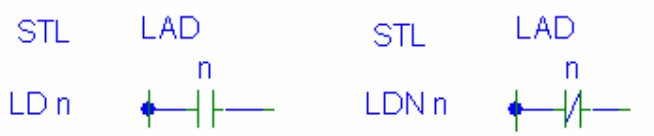
**Инструкция Load** (LD)

Инструкцията копира стойността на бита с адрес **n** на върха на логическия стек. Останалите стойности в стека слизат с едно ниво по-ниско.

**Инструкция Load Not**(LDN)

Инструкцията копира инверсната стойност (т.е. извършва операция "логическо НЕ" (logical Not)) на бита с адрес **n** на върха на логическия стек. Останалите стойности в стека слизат с едно ниво по-ниско.

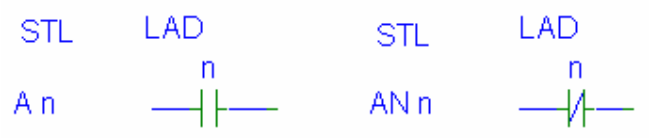
Последователност от контактни инструкции (логически стринг; блок от логически операции), винаги се започва с инструкциите Load и Load Not.




## Инструкции за операция логическо "И"- AND и AND NOT

Инструкцията **And** (A) извършва операция логическо "И" **(logical And)** между **стойността на бита с адрес "n" и стойността на върха на логическия стек.** Резултатът става нова стойност на върха на логическия стек.

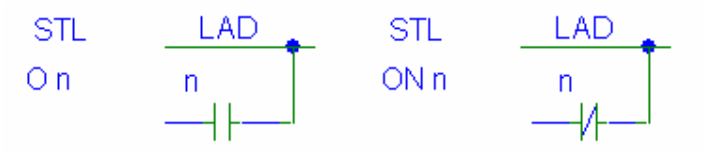
Инструкцията **And Not** (AN) извършва операция логическо "И" **(logical And)** между **инверсната стойност на бита с адрес "n" и стойността на върха на логическия стек.** Резултатът става нова стойност на върха на логическия стек.



## Инструкции за операция логическо “ИЛИ” - OR и OR NOT

Инструкцията OR (O) извършва операция логическо “ИЛИ” **(logical Or)** между **стойността на бита с адрес n** и **стойността на върха на логическия стек.** Резултатът става нова стойност на върха на логическия стек.

Инструкцията OR NOT (ON) извършва операция логическо “ИЛИ” **(logical Or)** между **инверсната стойност на бита с адрес n** и **стойността на върха на логическия стек.** Резултатът става нова стойност на върха на логическия стек.




**Механизмът на изпълнение логическите операции и представянето им във LAD и STL форма, поддържани от програмируемия контролер SIMANTIC S7 200 в следващите три примера:**

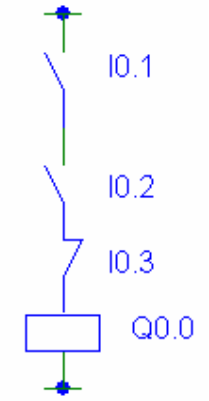
**Пример 1. Операция “логическо И”**

Зададена е логическата функция: **Q0.0 = I0.1 ^ I0.2 ^ /I0.3**

Графичното изображение на операцията “логическо И” със средствата на релейно-контактните схеми представлява съвкупност от последователно свързани контакти (фиг. 2а). Съответствието между релейно-контактните схеми и релейно-контактните езици за програмиране (LAD) позволява кодирането чрез графични инструкции да се извърши на същата база (последователно свързани контакти), както при релейно-контактните схеми -(фиг. 2б).

Кодирането на зададената функция чрез езика на мнемоничните инструкции започва с инструкция **Load** (LD) с цел формиране на начално уловие (установяване на RLO бита) и последователно използване на инструкциите извършващи операции логическо ”И” - **AND** (A), **AND Not** (AN) до изчерпване аргументите на зададената функция. Последната инструкция (в случая **Out**) присвоява изчисления резултат на адресирания бит.

В случая през бобината ще протича електрически ток, само ако нормално отворените контакти I0.1 и I0.2 са активирани (включени), а нормално затвореният I0.3 – не активиран (включен), т.е. ако: (I0.1 = 1) **и** (I0.2 = 1) **и** (I0.3 = 0).



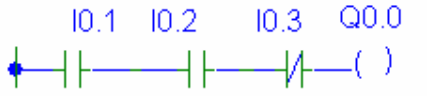

**STL**

**NETWORK 1**

LD I0.1
A I0.2
AN I0.3
= Q0.0

**LAD**

**NETWORK 1**




а) релейно-контактна схема б) представяне на програмата

Фиг. 2

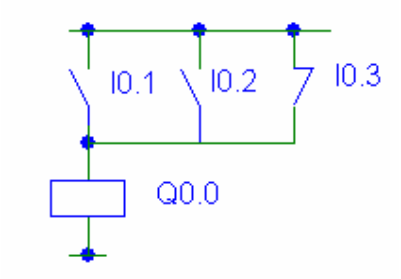
**Пример 2. Операция “логическо ИЛИ”**

Зададената логическа функция е: **Q0.0 = I0.1 v I0.2 v /I0.3**

Представянето на операцията “логическо ИЛИ” във вид на релейно-контактна схема представлява съвкупност от паралелно свързани контакти.

Кодирането на зададената функция чрез езика на мнемоничните инструкции започва с инстукция **Load** (LD) с цел формиране на начално условие (установяване на RLO бита) и последователно използване на инструкциите, извършващи операции логическо “ИЛИ”- **OR** (O), **OR Not** (ON) до изчерпване аргументите на зададената функция. Последната инструкция в случая “=” присвоява изчисления резултат на адресирания бит.

На фиг.3 са дадени релейно-контактната схема и съответното STL и LAD представяне на програмата.



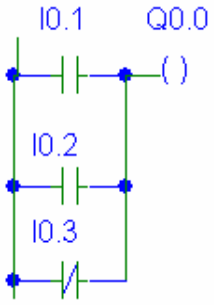

**STL**

**NETWORK 1**

| | |
|---|---|
| LD | I0.1 |
| O | I0.2 |
| ON | I0.3 |
| = | Q0.0 |

**LAD**

**NETWORK 1**



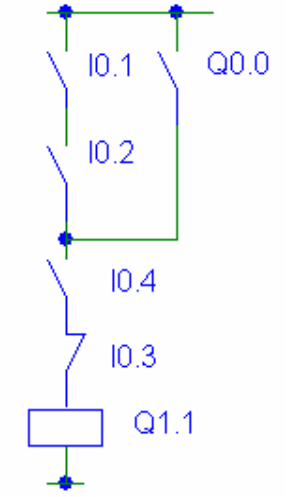

фиг. 3

В случая през бобината ще протича електрически ток, ако някой от нормално отворените контакти I0.1 или I0.2 е активиран (включен), или нормално затвореният I0.3 – не активиран (включен), т.е. ако (I0.1 = 1) **или** (I0.2 = 1) **или** (I0.3 = 0).

**Пример 3. Комбиниране на операциите “логическо ИЛИ” и “логическо И”**

Логическото уравнение е: **Q1.1 = (I0.1^I0.2 v Q0.0) ^ I0.4^ /I0.3**

В случая операцията “И”, която се извършва между логическите състояния на входовете I0.1 и I0.2, определя “**изпълнимото условие”** (състояние на RLO бита) за изпълнението на операциите “ИЛИ” със статуса на изход Q0.0. Резултатът от тази операция определя “**изпълнимото условие”** за изпълнението на следващата операция ( “И” със състоянието на вход I0.4), което пък от своя страна определя условието за извършване на следващата операция (“И” с инверсната стойност на състоянието на вход I0.3). Изчислената стойност на RLO бита се присвоява на изход Q1.1.



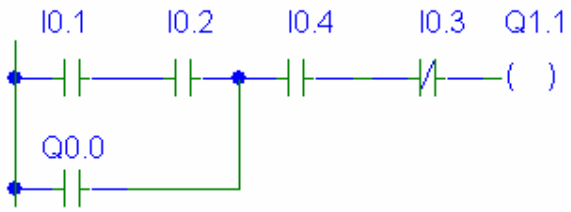

**SLT**

**NETWORK 1**

| | |
|---|---|
| LD | I0.1 |
| A | I0.2 |
| O | Q0.0 |
| A | I0.4 |
| AN | I0.3 |
| = | Q1.1 |

**LAD**

**NETWORK 1**




фиг. 4.

## 3.2. Указания за работа с програмата STEP 7-Micro/WIN 32

От **View** се избира начина на програмиране : STL, Ladder или FDB.

Програмирането се извършва в режим Ladder и се използуват само инструкциите, посочени по-горе. Те са приложени в **Instructions/Bit Logic -** ─┤ ├─, ─┤/├─ и **( )** .Копират се в съответния Network с двойно щракване на мишката.

За създаване на паралелни връзки се използуват : ↓, ↑, ←, → .

С десния бутон на мишката може да се изтриват инструкции, да се вмъкват или изтриват редове и колони в даден Network.

За компилиране на програмата се използува : √ **(Compile All**).

За зареждане на компилираната програма в контролера се използува : ▼(**Download**).

За стартиране на заредената програма се използува : ► (**RUN**).

От File се извършва Save и Open на даден проект.

**Системните битове SM0.0 и SM0.5 се дефинират в ладер диаграмите като нормално отворени контакти.**

**При стартиране на програма, която съдържа повече от един Network трябва да се има предвид, че се изпълняват всички въведени от 1 до N Networks, след което отново се изпълняват от 1 до N Networks и т.н.**

## II. Задание

1. Релейно-контактната схема от фиг. 2 да се представи като Ladder диаграма, да се компилира и зареди в контролера. С помощта на бутоните, свързани към входовете на контролера се проверява работоспособността на релейно-контактната схема. Да се наблюдават STL и FDB програмите.

2. Релейно-контактната схема от фиг. 3 да се представи като Ladder диаграма, да се компилира и зареди в контролера. С помощта на бутоните, свързани към входовете на контролера се проверява работоспособността на релейно-контактната схема. Да се наблюдават STL и FDB програмите.

3. Релейно-контактната схема от фиг. 4 да се представи като Ladder диаграма, да се компилира и зареди в контролера. С помощта на бутоните, свързани към входовете на контролера се проверява работоспособността на релейно-контактната схема. Да се наблюдават STL и FDB програмите.

4. Да се активира изход **Q0.0** в релейно-контактната схема дадена на фиг.4. За тази цел се добавя Network 2, в който се задава **Q0.0 = SM0.0** (бит **SM0.0** е винаги логическа 1). Програмата, съдържаща Network 1 от т.3 и Network 2 се компилира и зарежда в контролера. Да се провери, че бутоните свързани към входовете **I0.1** и **I0.2** не влияят на работата на релейно-контактната схема от фиг. 4 при **Q0.0 = 1**.

5. За логическото уравнение **Q0.0 = (I0.1^I0.2^/I0.3 v I0.4)^/I0.5** да се начертаят съответните релейно-контактна схема и Ladder диаграма. Компилираната програма да се зареди в контролера и да се провери работоспособността й с помощта на бутоните. Да се наблюдават STL и FDB програмите.

6. Като се знае, че системен бит **SM0.5** реализира импулсна поредица с период 1 sec., то:

6.1. В Network 1 да се напише Ladder програма за включване/изключване на изход **Q0.4** на всяка секунда при натиснат бутон 2, т.е при **I0.2 = 1**.

6.2. В Network 2 да се напише аналогична програма за изход **Q1.1** при ненатиснат бутон 3, т.е. при **I0.3 = 0**.

6.3. Като се използува бутон 1, в Network 3 да се напише Ladder диаграмата за логическото уравнение : **Q0.0 = I0.1^Q0.4^Q1.1** .

6.4. Компилираната програма да се зареди в контролера и да се провери работоспособността й с помощта на бутоните.

6.5. Да се наблюдават STL и FDB програмите.

*За повече информация на адрес:* www.automation.simens.com/ en/s7-200/