Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
ФГОУ СПО «Кузнецкий индустриальный техникум»

ОСНОВЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ STEP 7 И БАЗОВОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПРОМЫШЛЕННЫХ КОНТРОЛЛЕРОВ SIEMENS.

Учебно-методическое пособие

Для студентов специальностей 230106

Новокузнецк
2009

Рассмотрено на заседании цикловой предметной комиссией ЭВТ и программирования

Протокол № _____
От «____»_____2009 года
Председатель ПЦК:

_____Агаркова Н. А.

«___» _____ 2009г.

«УТВЕРЖДАЮ»

Зам. Директора по УМР

_____Терехина И. А.

«___» _____ 2009г.

СОСТАВИТЕЛЬ:
Романов В. П., преподаватель высшей квалификационной категории ФГОУ СПО «КИТ».

РЕЦЕНЗЕНТ:
Проректор «СИБГИУ» ИР и ОВ, зав кафедрой «автоматизации и информационных систем» доктор технических наук, профессор Кулаков С. М.

Методические указания включают описание пакета STEP 7 и основных приемов работы при создании, конфигурировании, написании программ и отладки проекта. Приводится пример создания простого проекта, контрольные задания, контрольные вопросы и список рекомендуемой литературы для углубленного изучения темы.

Табл. 6. Ил. 45. Библиогр.: 2назв.

Рекомендовано к изданию методическим советом колледжа
ПРОТОКОЛ №_____

«___»_____200__г.

Оглавление

ВВЕДЕНИЕ 4
1. ТИПОВАЯ АРХИТЕКТУРА ПРОЦЕССОРА S7 5
1.1. Регистры CPU 5
2. ОСНОВНЫЕ ПОНЯТИЯ ЯЗЫКА STEP 7 8
2.1. Структура программы 8
2.2. Инструкции языка STEP 8
2.3. Типы блоков 9
2.4. Типы данных 11
2.5. Виды адресации 13
2.6. Обращение к данным в областях памяти 14
3. ОБЗОР ОСНОВНЫХ ИНСТРУКЦИЙ ЯЗЫКА STEP 7 16
3.1 Обзор битовых логических инструкций 16
3.2 Обзор инструкций сравнения 17
3.3 Обзор операций со счетчиками 18
3.4 Обзор инструкций перехода 18
3.5 Обзор инструкций загрузки и передачи 19
3.6 Обзор математических инструкций с целыми числами 19
3.7 Обзор математических инструкций над числами с плавающей точкой 20
3.8 Обзор инструкций сдвига 21
3.9 Обзор инструкций с таймерами 21
3.10 Обзор инструкций с аккумуляторами и адресными регистрами 22
3.11 Примеры использования инструкций языка STEP 7 для составления программ. 23
4. ПРИМЕР СОЗДАНИЯ ПРОЕКТА 41
5. ПРАКТИЧЕСКИЕ ЗАДАНИЯ 43
6. КОНТРОЛЬНЫЕ ВОПРОСЫ 45
7. ЛИТЕРАТУРА 45

ВВЕДЕНИЕ

АСУТП - это автоматизированная (человеко-машинная) система для выработки и реализации управляющих воздействий на технологический объект управления в соответствии с принятым критерием управления.

При этом под технологическим объектом управления понимается совокупность технологического оборудования и реализованного на нём в соответствии с инструкциями и регламентами технологического процесса производства, рассматриваемые как объекты управления.

Процесс автоматизации производства зародился вместе с самим производством и в процессе своего развития прошел целый ряд этапов: от управления при помощи простейших технических устройств, до современных АСУ, построенных на базе вычислительной техники (ВТ).

Комплексное использование ВТ при автоматизации производства позволяет создавать гибкие автоматизированные производства (ГАП). Создание ГАП - генеральное направление развития и автоматизации производственных процессов. В настоящее время широкое применение при создании АСУТП широко применяются программируемые логические контроллеры фирмы SIMENS, при создании управляющих программ для которых применяется язык программирование STEP7.

Базовый пакет STEP 7 предназначен для создания проектов, решающих задачи автоматизации отдельных станков, участков, технологических процессов. Рассматриваемый пакет позволяет проводить разработку как программных, так и аппаратных средств в пределах одного проекта, в результате чего на основе требований к программной и аппаратной частям происходит создание и конфигурирование необходимых средств и сетей, рабочих программ и блоков данных для решения задач автоматизации.

Для создания программного обеспечения требуется разработать:

а) структуру программы;

б) управление данными автоматического процесса;

в) структуру данных;

г) передачу данных;

д) документацию программы и проекта.

Общая структура системы, работающей под управлением STEP 7, показана на рисунке 1.

Рисунок 1 - Структура системы автоматизации

Целью настоящей работы знакомство студентов внутренней структурой ПЛК S7, основными инструкциями языка программирования ПЛК STEP 7, освоение основных утилит программы STEP 7 и приемов работы, позволяющих научиться создавать проект, конфигурировать его аппаратную часть, создавать программное обеспечение и проводить тестирование проекта.

1. ТИПОВАЯ АРХИТЕКТУРА ПРОЦЕССОРА S7

1.1. Регистры CPU

Регистры CPU Регистры CPU используются для адресации или обработки данных. Данные могут с помощью соответствующих команд быть обменены между областями памяти CPU и регистрами. CPU содержит следующие программно доступные регистры:

- **Аккумуляторы:** Два (в S7-300) или четыре (в S7-400) аккумулятора используются для арифметики, сравнений с байтами, словами или двойными словами.
- **Адресные регистры:** Два адресных регистра используются как указатели для косвенной адресации памяти.
- **Регистры блоков данных:** Регистры блоков данных содержат номера открытых блоков данных. Таким образом возможно, что открыты одновременно два DB: один DB с помощью регистра DB, другой как экземпляр DB с помощью регистра DI. Когда DB открыт, его длина (в байтах) автоматически загружается в связанный с ним регистр.
- **Слово статуса:** Содержит различные биты, которые отражают результат или статус отдельных инструкций во время выполнения программы.

Области памяти Память S7-CPU может быть разделены на четыре области:

- **Загрузочная память** используется, чтобы хранить программу пользователя без символов и комментариев. Загрузочная память может быть выполнена в виде RAM или FLASH EPROM.
- **Рабочая память** (встроенная RAM) используется, чтобы хранить соответствующую часть S7- программы, необходимую для выполнения программы. Программа выполняется исключительно в рабочей памяти.
- **Область ввода** - вывода разрешает прямой доступ ко входам и выходам, связанных с ней сигнальных модулей.
- **Системная память** (RAM) содержит области отображения входного и выходного процессов, меркеры, таймеры и счетчики. Кроме того, она содержит локальный стек, стек блоков и стек прерываний.

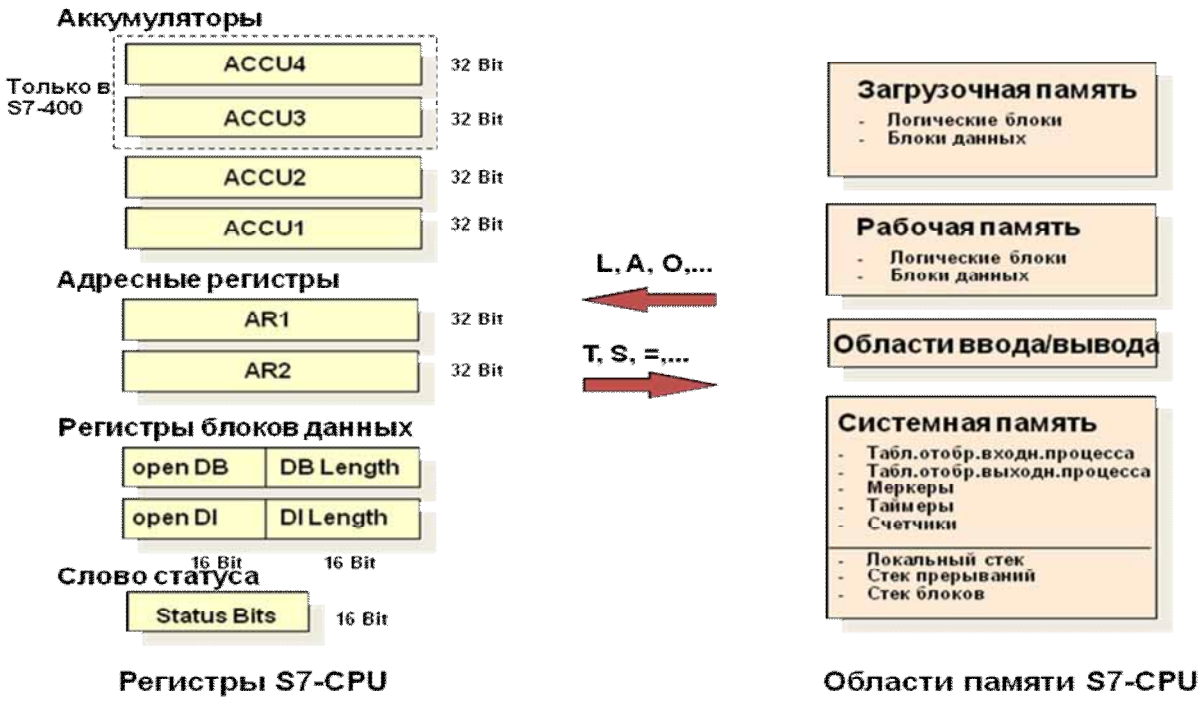


Рисунок 2 - Регистры и области памяти S7-CPU

Для создания программ управления необходимо знать назначение отдельных бит некоторых регистров (смотри рисунок 3).

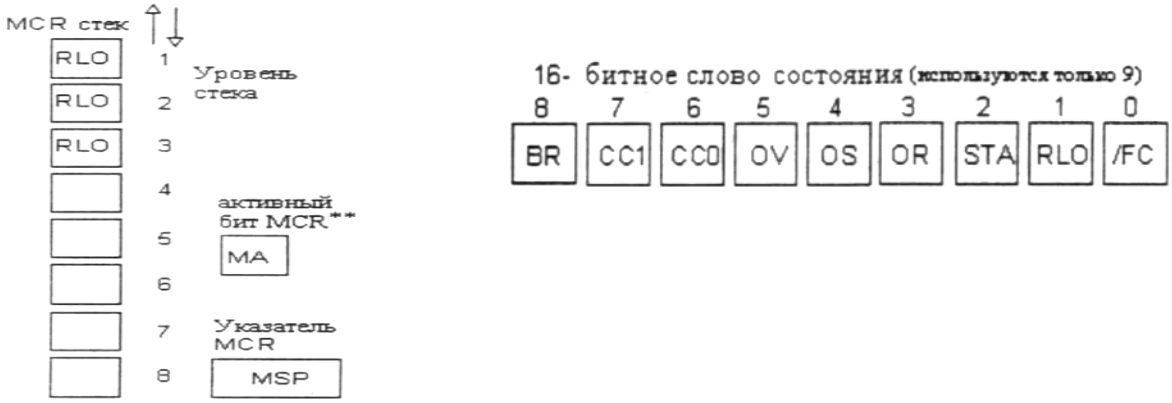


Рисунок 3 - Схематичное представление некоторых регистров процессора

<u>Назначение отдельных элементов:</u>

MCR (*Master Control Rela).* - *Блокировка выходных сигналов*

Master Control Relay - это "самый главный" выключатель, управляющий подачей энергии на элементы технологического процесса и, в случае аварии, позволяющий их обесточить. В STEP7 есть возможность в зависимости от состояния MCR-бита просто запретить прохождение ненулевых сигналов на выходы контроллера, т.е. заблокировать выходы.

СЛОВО СОСТОЯНИЯ (РЕГИСТР ФЛАГОВ)

Выполнение команды может изменить соответствующие биты слова состояния CPU. Эти биты могут использоваться как результат операции, а также как информация об результате выполнения той или иной команды.

/FC - First Check Bit - ***Флаг первичного опроса***

Состояние флага первичного опроса управляет цепочкой битовых логических команд. Если опрос первичный, то /FC = "О". Цепочка битовых логических команд заканчивается и /FC сбрасывается в "О" после команд присваивания (=, R, S), а также после команд переходов, анализирующих флаги RLO или BR. Любая другая битовая логическая команда устанавливает флаг первичного опроса. Каждая битовая логическая команда анализирует состояние флага /FC:

- Если /FC = "1", команда логически объединяет результат опроса бита с результатом логической операции, сформированным со времени первичного опроса и запомненным в бите RJLO.
- Если /FC = "О", строка логических команд начинается с первичного опроса, т.е. результаты предыдущих опросов не учитываются в текущей команде.

RLO - Result of Logic Operation - ***Флаг логического результата***

Во время выполнения программы процессор при обработке отдельных опросов образует сначала *результат опроса.* Если выполнено условие опроса, то результат опроса равен 1, если условие опроса не выполнено, результат опроса равен 0. Результат опроса накапливается процессором как результат логической операции (RLO) следующим образом:

- Результат опроса бита логически объединяется с предыдущим результатом логической операции, сохраненным в бите RLO. если опрос не первичный, т.е. /FC="1".
- Если опрос первичный, т.е. /FC= "О", то результат опроса бита и будет являться результатом логической операции RLO, т.е. все предыдущие опросы битов не учитываются в текущей команде.

Помимо битовых логических команд на состояние флага RLO влияют также команды равнения. Если условие сравнения выполнено. RLO="1", если условие не выполнено RLO"0".

Флаг результата логической операции можно использовать для анализа результата команд переходов, вызовов блоков, присваивания результата какому-либо биту и т.д.

STA - Status Bit - *Флаг статуса*

При выполнении команд чтения бита памяти (U, UN, O, ON, X. XN) флаг статуса всегда равен значению опрашиваемого бита.

При выполнении команд, которые могут изменить состояние бита памяти (=. R, S), флаг STA равен записываемому биту или, если запись в память не выполняется, равен значению адресуемого бита, т.е. того бита, состояние которого хотели изменить. Команды битовой логики, не выполняющие доступа к памяти, устанавливают флаг статуса в "1". Флаг статуса не анализируется командами, он отображается только для Вас, когда Вы просматриваете в отладчике состояние программных переменных.

OR - *Флаг операции ИЛИ*

Флаг OR используется для объединения команд "И" перед командами "ИЛИ". Устанавливается в "1", если RLO команды "И" =1. Это блокирует результат логической операции "ИЛИ". Любая другая команда работы с битами сбрасывает флаг OR.

OV - Overflow - *Флаг переполнения*

Флаг переполнения отображает ошибки, возникающие при выполнении математических команд или команд сравнения для чисел с плавающей точкой. Флаг OV устанавливается этими командами после того, как имела место ошибка (переполнение, недопустимая операция, сравнение невыполнимо). Флаг OV сбрасывается после исчезновения ошибки.

OS - Overflow Stored - *Флаг переполнения*

Флаг OS запоминает флаг OV, если имела место ошибка во время выполнения математических команд и команд сравнения для чисел с плавающей точкой. Флаг OS устанавливается вместе с флагом OV при возникновении ошибки, но, в отличие от флага OV. не сбрасывается при исчезновении ошибки. Таким образом, он отображает, <>или ли ошибки во время выполнения одной из предыдущих команд. Сбрасывается *в* "О" только командами вызова блоков, окончания блоков, а также командой перехода SI'S (переход, если OS="1").

BR - Binary Result Bit - *Флаг двоичного результата*

Флаг BR организует связь между обработкой бит и слов. Этот флаг позволяет Вашей программе интерпретировать результат логических операций над словами как битовый результат и использовать его в потоке битовой логики.

Например, флаг BR дает Вам возможность написать функцию (FC) или функциональный блок (FB) в STL. а затем вызывать их в LAD. Во время написания FB или FC в STL. если Вы хотите потом вызывать их в LAD, Вы должны запомнить RLO в бите BR так, чтобы должным образом обеспечить сигнал разрешения выхода (ENO) для блока LAD. Этот сигнал ENO будет затем использоваться для разрешения или запрета вызова других блоков, следующих за этим.

Когда Вы вызываете системные функции (SFC) или функционально назначенные блоки (SFB) из Вашей программы, бит BR отображает, были ли ошибки при выполнении SFC или SFB:

- Если при выполнении блока имели место ошибки. BR = "О".
- Если ошибок при выполнении блока не было. BR = " 1".

CCO, CC1 - Condition Codes - *Флаги условия*

Биты CCO и CC1 могут использоваться для определения результата выполнения достаточно большого числа команд.

2. ОСНОВНЫЕ ПОНЯТИЯ ЯЗЫКА STEP 7

2.1. Структура программы

В полную программу процесса входят

Операционная система: содержит общую часть всех инструкций и соглашений для реализации внутренних функций (например, сохранение данных при сбросе напряжения питания, управление реакцией пользователя при прерывании и т. д.). Она расположена на так называемом EPROMe (Erasable Programmable Read Only Memory) и является фиксированной составной частью процессора. Как пользователь, вы не имеете возможности обращаться к операционной системе.

Программа пользователя: содержит набор всех написанных пользователем инструкций и соглашений для обработки сигналов, с помощью которых производится управление установкой (процессом). Программа пользователя распределяется на блоки. Деление программы на блоки значительно проясняет структуру программы и подчеркивает программно-технические связи отдельных частей установки.

Блоком называется часть программы пользователя, ограниченная функционально и структурно или по целям использования.

Различают блоки, которые содержат

- инструкции для обработки сигналов;
- блоки, содержащие данные (блоки данных).

Блоки идентифицируются:

- типом блока (OB, PB, SB, FB, FX, DB, DX);
- номером блока (число от 0 до 255).

2.2. Инструкции языка STEP.

Инструкция языка STEP является наименьшей самостоятельной единицей программы пользователя. Она является предписанием для работы процессора. Инструкция может быть представлена в виде

- ***Список команд (инструкций) - STL***. Представляет собой список команд подобно обычному языку Ассемблера.
- ***Контактный план - LAD***. Управляющая программа записывается при помощи изображений элементов релейных контактных схем.
- ***Функциональный план – FUP***. Для отображения программы используются схемы логических элементов.

STL-программа состоит из ряда отдельных выражений (statement). Выражение - это наименьшая самостоятельная единица пользовательской программы. Выражение содержит описание работы для CPU. На рисунке 3.6 показана общая структура STL-выражения.

Инструкция языка STEP состоит из операции и операнда.

Операнд может быть представлен абсолютно или символически (через список соответствия).

а) в)

Рисунок 4 - Инструкции языка STEP 7
а)структура STL-выражения; в) виды представления инструкций языка STEP 7

2.3. Типы блоков

Таблица 1 содержит перечень и краткое описание программных блоков для контроллеров SIMATIC S7, а также блоков данных.

В контроллерах SIMATIC S7 существует несколько способов обработки управляющей программы:

1. **Циклическая обработка**. Состоит из повторных (периодически повторяющихся) обработок управляющей программы, которая начинается с вызова организационного блока OB1. В начале цикла обработки программы ОС заполняет область отображения входов, сбрасывает таймер контроля длительности цикла, после этого вызывает для обработки блок OB1. В конце цикла обработки ОС переписывает в выходные модули значения из области отображения выходов, после чего начинается следующий цикл обработки. В блоке OB1 можно вызывать функции и функциональные блоки. После обработки вызванного блока управление передается блоку, из которого был произведен вызов данного блока.

Таблица 1-Программные блоки и блоки данных

Блок	Выполняемые функции
OB	Организационные блоки. Предназначены для: организации циклического выполнения программы пользователя (OB1), обработки событий: • включения питания (OB100), • циклического прерывания (OB30-OB38), • прерывания по дате и времени (OB10-OB17), • прерывания по задержке времени (OB20-OB23), • возникновения ошибки (OB40-OB47, OB80-OB87, OB121, OB122). Блоки вызываются *автоматически* операционной системой контроллера в случае возникновения того или иного события.[1]
FC	Функция. Может быть вызвана из любого блока. Допускается передача параметров в функцию и обратно. Функция может также иметь локальные переменные, которые теряются при выходе из блока.

Блок	Выполняемые функции
FB	Функциональный блок. Также может быть вызван из любого блока и может иметь формальные и локальные параметры. Особенностью FB является наличие переменных типа STAT, которые сохраняют свое значение при выходе из блока. Поэтому функциональный блок имеет один или несколько связанных с ним блоков
SFC	Системная функция. Это функция, уже имеющаяся в ОС CPU. Предназначена для выполнения определенных стандартных действий.
SFB	Системный функциональный блок. Аналогичен FB, но, как и SFC. уже имеется в составе ОС контроллера.2
DB	Блок данных программы пользователя. Предназначен для долговременного хранения информации.
DI	Блок данных функционального блока. Используется для хранения значений переменных функционального блока. Отличается от DB наличием жесткой структуры, определяемой связанным с ним функциональным блоком.

2. **Циклическая обработка**. Состоит из повторных (периодически повторяющихся) обработок управляющей программы, которая начинается с вызова организационного блока OB1. В начале цикла обработки программы ОС заполняет область отображения входов, сбрасывает таймер контроля длительности цикла, после этого вызывает для обработки блок OB1. В конце цикла обработки ОС переписывает в выходные модули значения из области отображения выходов, после чего начинается следующий цикл обработки. В блоке OB1 можно вызывать функции и функциональные блоки. После обработки вызванного блока управление передается блоку, из которого был произведен вызов данного блока.

3. **Циклические прерывания**. При управлении ТП всегда существуют программы, которые должны обрабатываться через одинаковые, заранее заданные, промежутки времени. Для этих целей в контроллерах SIMATIC S7 существуют блоки обработки циклических прерываний. Промежуток времени, через который должен вызываться данный блок, задается программистом.

4. **Прерывания по дате и времени**. Существуют программы, которые должны выполниться один раз в определенный день и час или выполняться периодически. начиная с определенных даты и времени. Для этих целей в контроллерах SIMATIC S7 можно запрограммировать блоки прерываний по дате и времени.

5. **Прерывания по задержке времени**. Такие блоки вызываются по истечении определенного времени после возникновения какого-либо события.

6. **Обработка включения питания.** Часто при включении питания необходимо выполнить какие-либо однократные действия: первичную установку, инициализацию и т.д. Для этих целей предусмотрены блоки обработки включения питания.

7. **Обработка ошибок**. Такие блоки выполняются в случае возникновения аппаратных или программных ошибок.

Организационные блоки обрабатываются циклически. Период и приоритет обработки определяется номером организационного блока (см. таблицу 2).

Таблица 2 - Период и приоритет обработки организационных блоков

Организационный блок	Условия вызова	Приоритет	
		по умолчанию	изменения
OB1	Вызывается операционной системой	1	Нет
OB10 – OB17	В заданное время	2	2 … 24
OB20 – OB23	По истечении времени	3 … 6	2 … 24

Организационный блок	Условия вызова	Приоритет	
		по умолчанию	изменения
OB30 – OB38	Через заданный интервал	7 … 15	2 … 24
OB40 – OB47	От входов и выходов	16 … 23	2 … 24
OB60	Мультипроцессорное	25	Нет
OB70, OB72, OB73	Ошибки резервирования	25, 28, 25	2 … 28
OB80 – OB85	Асинхронные ошибки	26	2 … 26
OB90	Фоновая обработка	29	Нет
OB100	При запуске	27	Нет
OB121, OB122	Ошибки выполнения программы	Приоритет блока, вызвавшего программу	

2.4. Типы данных

Контроллеры SIMATIC S7 могут работать со следующими типами данных:

1. Элементарные типы данных (до 32 бит)

a. ***Битовые типы данных*** представлены следующими типами*:*

Бит (BOOL)

Бит - это единица, соответствующая одному двоичному разряду. Два возможных значения бита обозначаются "0" (FALSE) и "1" (TRUE).

Байт (BYTE)

Байт состоит из 8 бит, которым соответствуют битовые адреса от 0 до 7 (справа налево). Старшим является бит с большим адресом. Байт могут образовать только те биты, адрес младшего из которых кратен 8, например: 0, 8, 16 и т.д. В контроллерах Simatic S7 байт может интерпретироваться как просто байт (набор бит) или как ASCII-символ.

Слово (WORD)

Слово - это следующая после байта по величине единица, ее длина 16 бит. Любые два соседних байта можно объединить в слово, старшим будет являться байт с меньшим адресом. Адрес слова - это адрес байта с меньшим адресом. В контроллерах Simatic S7 слово может интерпретироваться как просто слово (набор бит), целое число со знаком, дата, время и т.д.

Двойное слово (DWORD)

Любые два соседних слова можно объединить в двойное слово, его длина - 32 бита или 4 байта. Старшим словом (байтом) является слово (байт) с меньшим адресом. Адрес двойного слова - это адрес байта с меньшим адресом. В контроллерах Simatic S7 двойное слово можно интерпретировать как просто двойное слово, длинное целое число со знаком, вещественное число в формате IEEE и т.д.

Таблица 3 - Представление битовых типов данных

Представление	Байт(B)	Слово (W)	Двойное слово (D)
Целое без знака	от 0 до 255 от 0 до FF	от 0 до 65 535 от 0 до FFFF	от 0 до 4 294 967 295 от 0 до FFFF FFFF
Целое со знаком	от-128до+127 от 80 до 7F	от-32 768 до+32 767 от 8000 до7FFF	от -2 147 483 648 до +2 147 483 647 от 8000 0000 до 7FFF FFFF
Вещественное IEEE 32-битовое с плавающей точкой	Неприменимо	Неприменимо	от +1.175495E-38 до +3.402823E+38 (положительное) от -1.175495E-38 до -3.402823E+38 (отрицательное)

Char (литера)

Переменная типа CHAR (character, литера) занимает один байт. Тип данных CHAR представляет одну литеру в ASCII-формате, например, 'A'.

Работая с этим типом данных, вы можете использовать любую печатную литеру в апострофах.

b. ***Математические типы данных*** представлены следующими типами*:*

INT (целое число)

Переменная типа INT (integer) хранится как целое число (16-битное число с фиксированной запятой или десятичной точкой). Тип данных INT не имеет специального идентификатора.

Целочисленная переменная занимает одно машинное слово. Сигнальные состояния битов с 0-го по 14-ый представляют цифровые разряды (позиции) числа. Сигнальное состояние 15-го бита представляет знак (sign, S).

DINT (двойное целое число)

Переменная типа DINT хранится как целое число (32-битное число с фиксированной запятой). Целое сохраняется в DINT-переменной. когда оно превышает 32 767 или меньше -32 768, или когда число предваряется идентификатором типа L#.

Под переменную типа DINT отводится двойное слово. Сигнальные состояния битов с 0-го по 30-ый представляют цифровые позиции числа. Знак хранится в 31-м бите.

REAL (вещественный)

Переменная типа REAL представляет дробь и хранится как 32-битное число с плавающей запятой (десятичной точкой). Целое сохраняется как переменная типа REAL при добавлении десятичной точки и тая.

В экспоненциальном представлении вы можете предварить «e» или «E» целым числом или дробью из семи соответствующих чисел и знака. Цифры, которые расположены за «e» или «E» представляют экспоненту по базе 10. STEP 7 производит преобразование REAL-переменной во внутренне представление числа с плавающей точкой.

c. ***Временные типы данных*** представлены следующими типами*:*

S5TIME

Переменная типа S5TIME используется в базовых языках STL, LAD и FBD для установки таймеров системы SIMATIC. Она занимает одно 16-битное слово с 1 + 3 декадами.

Время устанавливается в часах (hours), минутах (minutes), секундах (seconds) и миллисекундах (milliseconds).

DATE (Дата)

Переменная типа DATE хранится в машинном слове как число с фиксированной точкой без знака. Содержимое переменной соответствует количеству дней, начиная с 01.01.1990. Ее представление показывает год, месяц и день, разделенные дефисом.

TIME (Время)

Переменная типа TIME резервирует одно двойное слово. Ее представление содержит информацию о днях (d), часах (li), минутах (т), секундах (s) и миллисекундах (ms), отдельные элементы этих данных могут быть опущены. Содержимое переменной интерпретируется в миллисекундах (ms) и хранится как 32-битное число с фиксированной точкой со знаком.

TIME_OF_DAY (Время суток)

Переменная типа данных TTMEOFDAY резервирует для себя одно двойное слово. Она содержит количество миллисекунд с начала суток (со времени 00:00) в виде числа с фиксированной точкой без знака. Ее представление содержит информацию о часах, минутах и секундах, разделенных двоеточием. Миллисекунды, которые следуют за секундами, отделены от них десятичной точкой. Миллисекунды могут отсутствовать.

2. Сложные типы данных (более чем 32 бита)

STEP 7 определяет следующие четыре сложных типа данных:

- **DATE_AND_TIME** (DT, Дата и время) Дата и время (в формате BCD-числа);
- **STRING** (Строка) Строка литер длиной до 254 знаков;
- **ARRAY** (Массив) Переменная-массив (совокупность переменных одного типа);

- **STRUCT** (Структура) Переменная-структура (совокупность переменных разных типов).

Типы данных предопределяются пользователем при их использовании: задается длина в типе STRING (строка литер), сочетание и размер в типах ARRAY и STRUCT (структура).

3. Типы данных определенные пользователем (более чем 32 бита)

User data type - UDT (Пользовательский тип данных) соответствует структуре (комбинация компонентов любых типов) с действием на глобальном уровне. Вы можете воспользоваться пользовательским типом данных, если в вашей программе часто фигурирует структурный тип и переменные, или вы хотите структуре данных присвоить имя.

Типы UDT обладают глобальным действием; то есть, они описываются один раз и доступны для использования во всех блоках.

При объявлении переменных определяются следующие их свойства:

- символическое имя
- тип данных
- видимость переменной

Переменные могут быть объявлены:

- в глобальной символьной таблице (элементарные типы данных)
- в таблице описаний глобального блока данных (все типы данных)
- в таблице описаний логического блока (OB, FB и FC)

Рисунок 5 - Типы данных языка STEP 7

2.5. Виды адресации

При написании программ в STEP 7 можно применять прямую адресацию или косвенная адресация.

Прямая адресация может быть представлена в виде

- абсолютной адресации;
- символьной адресации.

Абсолютная адресация состоит из следующих основных полей – *идентификатор области памяти и адрес в этой области*.

Однако при большом числе переменных такая адресация неудобна, поэтому для придания смысловой нагрузки переменных вводятся их символьные обозначения, те применяется **символьная адресация.**

Для хранения символьных обозначений используется специальная таблица, содержащая четыре столбца, с названием, адресом, типом данных и комментарием.

Символьное имя Symbol содержит до 24 символов, начинается с буквы, может содержать подчеркивания.

Например, если входной дискретный модуль занимает адреса от 0 до 3, то входы могут обозначаться как I 0.0, I 0.1 и т.д. Аналогично выходы для цифрового модуля вывода, который занимает адреса с 4 по 7, обозначаются как Q 4.0, Q 4.1 и т.д.

Косвенная адресация является более сложным видом адресации и в данном пособии не рассматривается.(данный метод адресации описан в [1]). Основные виды адресации доступные с STEP 7 представлены на рисунке .

Рисунок 6 - Виды адресаций доступные в языке STEP 7

2.6. Обращение к данным в областях памяти

Контроллер S7 хранит информацию в различных местах памяти, которые имеют однозначные адреса. Программист можете явно указать адрес в памяти, к которому он хочет обратиться. Благодаря этому программа имеет прямой доступ к информации.

Для обращения к биту в некоторой области памяти программист должен указать адрес бита. Этот адрес состоит из *идентификатора области памяти, адреса байта и номера бита.*

В контроллер S7 существуют следующие области памяти

Память входов образа процесса: обозначается - I

В начале каждого цикла S7-200 опрашивает физические входы и записывает полученные значения во регистр входов образа процесса. К образу процесса можно обратиться в формате бита, байта, слова и двойного слова:

Тип данных	Обращение	Пример
Бит	**I**[адрес байта]. [адрес бита]	I0.1
Байт	**I**[длина(B)][начальный адрес байта]	IB4
Слово	**I**[длина(W)][начальный адрес байта]	IW4
Двойное слово	**I**[длина(D)][начальный адрес байта]	ID4

Память выходов образа процесса: Q

В конце цикла S7-200 копирует значения, хранящиеся в регистре выходов образа процесса, в физические выходы. К образу процесса можно обратиться в формате бита, байта, слова и двойного слова:

Тип данных	Обращение	Пример
Бит	**Q**[адрес байта]. [адрес бита]	Q0.1
Байт	**Q**[длина(B)][начальный адрес байта]	QB4
Слово	**Q**[длина(W)][начальный адрес байта]	QW4
Двойное слово	**Q**[длина(D)][начальный адрес байта]	QD4

Память памяти переменных: V

Память переменных можно использовать для хранения промежуточных результатов операций, выполняемых в вашей программе. В памяти переменных можно хранить также другие данные, имеющие отношение к процессу или к решению конкретной задачи автоматизации. К памяти переменных можно обратиться в формате бита, байта, слова и двойного слова:

Тип данных	Обращение	Пример
Бит	**V**[адрес байта]. [адрес бита]	V10.2
Байт	**V**[длина(B)][начальный адрес байта]	VB100
Слово	**V**[длина(W)][начальный адрес байта]	VW200
Двойное слово	**V**[длина(D)][начальный адрес байта]	VD200

Область битовой памяти (меркерная память): M

Биты памяти (меркеры) можно использовать как управляющие реле для хранения промежуточных результатов операций или другой управляющей информации. К битам памяти можно обратиться в формате бита, байта, слова и двойного слова:

Тип данных	Обращение	Пример
Бит	**M**[адрес байта]. [адрес бита]	M10.2
Байт	**M**[длина(B)][начальный адрес байта]	MB100
Слово	**M**[длина(W)][начальный адрес байта]	MW200
Двойное слово	**M**[длина(D)][начальный адрес байта]	MD200

К данным в других областях памяти

- **Область памяти таймеров – T;**
- **Область памяти счетчиков – C;**
- **Аккумуляторам**

обращаются, указывая в качестве адреса идентификатор области и номер элемента.

Тип данных	Обращение	Пример
Область памяти таймеров	**T**[номер элемента]	T37
Область памяти счетчиков	**C**[номер элемента]	C5
Аккумуляторам	**AC**[номер элемента]	AC0

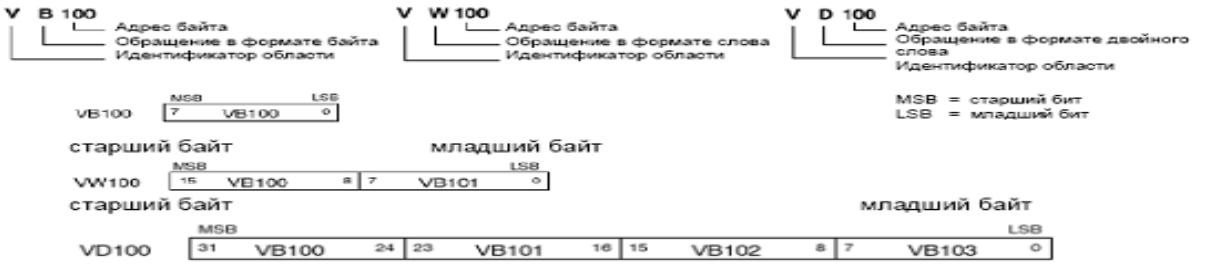


Рисунок 7 - Пример обращения к одному и тому же адресу в формате байта, слова и двойного слова

3. ОБЗОР ОСНОВНЫХ ИНСТРУКЦИЙ ЯЗЫКА STEP 7

3.1 Обзор битовых логических инструкций

Битовые логические инструкции работают с двумя числами, 1 и 0. Эти две цифры образуют базис системы счисления, называемой двоичной системой. Цифры 1 и 0 называются двоичными цифрами (**bi**nary digi**t**s) или просто битами. При работе со схемами, использующими контакты и катушки, значение 1 означает активное состояние или протекание тока, а 0– неактивное состояние или отсутствие протекания тока.

Битовые логические инструкции интерпретируют состояния сигналов 1 и 0 и комбинируют их по правилам булевой логики. Эти комбинации дают результат 1 или 0, называемый «результатом логической операции» (RLO).

Для приложений битовой логики используются следующие битовые логические инструкции:

- **A И**
- **AN И-НЕ**
- **O ИЛИ**
- **ON ИЛИ-НЕ**
- **X ИСКЛЮЧАЮЩЕЕ ИЛИ**
- **XN ИСКЛЮЧАЮЩЕЕ ИЛИ - НЕ**
- **O И перед ИЛИ**

Вы можете использовать следующие инструкции для выполнения вложенных функций :

- **A(И с открытием скобки**
- **AN(И – НЕ с открытием скобки**
- **O(ИЛИ с открытием скобки**
- **ON(ИЛИ - НЕ с открытием скобки**
- **X(ИСКЛЮЧАЮЩЕЕ ИЛИ с открытием скобки**
- **XN(ИСКЛЮЧАЮЩЕЕ ИЛИ- НЕ с открытием скобки**
- **) Закрытие скобки**

Вы можете закончить битовое логическое выражение с помощью одной из следующих инструкций:

- **= Присвоение**
- **R Сброс**
- **S Установка**

Для изменения результата логической операции Вы можете использовать следующие:

- **NOT Инверсия RLO**
- **SET Установка RLO в 1**
- **CLR Сброс RLO в 0**
- **SAVE Сохранение RLO в BR регистре**

Другие инструкции реагируют на появление нарастающего или падающего фронта RЛО:

- **FN Выделение падающего фронта RLO**
- **FP Выделение нарастающего фронта RLO**

Пример применения основных битовых инструкций представлен на рисунке 8

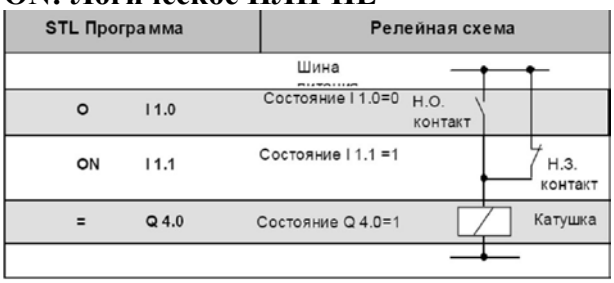
A: Логическое И

AN: Логическое И – НЕ

O: Логическое ИЛИ

O: Логическое ИЛИ

ON: Логическое ИЛИ-НЕ



X: Исключающее ИЛИ

A(: И с открывающей скобкой

= Присвоение

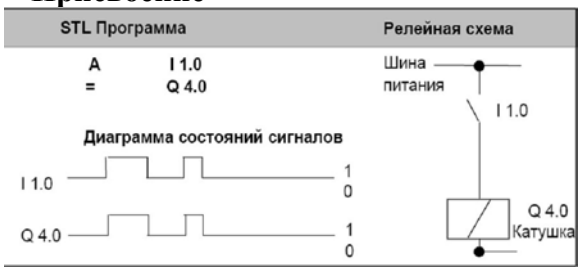


Рисунок 8 - Пример применения основных битовых инструкций

3.2 Обзор инструкций сравнения

Аккумуляторы 1 (ACCU1) и 2 (ACCU2) сравниваются в соответствии с выбранным Вами типом сравнения :

== ACCU1 равен ACCU2
<> ACCU1 не равен ACCU2
> ACCU1 больше ACCU2
< ACCU1 меньше ACCU2
>= ACCU1 больше или равен ACCU2
<= ACCU1 меньше или равен ACCU2

Если условие сравнения выполняется, то RLO получает значение "1". Биты слова состояния CC1 и CC0 изменяются в соответствии с выполняемыми инструкциями сравнения на “меньше”, ”равно” или ”больше”.

Вы можете использовать следующие типы сравнения:

• ? I : Сравнение чисел типа Integer (16-битовых),

• ? D : Сравнение чисел типа Double Integer (32-битовых),

• ? R : Сравнение чисел с плавающей точкой (32-битовых).

3.3 Обзор операций со счетчиками

Счетчики являются функциональным элементом языка программирования STEP7 функций счета. Счетчики имеют область, зарезервированную для них в памяти CPU. Эта область памяти резервирует по одному 16-битному слову для каждого адреса счетчика. При программировании в STL Вы можете адресоваться к 256 счетчикам. Инструкции счета являются единственными функциями, которые имеют доступ к области памяти счетчиков. Вы можете изменять значение счетчика, используя следующие инструкции:

• FR Деблокировка счетчика

• L Загрузка текущего значения счетчика в ACCU 1

• LC Загрузка текущего значения счетчика в BCD-коде в ACCU 1(с плавающей запятой)

• R Сброс счетчика

• S Установка счетчика на заданное значение

• CU Прямой счет

• CD Обратный счет

3.4 Обзор инструкций перехода

Вы можете использовать инструкции перехода для управления ходом выполнения программы, позволяя ей прерывать последовательную процедуру выполнения и возобновить обработку с другого места. Вы можете использовать инструкцию циклического выполнения LOOP для обработки участка программы несколько раз подряд.

Операндом команды перехода и циклического выполнения является метка. Метка может состоять из четырех символов, первый из которых должен быть буквой. Метка перехода должна заканчиваться двоеточием "**:**" и ставиться в строке, содержащей инструкцию. Вы можете использовать следующие инструкции перехода для безусловного прерывания линейного выполнения программы:

• JU Безусловный переход

• JL Распределенный переход

Вы можете использовать следующие инструкции условного перехода в зависимости от результата логической операции (RLO) предыдущей логической инструкции:

• JC Переход при RLO = 1

• JCN Переход при RLO = 0

• JCB Переход при RLO = 1 с сохранением в BR

• JNB Переход при RLO = 0 с сохранением в BR

Следующие инструкции выполняют условный переход в зависимости от результатов вычислений:

• JZ Переход при нулевом результате

• JN Переход при ненулевом результате

• JP Переход при положительном результате

• JM Переход при отрицательном результате

• JPZ Переход при неотрицательном результате
• JMZ Переход при отрицательном или нулевом результате
• JUO Переход при недействительном результате

3.5 Обзор инструкций загрузки и передачи

Инструкции загрузки (L) и передачи (T) позволяют программировать обмен информацией между различными областями памяти или между областями памяти и периферийными модулями ввода - вывода. CPU выполняет эти инструкции в каждом цикле как безусловные команды, т.е. результат логической операции на них не влияет.

Следующие инструкции загрузки и передачи могут использоваться:

• L Загрузка
• L STW Загрузка битов слова состояния в ACCU 1
• LAR1 AR2 Загрузка в адресный регистр1 (AR1) значения из AR2
• LAR1 <D> Загрузка в адресный регистр 1 константы (32 -битовый указатель)
• LAR1 Загрузка в адресный регистр 1 значения из ACCU 1
• LAR2 <D> Загрузка в адресный регистр константы (32 -битовый указатель)
• LAR2 Загрузка в адресный регистр 2 значения из ACCU 1
• T Передача
• T STW Передача ACCU 1 в слово состояния
• TAR1 AR2 Передача адресного регистра 1 в адресный регистр 2
• TAR1 <D> Передача адресного регистра 1 в целевую область (32-битовый указатель)
• TAR2 <D> Передача адресного регистра 2 в целевую область (32-битовый указатель)
• TAR1 Передача адресного регистра 1 в ACCU 1
• TAR2 Передача адресного регистра 1 в ACCU 1
• CAR Обмен содержимым адресных регистров 1 и 2

STL		Комментарий
L	IB10	//Загрузка входного байтаIB10 в ACCU 1-L-L.
L	MB120	//Загрузка меркерного байта MB120 в ACCU 1-L-L.
L	DBB12	//Загрузка байта данных DBB12 в ACCU 1-L-L.
L	DIW15	//Загрузка слова данных блока экземпляра DIW15 в ACCU 1-L.
L	LD252	//Загрузка локального двойного слова данных LD252 в ACCU 1.
L	P# I 8.7	//Загрузка указателя в ACCU 1.
L	OTTO	//Загрузка параметра "OTTO" в ACCU 1.
L	P# ANNA	//Загрузка указателя на параметр в ACCU 1. (Это позволяет загрузить относительный адресный указатель на параметр. Для вычисления абсолютного смещения в блоке мультиэкземпляре FB, к полученному относительному указателю должно быть прибавлено содержимое адресного регистра AR2.)

Рисунок 9 – Пример применения инструкций загрузки

3.6 Обзор математических инструкций с целыми числами

Математические инструкции производят обработку содержимого аккумуляторов 1 и 2. Старое содержимое аккумулятора 1 при выполнении инструкции загрузки сдвигается в аккумулятор 2. При выполнении инструкции, результат сохраняется в аккумуляторе 1, содержимое аккумулятора 2 остается неизменным. В CPU с четырьмя аккумуляторами после выполнения математической инструкции, содержимое аккумулятора 3 копируется в аккумулятор 2, а содержимое аккумулятора 4 в аккумулятор 3. Старое содержимое

аккумулятора 4 не меняется. С помощью математических инструкций, Вы можете выполнять следующие операции с **двумя целыми числами** (16 и 32 бита):

- **+I Сложение ACCU 1 и ACCU 2 в формате Integer (16-бит)**
- **-I Вычитание ACCU 1 из ACCU 2 в формате Integer (16-бит)**
- ***I Умножение ACCU 1 на ACCU 2 в формате Integer (16-бит)**
- **/I Деление ACCU 2 на ACCU 1 в формате Integer (16-бит)**
- **+ Сложение констант типа Integer (16, 32 Бит)**
- **+D Сложение ACCU 1 и ACCU 2 в формате Double Integer (32-бит)**
- **-D Вычитание ACCU 1 из ACCU 2 в формате Double Integer (32-бит)**
- ***D Умножение ACCU 1 и ACCU 2 в формате Double Integer (32-бит)**
- **/D Деление ACCU 2 на ACCU 1 в формате Double Integer (32-бит)**
- **MOD Получение остатка от деления в формате Double Integer (32-бит)**

Смотрите также оценку битов слова состояния при выполнении математических инструкций над целыми числами.

STL		Комментарий
L	IW10	//Загрузка значения IW10 в ACCU 1-L.
L	MW14	//Перемещение содержимого ACCU 1-L в ACCU 2-L. Загрузка значения MW14 в ACCU 1-L.
+I		//Сложение ACCU 2-L и ACCU 1-L; сохранение результата в ACCU 1-L.
T	DB1.DBW25	//Содержимое ACCU 1-L (результат) передается в DBW25 блока данных DB1.

Рисунок 10 - Пример применения математических инструкций

3.7 Обзор математических инструкций над числами с плавающей точкой

Математические инструкции производят обработку содержимого аккумуляторов 1 и 2 . Старое содержимое аккумулятора 1 при инструкции загрузки сдвигается в аккумулятор 2. При выполнении инструкции, результат сохраняется в аккумуляторе 1, содержимое аккумулятора 2 остается неизменным.

В CPU с четырьмя аккумуляторами после выполнения математической инструкции, содержимое аккумулятора 3 копируется в аккумулятор 2, а содержимое аккумулятора 4 в аккумулятор 3. Старое содержимое аккумулятора 4 не меняется. IEEE 32-битовые числа с плавающей точкой соответствуют типу данных REAL. Вы можете использовать следующие инструкции с плавающей точкой для обработки **двух 32-битовых IEEE чисел с плавающей точкой**:

- **+R Сложение ACCU 1 и ACCU**
- **-R Вычитание ACCU 1 из ACCU 2**
- ***R Умножение ACCU 1 и ACCU 2**
- **/R Деление ACCU 2 на ACCU 1**

Вы можете использовать следующие инструкции с плавающей точкой для обработки **одного 32-битового IEEE числа с плавающей точкой**:

- **ABS Модуль числа**
- **SQR Вычисление квадрата**
- **SQRT Вычисление квадратного корня**
- **EXP Вычисление экспоненты**
- **LN Вычисление натурального логарифма**
- **SB Вычисление синуса угла**
- **COS Вычисление косинуса угла**
- **TAN Вычисление тангенса угла**
- **ASB Вычисление арксинуса**
- **ACOS Вычисление арккосинуса**
- **ATAN Вычисление арктангенса**

3.8 Обзор инструкций сдвига

С помощью инструкций сдвига Вы можете побитно сдвигать содержимое младшего слова аккумулятора или весь аккумулятор (см. Регистры CPU) влево или вправо. Сдвиг на n битов влево умножает содержимое аккумулятора на 2n; сдвиг на n битов вправо делит содержимое аккумулятора на 2n. Например, если Вы сдвигаете значение 3 в двоичном коде на 3 бита влево, то получается десятичное значение 24. Если Вы сдвигаете десятичное значение 16 на 2 бита вправо, то в аккумуляторе получается двоичный

эквивалент десятичного значения 4. Число, задаваемое в виде параметра инструкции сдвига или, при его отсутствии, содержимое младшего байта аккумулятора 2, показывает, на сколько битов должен производиться сдвиг. Разряды, освобождающиеся вследствие операции сдвига, заполняются нулями **или** состоянием знакового бита (”0” в случае положительного числа, ”1” в случае отрицательного числа). Бит, сдвигаемый последним, загружается в бит CC1 слова состояния . Биты CC0 и OV сбрасываются в “0”. Вы можете оценить бит CC1 слова состояния с помощью операций перехода. Инструкции сдвига являются безусловными. Это значит, что они выполняются независимо от каких-либо условий. Они не влияют на бит RLO. Вы можете использовать следующие инструкции сдвига:

- **SSI Сдвиг вправо целого числа со знаком**
- **SSD Сдвиг вправо двойного целого числа со знаком**
- **SLW Сдвиг слова влево**
- **SRW Сдвиг слова вправо**
- **SLD Сдвиг двойного слова влево**
- **SRD Сдвиг двойного слова вправо**

STL		Комментарий
L	MW4	//Загрузка значения ACCU 1.
SRW	6	//Сдвиг бит со знаком ACCU 1 на шесть позиций вправо.
T	MW8	//Сохранение результата в MW8.

Рисунок 11 - Пример применения инструкций сдвига

3.9 Обзор инструкций с таймерами

Возможны следующие инструкции с таймерами:

- **FR Деблокировка таймера**
- **L Загрузка текущего значения таймера в ACCU 1 в формате Integer**
- **LC Загрузка текущего значения таймера в ACCU 1 в BCD - коде**
- **R Сброс таймера**
- **SD Таймер задержки включения**
- **SE Удлиненный импульс**
- **SF Таймер задержки выключения**
- **SP Импульс**
- **SS Таймер задержки включения с памятью**

В таблице 4 представлены основные типы таймеров, используемых в программах на языке STEP 7.

Таблица 4 - Основные типы таймеров и алгоритм их работы

Таймер	Описание
SP_PULSE Таймер «Импульс»	Максимальное время в течение которого выходной сигнал остается равным 1, совпадает с заданным временем t. Выход сбрасывается раньше, если входной сигнал меняется на 0.
SE_PEXT Таймер «Импульс с памятью»	Выходной сигнал остается равным 1 в течение заданного времени независимо от того, как долго остается равным 1 входной сигнал.
SD_ODT Таймер «Задержка включения»	Выходной сигнал устанавливается в 1 только по истечении заданного времени, при этом входной сигнал все еще должен быть равен 1.
SS_ODTS Таймер «Задержка включения с памятью»	Выходной сигнал устанавливается в 1 только по истечении заданного времени независимо от того, как долго остается равным 1 входной сигнал.
SF_OFFDT Таймер «Задержка выключения»	Выходной сигнал устанавливается в 1, когда устанавливается в 1 входной сигнал, и остается равным 1, пока таймер работает. Отсчет времени начинается, когда входной сигнал меняется с 1 на 0.

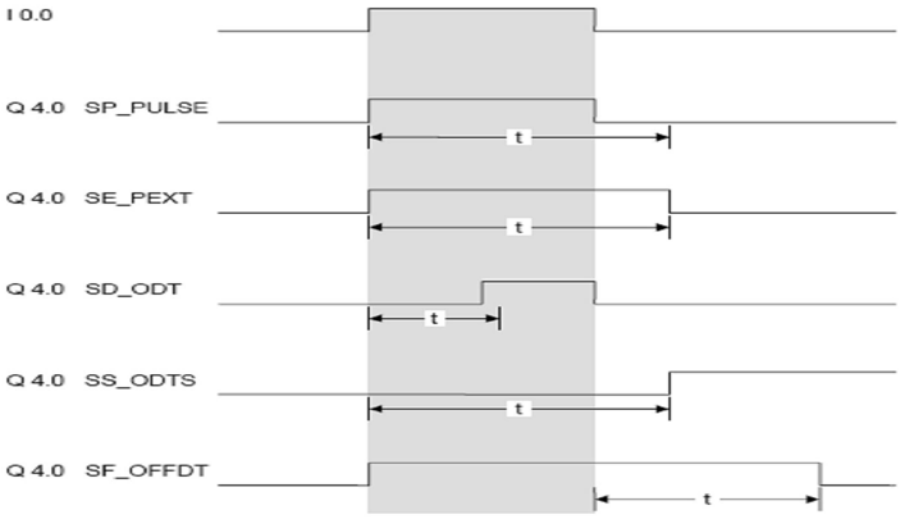


Рисунок 12 - Временные диаграммы работы таймеров

STL		Комментарий
A	I 2.0	
FR	T1	//Деблокировка таймера T1.
A	I 2.1	
L	S5T#10s	//Задание времени 10 секунд в ACCU 1.
SI	T1	//Запуск T1 как импульс.
A	I 2.2	
R	T1	//Сброс T1.
A	T1	//Оценка состояния таймера T1.
=	Q 4.0	
L	T1	//Загрузка текущего значения таймера T1 в двоичном коде.
T	MW10	

Рисунок 13 - Пример программы с использованием таймеров

3.10 Обзор инструкций с аккумуляторами и адресными регистрами

В Вашем распоряжении имеются следующие инструкции для обработки содержимого одного или обоих аккумуляторов:

- **TAK Обмен содержимым аккумуляторов ACCU 1 и ACCU 2**
- **PUSH Для CPU с двумя аккумуляторами**
- **PUSH Для CPU с четырьмя аккумуляторами**
- **POP Для CPU с двумя аккумуляторами**
- **POP Для CPU с четырьмя аккумуляторами**
- **ENT Ввод в стек аккумуляторов**

- **LEAVE Вывод в стек аккумуляторов**
- **INC Инкремент ACCU 1-L-L**
- **DEC Декремент ACCU 1-L-L**
- **+AR1 Сложение ACCU 1 с адресным регистром AR 1**
- **+AR2 Сложение ACCU 1 с адресным регистром AR 2**
- **BLD Инструкция отображения программы**
- **NOP 0 Нулевая инструкция**
- **NOP 1 Нулевая инструкция**

3.11 Примеры использования инструкций языка STEP 7 для составления программ.

Пример 1: Управление лентой транспортера

На рисунке 14 показана лента транспортера, которая может приводиться в движение с помощью электродвигателя. В начале транспортера имеются две кнопки: S1 для запуска и S2 для останова. В конце транспортера тоже имеются две кнопки: S3 для запуска и S4 для останова. Транспортер можно запускать или останавливать с любого конца. Также датчик S5 останавливает транспортер, когда предмет, находящийся на ленте, достигает конца.

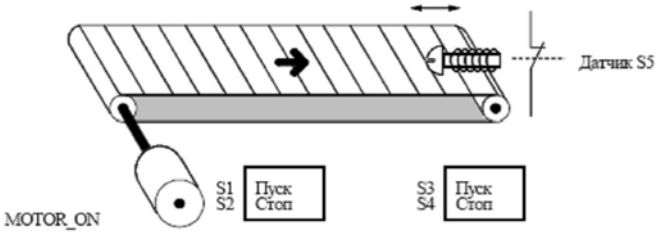


Рисунок 14 - Внешний вид объекта управления (транспортера)

Абсолютное и символьное программирование

Вы можете написать программу для управления лентой транспортера, показанного на рисунке 14, используя **абсолютные значения** или их **символьные имена**, представляющие различные компоненты конвейера. В начале следует создать таблицу символов для того, чтобы поставить в соответствие выбранным символьным именам абсолютные адреса.

Таблица 5 - Таблица адресации

Компонент системы	Абсолютный адрес	Символ	Таблица символов	
Push Button Start Switch	I 1.1	S1	I 1.1	S1
Push Button Stop Switch	I 1.2	S2	I 1.2	S2
Push Button Start Switch	I 1.3	S3	I 1.3	S3
Push Button Stop Switch	I 1.4	S4	I 1.4	S4
Sensor	I 1.5	S5	I 1.5	S5
Motor	Q 4.0	MOTOR_ON	Q 4.0	MOTOR_ON

STL		Комментарий
O	I 1.1	//Нажатие любой кнопки включает конвейер.
O	I 1.3	
S	Q 4.0	
O	I 1.2	//Нажатие любой стоповой кнопки или срабатывание нормально закрытого концевика останавливает конвейер.
O	I 1.4	
ON	I 1.5	
R	Q 4.0	

Рисунок 15- Список инструкций для управления конвейером

Программа с абсолютными адресами		Программа с символьными именами	
O	I 1.1	O	S1
O	I 1.3	O	S3
S	Q 4.0	S	MOTOR_ON
O	I 1.2	O	S2
O	I 1.4	O	S4
ON	I 1.5	ON	S5
R	Q 4.0	R	MOTOR_ON

Рисунок 16 - Варианты программы управления

Пример2: Математические инструкции с целыми числами

Следующий пример программы показывает, как использовать арифметические операции с целыми числами и команды L и T для вычисления результата следующего уравнения:

MD4 = ((IW0 + DB5.W3) x 15) / MW2

STL		Комментарий
L	IW0	//Загрузка значения из IW0 в аккумулятор 1.
L	DB5.DBW3	//Загрузка слова данных DBW3 из глобального блока данных DB5 в //аккумулятор1. Старое содержимое аккумулятора 1 предварительно сдвигается //в аккумулятор 2.
+I	I 0.1	//Сложение содержимого аккумуляторов 1 и 2. Результат сохраняется в //младшем слове аккумулятора 1. Содержимое аккумулятора 2 и старшее слово //аккумулятора 1 остаются неизменными.
L	+15	//Загрузка константы +15 в аккумулятор 1. Содержимое аккумулятора 1 //предварительно сдвигается в аккумулятор 2.
*I		// Перемножение содержимого аккумуляторов 1 и 2. Результат сохраняется в //младшем слове аккумулятора 1. Содержимое аккумулятора 2 и старшее слово //аккумулятора 1 остаются неизменными.
L	MW2	// Загрузка значения из MW2 в аккумулятор 1. Старое содержимое аккумулятора //1 предварительно сдвигается в аккумулятор 2.
/I		//Деление содержимого аккумулятора 2 на содержимое аккумулятора 1. //Результат сохраняется в младшем слове аккумулятора 1. Содержимое //аккумулятора 2 и старшее слово аккумулятора 1 остаются неизменными.
T	MD4	//Передача окончательного результата в двойное слово MD4. Содержимое //аккумуляторов не изменяется.

Рисунок 17 - Список инструкций программы

STEP7 - ИНСТРУМЕНТАЛЬНЫЙ ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ РАЗРАБОТКИ, ТЕСТИРОВАНИЯ И ДОКУМЕНТИРОВАНИЯ ПРОГРАММ

Основными утилитами пакета STEP 7, которые доступны из папки SIMATIC - STEP 7, являются:

- SIMATIC Manager;
- LAD, STL, FDB – Programming S7;
- Memory Card Parameter Assignment;
- NetPro – Configuring Networks;
- PID Control Parameter Assignment;
- S7 SCL – Programming S7 Blocks;
- S7-GRAPH – Programming Sequential Control System;
- S7-PDIAG – Configuring Process Diagnostic;
- S7-PLCSIM Simulating Modules;
- Setting the PG-PC Interface;
- Configure SIMATIC Workspace.

Основной программой STEP 7 является **SIMATIC Manager**, который позволяет производить основные операции с проектом, такие как создание, сохранение, открытие, а также управлять работой проекта, запускать различные утилиты, связывать их между собой и т.д.

Программа LAD, STL, FDB – Programming S7 Blocks – редактор, позволяющий программировать блоки, основываясь на одном из трех представлений языка программирования. Язык LAD – Ladder Diagram (контактный план) – использует представление программы в виде коммутационной схемы, состоящей из переключателей, линий связи, ключей и т.п. STL – Statement List (список операторов) – язык, подобный ассемблеру. FDB – Function Block Diagram – функциональная схема, основанная на логических элементах, триггерах и т.п.

Утилита **Memory Card Parameter Assignment** позволяет сохранять пользовательскую программу в память EPROM (электрически программируемая постоянная память), используя программатор или, в случае персональной ЭВМ, на внешнее устройство.

Программа **NetPro** – Configuring Networks позволяет конфигурировать промышленные сети, такие как MPI, PROFIBUS или Industrial Ethernet.

Утилита **PID Control Parameter Assignment** позволяет автоматизировать процедуру расчета и настройки параметров ПИД-регуляторов, используемых в системах управления.

С базовым пакетом обычно поставляются специальные утилиты, позволяющие проводить создание программ различными способами, такими как: написание программ на языке программирования высокого уровня SCL, который похож на паскаль, с помощью программы S7 SCL; графическая разработка программ в виде последовательности шагов и переходов между ними посредством утилиты S7-GRAPH. Могут также поставляться дополнительные пакеты.

S7-PDIAG – Configuring Process Diagnostic – это программа, используемая для диагностики проектов.

Утилита **S7-PLCSIM** Simulating Modules предназначена для программной имитации работы контроллера, что позволяет разрабатывать проекты и проверять и отлаживать работу программ без подключения реального оборудования.

Программа **Setting the PG-PC Interface** применяется для установки параметров локальных станций, подключенных к многоточечному интерфейсу MPI.

Configure SIMATIC Workspace позволяет конфигурировать проекты, создаваемые с использованием нескольких терминалов.

5.1 ЭЛЕМЕНТЫ ПРОЕКТА В SIMATIC MANAGER

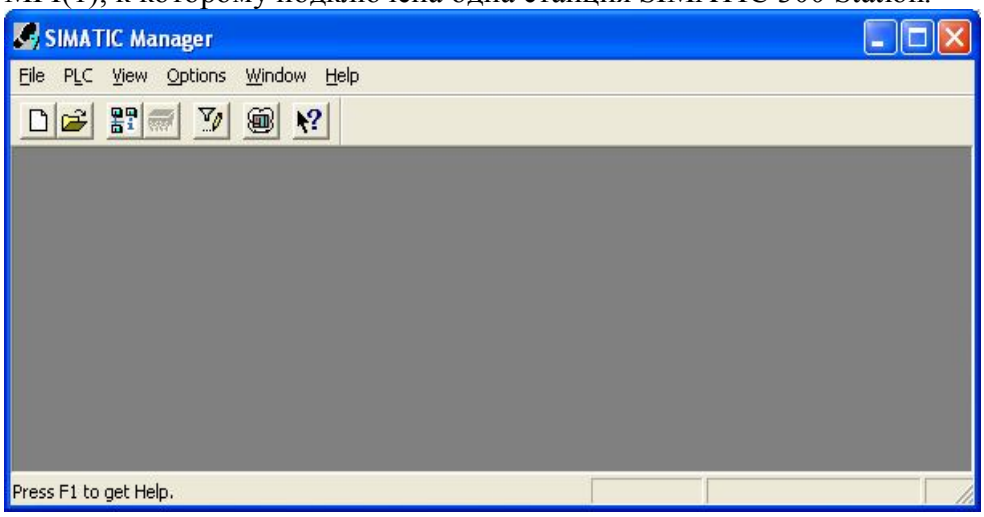
SIMATIC Manager – это графический интерфейс для редактирования объектов S7 (проектов, файлов пользовательских программ, блоков, оборудования станций и инструментов). Основное окно утилиты показано на рисунке 16 .

Основными элементами панели главного меню программы SIMATIC Manager являются разделы File, PLC, View, Options, Window и Help, содержание которых зависит от текущего окна.

На панели инструментов вынесены наиболее часто используемые кнопки.
Вначале рассмотрим структуру проекта в SIMATIC Manager, которая показа на рисунке 17.

Данные хранятся в проекте в виде объектов. Объекты в проекте размещаются в древовидной структуре, которая показана в левой части рисунке 17. Она подобна структуре, используемой в Windows Explorer. Различаются только иконки объектов. Содержимое правой части окна SIMATIC Manager зависит от выбранного в левой части объекта.

На самом верхнем уровне, который (см. рисунке 17) называется S7_Pro1, расположен сам проект. Каждый проект представляет базу, в которой хранятся все относящиеся к нему данные. Элементами проекта являются сети и их элементы – станции и другие узлы. В данном примере проект S7_Pro1 содержит многоточечный интерфейс MPI(1), к которому подключена одна станция SIMATIC 300 Station.



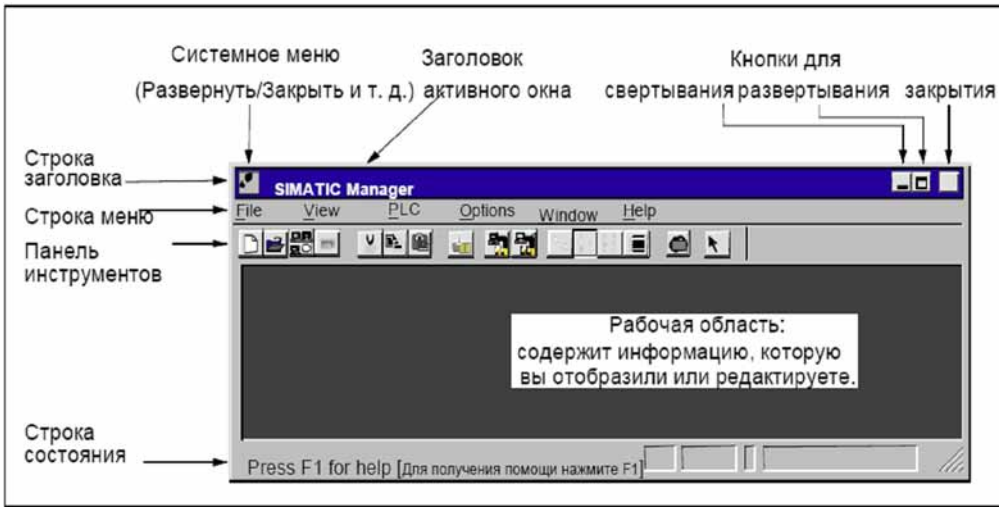


Рисунок 18 - Меню и панель инструментов SIMATIC Manager

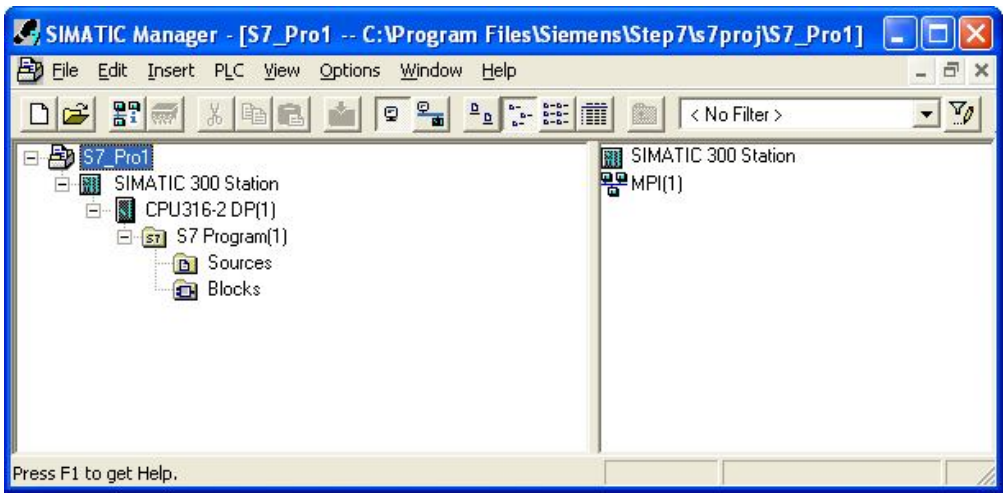


Рисунок 19 - Структура проекта в SIMATIC Manager

На втором уровне, который показан на рисунке 18, находятся станции, которые являются исходными объектами для конфигурирования аппаратуры. Здесь хранится информация о конфигурации аппаратуры и параметрах модулей. На рисунке 18 уровень станций содержит один элемент – SIMATIC 300 Station, который в свою очередь содержит контроллер CPU316-2DP(1). Другое оборудование можно просматривать утилитой Hardware.

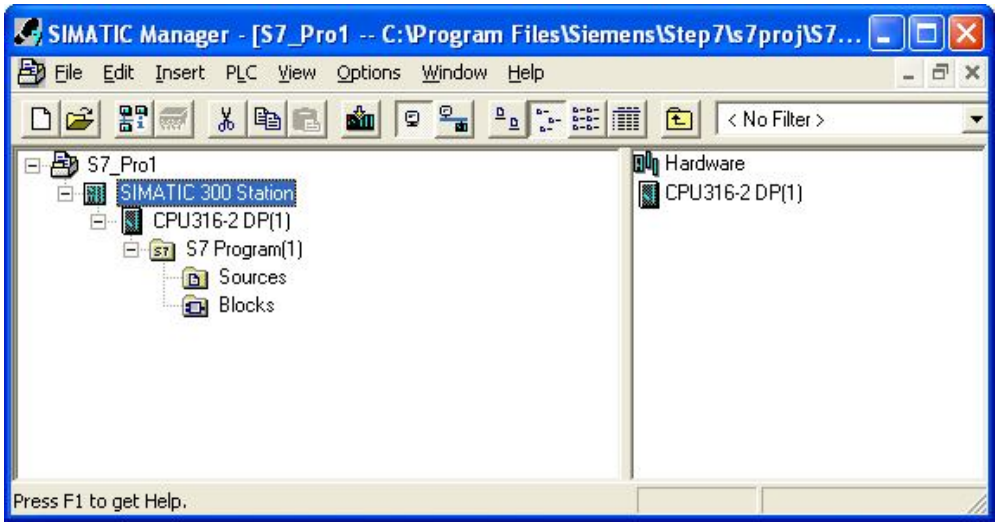


Рисунок 20 - Уровень станции в SIMATIC Manager

В свою очередь, процессор CPU316-2DP(1) содержит пользовательские программы, в данном случае S7 Program(1), которые могут быть написаны в виде блоков Blocks или исходных кодов Sources. Последующие уровни зависят от содержимого предыдущих.

На рисунке 19 показан один из примеров уровня Blocks

Основными блоками, которые используются в STEP 7, являются:

а) организационный блок, например OB1, который является основной циклически исполняемой программой;

б) функция, например FC1, применяемая для замены типовых или часто повторяющихся блоков;

в) функциональный блок, например FB1, в отличие от функции имеет отдельную память в глобальном пространстве, называемую блоком данных, за счет чего функциональный блок может сохранять свои переменные в общем адресном пространстве;

г) блоки данных, например DB1, наличие которых обусловлено гарвардской архитектурой контроллеров.

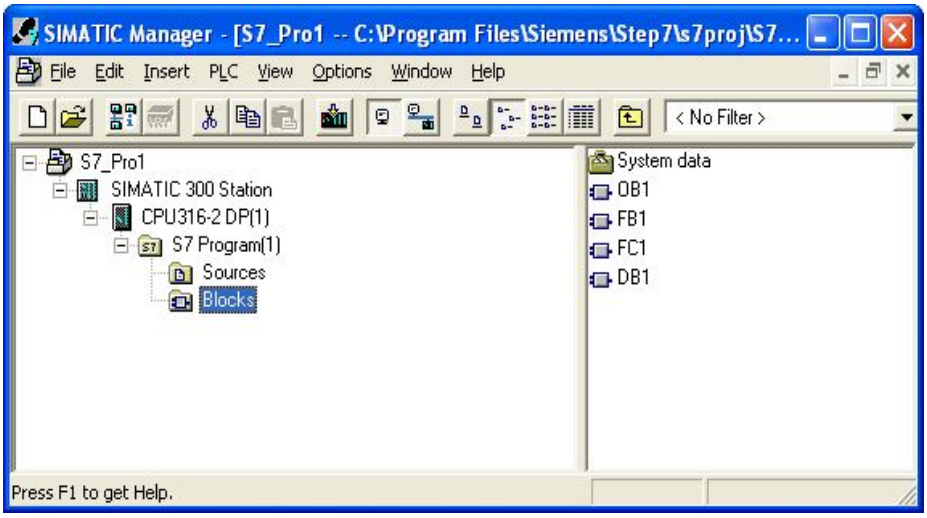


Рисунок 21 - Уровень блоков

5.2 СОЗДАНИЕ ПРОЕКТА В SIMATIC MANAGER

Рассмотрим основные этапы создания проекта с помощью мастера «New Project Wizard», который находится в разделе «File» главного меню утилиты SIMATIC Manager. Создание проекта состоит из четырех шагов, которые демонстрируются на рисунках 20-23.

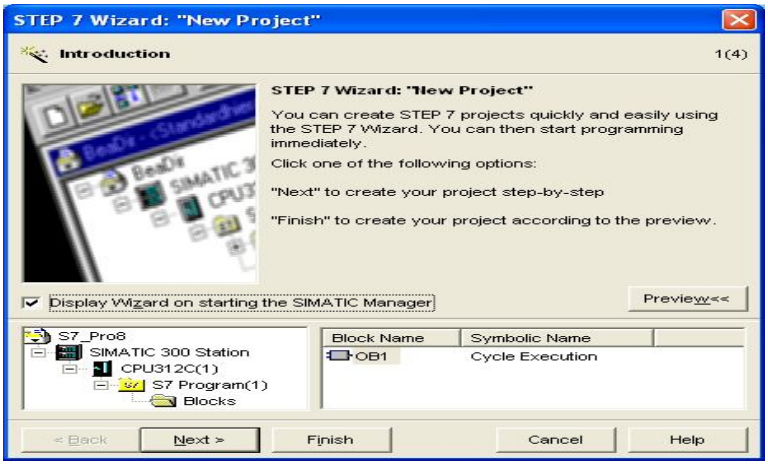


Рисунок 22 - Первый шаг создания проекта

В первом окне, показанном на рисунке 20, пользователю предлагается выбрать структуру проекта по умолчанию, показанную в двух окнах, нажав кнопку «Finish», или продолжить пошаговое создание проекта, нажав кнопку «Next». При выборе пошагового режима появляется второе окно, демонстрируемое на рисунке 21, в котором предлагается выбрать тип процессора (процессоров) из списка и установить его MPI-адрес – адрес подключения к многоточечному интерфейсу (Multi Point Interface).

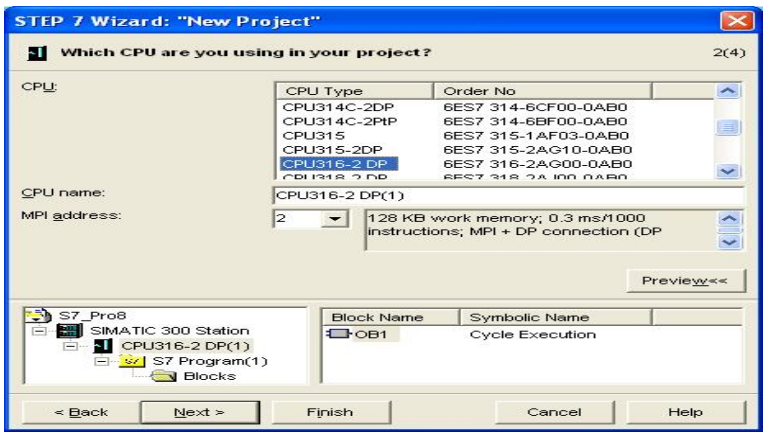


Рисунок 23 - Второй шаг при создании проекта

Нажав кнопку «Next», можно перейти к третьему шагу, который поясняется на рисунке 22.

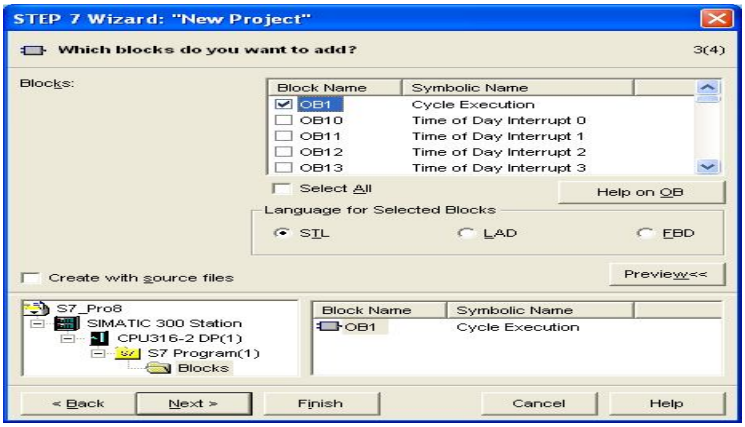


Рисунок 24 - Третий шаг при создании проекта

На третьем этапе можно выбрать тип организационных блоков, которые планируется использовать в программе, например: блок OB1 «Cycle Execution» – означает циклически исполняемую программу; блок OB10 «Time of Day Interrupt» – означает прерывание, вызываемое по времени суток; блок OB20 «Time Delay Interrupt» – означает прерывание, вызываемое по истечении заданного временного интервала; блок OB30 «Cycle Interrupt» – это циклически вызываемое прерывание; блок OB40 «Hardware Interrupt 1» – программа, выполняемая по приходу прерывания от внешней аппаратуры; блок OB60 «Multicomputing Interrupt» – предназначен для прерываний, вызываемых различными способами. Кроме того, существует ряд блоков, предназначенных для обработки ошибок, таких как ошибка таймера (OB80 «Cycle Time Fault»), ошибка системы питания (OB81 «Power Supply Fault»), ошибка ввода-вывода (OB82 «I/O Point Fault»), ошибка процессора (OB84 «CPU Fault»), ошибка загрузки организационного блока (OB85 «OB Not Loaded Fault»), отсутствие контакта в соединительном разъеме (OB86 «Loss of Rack Fault»), ошибка соединения (OB87 «Communication Fault»). Также существует три блока для перезапуска: полный перезапуск (OB100 «Complete Restart»); обычный перезапуск (OB101 «Restart»); холодный перезапуск (OB102 «Cold Restart»). Последние два блока – ошибка программирования контроллера (OB121 «Programming Error») и ошибка доступа к блоку (OB122 «Module Access Error»).

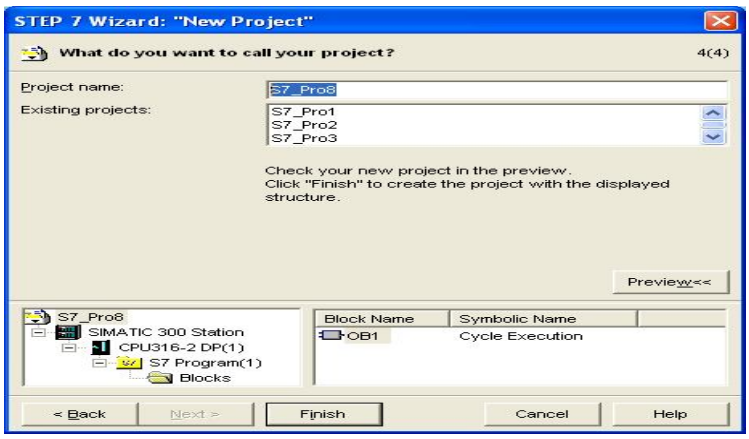


Рисунок 25 - Присвоение имени проекту

Кроме того, в окне, показанном на рисунке 22, имеется возможность установить язык программирования, наиболее удобный для пользователя – STL (список операторов), LAD (контактный план) или FBD (функциональный оператор).

В последнем окне, показанном на рисунке 23, предлагается задать имя проекта. Результатом работы «New Project Wizard» является созданный проект, который появляется после нажатия кнопки Finish в последнем диалоговом окне.

Добавление новых элементов в проект осуществляется через меню «Insert». Сохранение проекта, копирование блоков происходит стандартным способом через буфер или с помощью манипулятора.

5.3 КОНФИГУРИРОВАНИЕ АППАРАТНЫХ СРЕДСТВ

Конфигурирование аппаратных средств проекта осуществляется посредством утилиты «Hardware Configuration». Чтобы запустить указанную программу, необходимо перейти на уровень станций, который показан на рисунке 18, и двойным щелчком нажать кнопку «Hardware», в результате чего появится окно, показанное на рисунке 24.

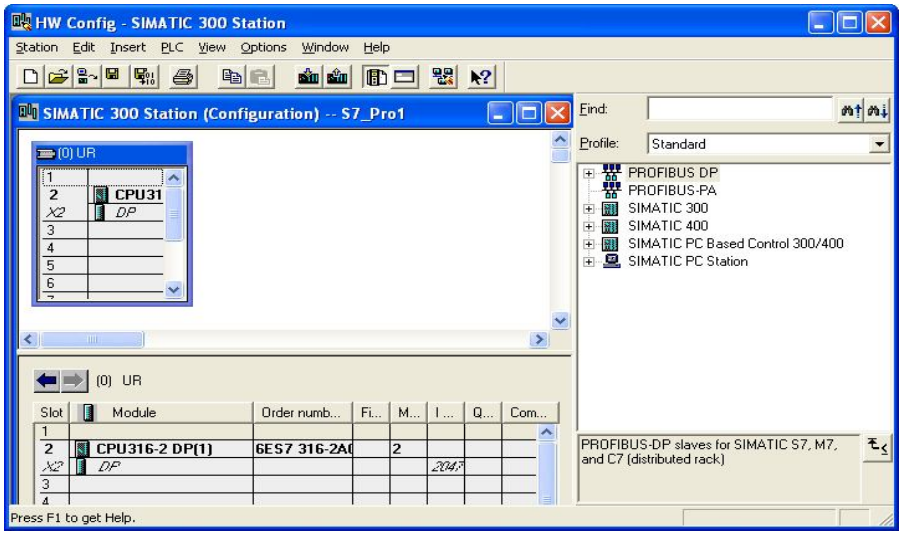


Рисунок 26 - Окно конфигурации аппаратной части

Рабочее поле утилиты Hardware Configuration разбито на три основные части. В левой верхней части показаны стойки с отдельными слотами. Они расположены на шинах. В левой нижней части находится таблица с адресами входов-выходов, различных блоков и контроллеров. В правой части окна расположена библиотека элементов, из которых можно собирать стойки.

Создание аппаратной части начинается с добавления стойки (Rack), которая находится в соответствующем каталоге. Например, при создании станции SIMATIC 300

необходимо открыть каталог элементов SIMATIC 300 и из папки Rack-300 добавить элемент Rail. Добавление можно производить либо двойным щелчком, либо перетаскиванием по технологии «drag & drop».

Если требуется установить блок питания, то необходимо вставлять его в слот 1 стойки. Соответствующий модуль станции SIMATIC 300 находится в группе PS-300.

CPU контроллера можно найти в каталоге CPU-300, он вставляется в слот 2.

S7-300 слот 3 зарезервирован для интерфейсного модуля IM, необходимого для многоуровневых конфигураций, поэтому на рисунке 24 этот слот пустой.

Если эта позиция должна быть резервирована для последующей фактической установки интерфейсного модуля, то необходимо вставить в фактическую конфигурацию холостой модуль DM370 DUMMY из каталога SM-300\Special-300.

Начиная с четвертого слота, можно вставлять сигнальные модули. Можно добавить на выбор до 8 сигнальных блоков (SM), коммуникационных процессоров (CP) или функциональных модулей (FM). Необходимо отыскивать нужный модуль в папке и вставлять его, выбирая слот в стойке.

В стандартной конфигурации в стойку может входить процессор, блок питания и модули ввода и вывода, которые бывают аналоговые или дискретные.

Чтобы просмотреть адресное пространство, образованное модулями стойки, необходимо войти в меню «View»->«Address Overview», в результате чего появится окно, показанное на рисунке 25. В данном окне отражаются те блоки, которые имеют входы или выходы, в данном случае модуль дискретного ввода DI32xDC24V и блок дискретного вывода DI4xNAMUR.

В первом столбце Type указывается тип адресного пространства: I – для входов, Q – для выходов.

Во втором и третьем столбцах Addr. from и Addr. to указывается диапазон адресов в байтах, который занимает данное устройство. В данном случае модуль дискретного ввода DI32xDC24V имеет 32 входа, поэтому он занимает 4 байта с номерами от 0 до 3. Следующие байт с номером 4 занимает блок дискретного вывода DI4xNAMUR.

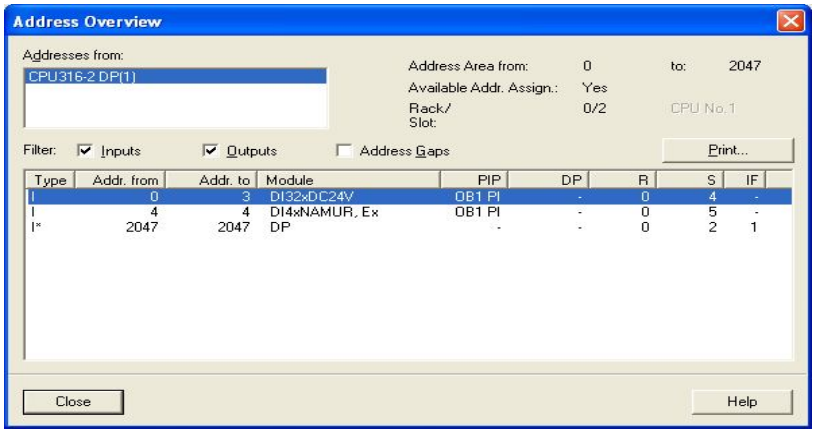


Рисунок 27 - Адресное пространство аппаратуры станции

В следующих двух столбцах указываются названия блоков и организационный блок, который осуществляет опрос входов и назначение выходов. В данном случае оба модуля принадлежат одному блоку OB1.

Столбец R отображает номер стойки, а столбец S – слота для модуля. В данном случае модуль DI32xDC24V занимает слот 4, а модуль DI4xNAMUR – слот 5.

Столбец DP используется для системы распределенных выходов, а IF в тех случаях, когда используется специальный интерфейсный модуль при программировании системы на C++.

Чтобы получить доступ к свойствам блока достаточно открыть его пиктограмму с помощью двойного щелчка мыши. Основные параметры сосредоточены в контроллере,

поэтому рассмотрим его свойства. Свойства контроллера отображаются в окне, которое содержит девять раскрывающихся вкладок. Вкладка «General», показанная на рисунке 26, содержит информацию о типе модуля, его название и, если он программируемый, MPI адрес. Чтобы назначить адрес многоточечного интерфейса, например в случае создания многоконтроллерной конфигурации, достаточно нажать кнопку «Properties, в которой имеется возможность задать параметр «Adress».

На рисунке 27 показана вкладка «Startup», которая позволяет задавать характеристики запуска. Для S7-300 единственным возможным типом запуска является «Warm restart». Только некоторые варианты имеют вариант «Cold restart».

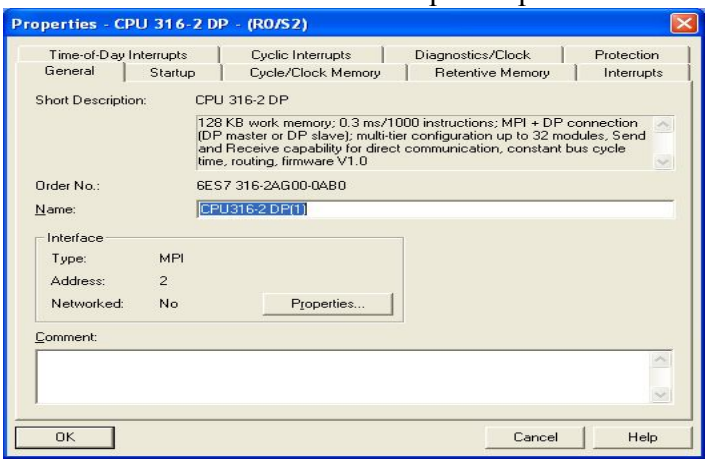


Рисунок 28 - Общие параметры контроллера S7-300

Параметр «Finished Message by Modules (ms)» означает максимальное время на получение сигнала готовности модулей. Если модули не подтверждают приема параметров в пределах установленного времени, то реальная конфигурация не соответствует проектной.

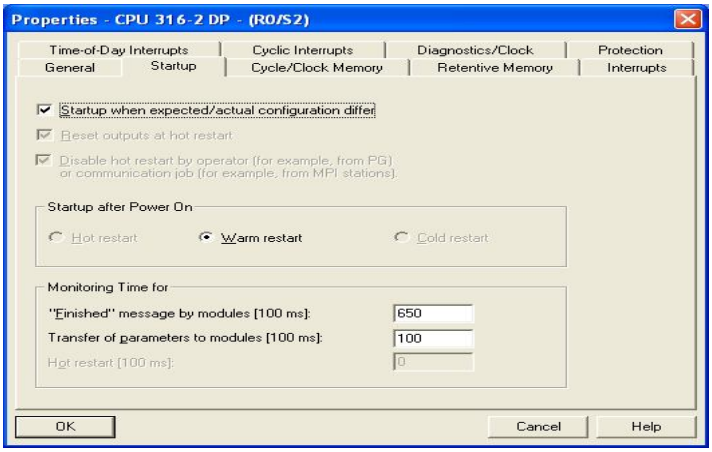


Рисунок 29 - Параметры запуска контроллера S7-300

Параметр «Transfer of Parameters to Modules» – максимальное время для передачи параметров в настраиваемые модули, после того как получен сигнал готовности.

Параметр «Startup when expected/actual configuration differ» позволяет для контроллеров со встроенным интерфейсом распределенных входов-выходов DP и для S7-400 запретить или разрешить запуск, если реальная конфигурация оборудования отличается от проектной. Остальные контроллеры запускаются в любом случае.

Также для контроллеров S7-400 можно указать сбрасывать выходы при горячем перезапуске – «Reset outputs at hot restart» и запретить перезапуск от другой станции или оператора.

Закладка «Retentive Memory» (сохраняемая память) используется для определения областей памяти, которые должны сохраняться после пропадания питания или переходе процессора из режима STOP в RUN. В обоих случаях в S7-300 выполняется полный

перезапуск, при котором блоки (OB, FC, FB, DB), хранимые в памяти с батарейной подпиткой, а также меркеры, таймеры и счетчики, определенные как сохраняемые, не изменяются. Только несохраняемые меркеры, таймеры и счетчики сбрасываются при запуске CPU.

На рисунке 28 показана вкладка «Cycle/Clock Memory» – временные параметры контроллеров.

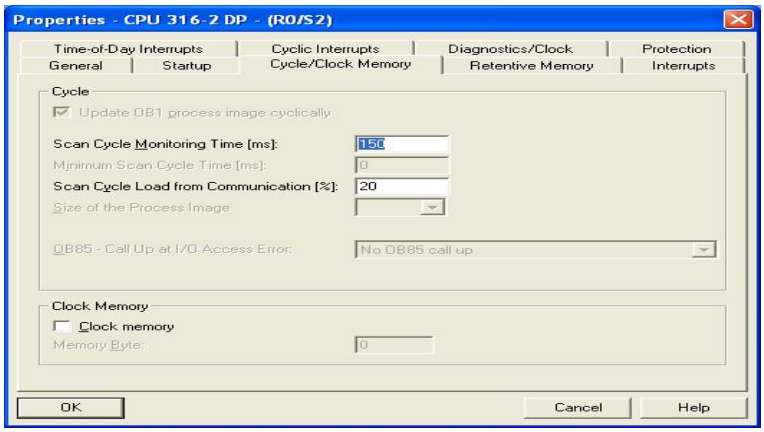


Рисунок 30 - Временные параметры контроллера S7-300

Закладка «Cycle/Clock Memory» позволяет с помощью параметра «Scan Cycle Monitoring Time (ms)» задавать время контроля цикла. Если это время превышено, то контроллер переходит в режим STOP. Возможными причинами превышения времени могут быть коммуникационные процессы, часто от событий прерываний, ошибки в программе.

Параметр «Cycle Load from Communication (%)» задает время связи, например время передачи данных в другой контроллер через многоточечный интерфейс. Это время ограничивается значением, выраженным в процентах от текущего времени цикла.

Например, ограничение связи до 20% приведет к тому, что для времени цикла сканирования 100 мс максимальное время для связи составит 20 мс.

Для синхронизации работы программы используется синхробайт «Clock Memory», который является байтом из области меркеров. Его биты периодически изменяют свое значение, причем каждый бит в синхробайте связан с конкретной частотой.

Во вкладке «Protection», показанной на рисунке 29, можно изменять параметры защиты.

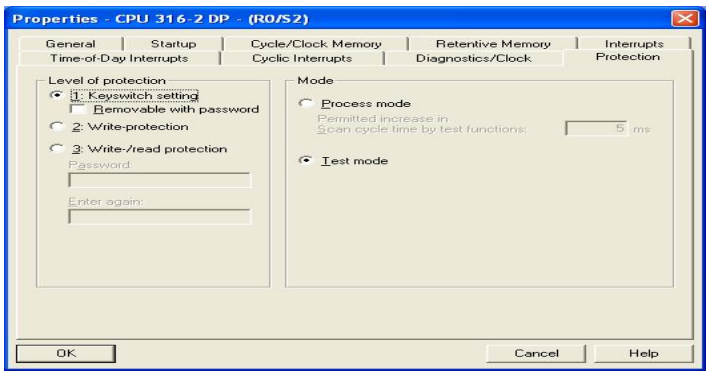


Рисунок 31 - Параметры защиты контроллера S7-300

Вкладка «Protection» позволяет задать три уровня защиты. На первом уровне «Keyswitch setting» можно работать без ограничений. Если назначен пароль, то он определяет следующие ограничения: для уровня 1 – в режиме останова (STOP) возможен полный доступ, а в режиме работы (RUN) только чтение; для уровня защиты 2 – существует доступ только для чтения, а для уровня 3 – невозможно ни чтение, ни запись

независимо от режима работы. Чтобы ввести пароль, необходимо либо задать уровень 2 или 3, либо выбрать режим «Removable with password» на первом уровне.

Вкладка «Diagnostic/Clock», показанная на рисунке 30, позволяет с помощью флага «Report cause of stop» обнаруживать причину останова, а также синхронизировать часы нескольких контроллеров (раздел «Synchronization») и вводить коррекцию для часов (раздел «Correction factor»). Например, если часы отстают на 5 с в сутки, то можно ввести параметр «Correction factor» +5000 ms.

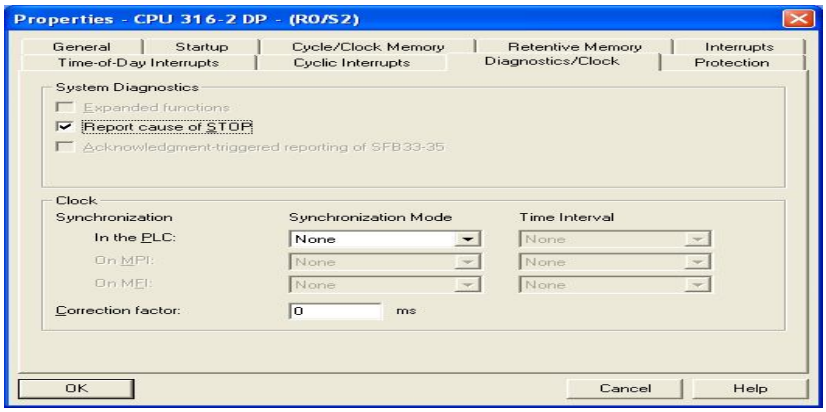


Рисунок 32 - . Параметры диагностики и синхронизации контроллера S7-300

Кроме перечисленных, также имеются вкладки, определяющие параметры прерываний «Interrupts», «Cyclic Interrupts», «Time-of-Day Interrupts».

Вкладка «Interrupts» показана на рисунке 31. Приоритеты программ задаются по возрастанию, т.е. чем выше номер, тем более высокий приоритет. Для циклических прерываний имеется возможность указать интервал выполнения через параметр «Execution», а для прерываний, вызываемых по времени суток, указываются параметры «Start Date» и «Time of Day».

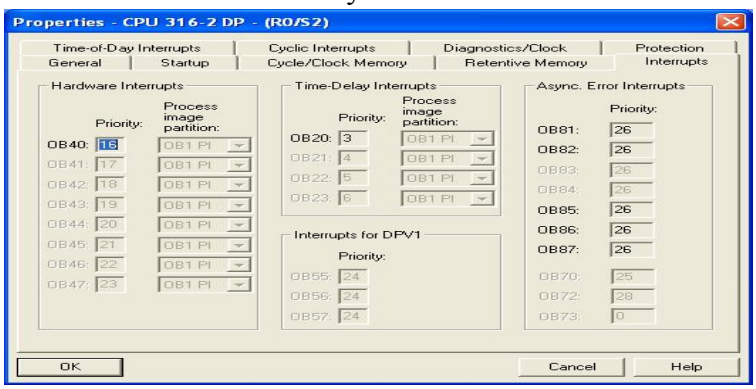


Рисунок 33 - Вкладка «Interrupts»

Таблица 1

Чтобы сохранить конфигурацию, нужно войти в меню «Station» и выбрать вкладку «Save». При выборе вкладки «Save and Compile» конфигурация загружается в блоки данных DB проекта. Чтобы проверить правильность конфигурации, можно воспользоваться меню «Station» -> «Consistency Check». Загрузка конфигурации в контроллер или его эмулятор возможна через меню, «PLC» -> «Download», при этом контроллер должен находиться в режиме STOP.

Для входных и выходных модулей можно задавать их адреса, однако необходимо помнить, что после перезапуска контроллера снова применяется адресация по умолчанию.

5.4 РЕДАКТИРОВАНИЕ БЛОКОВ

При программировании блоков в STEP 7 можно применять три языка программирования: LAD (контактный план), STL (список операторов), FBD (функциональный план).

Отличия языков показаны на рисунке 34, где демонстрируется логическая функция *И*. Язык LAD удобен для инженеров-электриков, STL – для программистов, а FBD – для инженеров-схемотехников. Отметим, что переход от одного языка к другому в SIMATIC Manager может осуществляться автоматически.

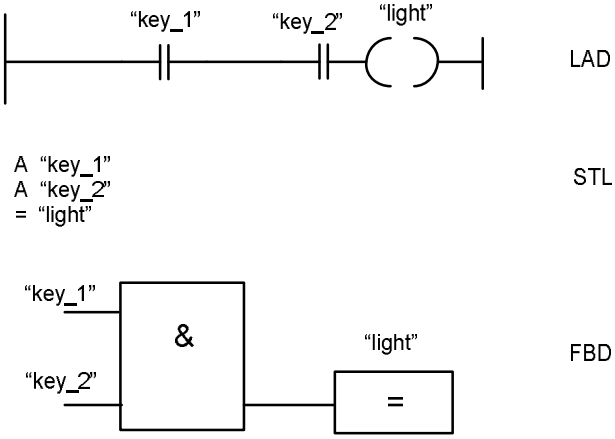


Рисунок 34 - Функция И в STEP 7

Чтобы начать редактировать существующий блок, необходимо перейти на уровень блоков и в окне, показанном на рисунке 21, дважды щелкнуть мышью нужный блок. При создании нового блока в том же окне нужно войти в меню «Insert» -> «S7 Block» и выбрать соответствующий организационный блок. При этом на экране появится окно редактора LAD/STL/FBD, показанное на рисунке 35.

Основными элементами редактора являются: таблица деклараций; раздел кода; элементы.

Таблица деклараций – часть программного блока. Она используется для объявления переменных и параметров блока.

Раздел кода содержит саму программу, разделенную, если это необходимо, на отдельные сегменты, называемые networks.

Содержимое окна элементов зависит от выбранного языка программирования. Элементы представляют собой функции, функциональные блоки или библиотеки, как часто используемые, так и специальные. Чтобы добавить элемент в программу, можно дважды щелкнуть на него или перетащить его указателем.

Выбор языка программирования осуществляется через меню «View» окна, показанного на рисунке 35. В этом меню можно выбрать одно из трех представлений программы: LAD, STL или FBD. Замена языка программирования возможна для синтаксически законченной программы.

Можно писать программы или сегменты на LAD или FBD и затем автоматически преобразовывать сегменты программы в STL. Однако результат этого преобразования не всегда является наиболее эффективным решением для STL (программа, созданная непосредственно на STL, может быть короче).

Обратное преобразование из STL в LAD или FBD не всегда возможно. Сегменты программы, которые не могут быть преобразованы, остаются в STL. При преобразованиях никакие сегменты программы не теряются.

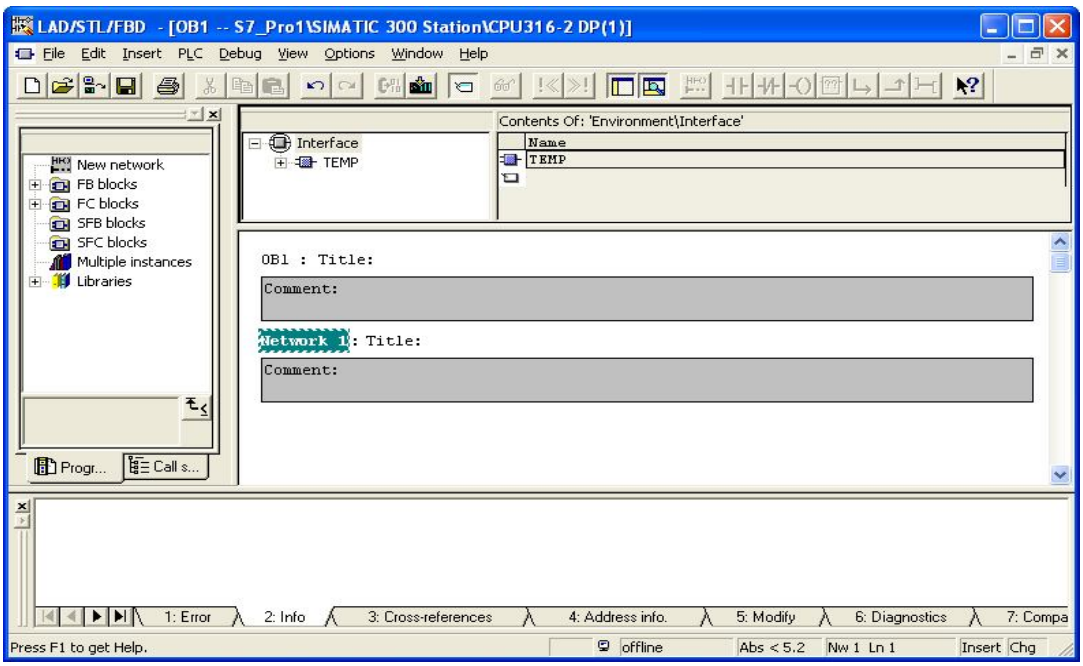


Рисунок 35 - Окно редактора LAD/STL/FBD

При редактировании блоков в FBD или LAD часто используемые элементы представлены кнопками в панели инструментов. Можно щелкать на них мышью, чтобы установить эти элементы на выбранную в программе позицию. Другие элементы можно вставить в программу из списка в любую позицию перетаскиванием или в выделенную позицию – дважды щелкнув на элементе из списка. Соединять элементы можно посредством мыши, захватывая выход и перетаскивая его к нужному выходу.

Для добавления нового сегмента достаточно нажать на кнопку «New Network» в панели инструментов, после текущего сегмента добавляется новый сегмент.

При программировании на STL нужно знать инструкции для записи программы. Можно получить информацию о синтаксисе и функциональном назначении через подсказку: «Help» -> «Help on STL». В справке доступна следующая информация: «Statement List Instructions» – описывает все инструкции, которые имеются в этом языке программирования; «Working with Statement List» (работа с списком команд) – описывает список команд и основы синтаксиса, ввод и наблюдение констант, типы блоков, контакты и состояния сигнала.

При программировании на STL окно элементов содержит только список существующих блоков, которые могут быть вызваны из текущего блока. Сегменты вставляются в программу так же, как в редакторе LAD/FBD.

После редактирования блока его необходимо сохранить, что возможно только в случае отсутствия синтаксических ошибок.

Часто удобно писать программы в виде отдельных функций. В этом случае основная программа, например блок OB1, будет содержать список вызовов. Например, на языке STL:

```
CALL FC 1
CALL FC 2
```

Чтобы загрузить блоки в контроллер, нужно выделить их в окне на рис. 5 и воспользоваться меню «PLC» -> «Download». При этом нужно, чтобы был предварительно подключен контроллер или его программный эмулятор.

5.5 СИМВОЛЬНЫЕ ПЕРЕМЕННЫЕ

При написании программ в STEP 7 можно применять прямую адресацию.

Таблицу символов можно вызвать либо из окна SIMATIC Manager, находясь на уровне программ и выбрав значок «Symbols», либо непосредственно из редактора

LAD/STL/FBD, воспользовавшись меню «Options» -> «Symbol Table». При этом появляется окно, показанное на рисунке 36.

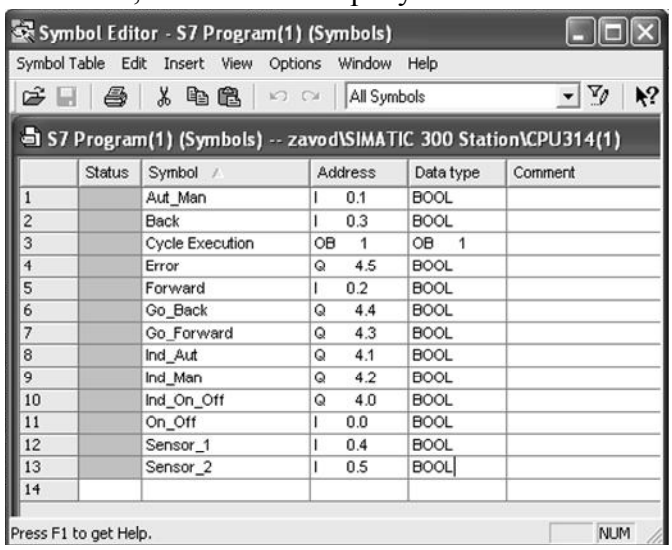


Рисунок 36 - Окно таблицы символов

Каждое символическое имя занимает одну строку в таблице. Пустая строка автоматически добавляется в конце таблицы для ввода нового символа. Символьная таблица является общей базой данных и может быть использована различными утилитами.

В меню Edit окна (см. рисунке 36) для поиска и замены текста доступны такие элементы, как: «Search For» – ввод текста для поиска; «Replace With» – ввод текста для замены; «Search Only» – поиск и выделение указанного текста; «From Cursor Down» – поиск вниз от курсора до последней строки символьной таблицы; «From Cursor Up» – поиск вверх от курсора до первой строки символьной таблицы; «Match Case» – поиск только указанного текста с анализом строчных и прописных букв; «Whole Word Only» – поиск указанного текста как отдельного слова; «All» – поиск по всей таблице, начиная с текущей позиции курсора; «Selection» – поиск текста только для выделенных строк таблицы.

При поиске адресов необходимо вводить звездочку «*» после идентификатора адреса, иначе адрес не будет найден.

Полезной функцией меню «View» окна таблицы символов является вкладка «Filter», показанная на рисунке 37, с помощью которой отображаются только символы, которые отвечают критерию, указанному в фильтре («symbol properties»).

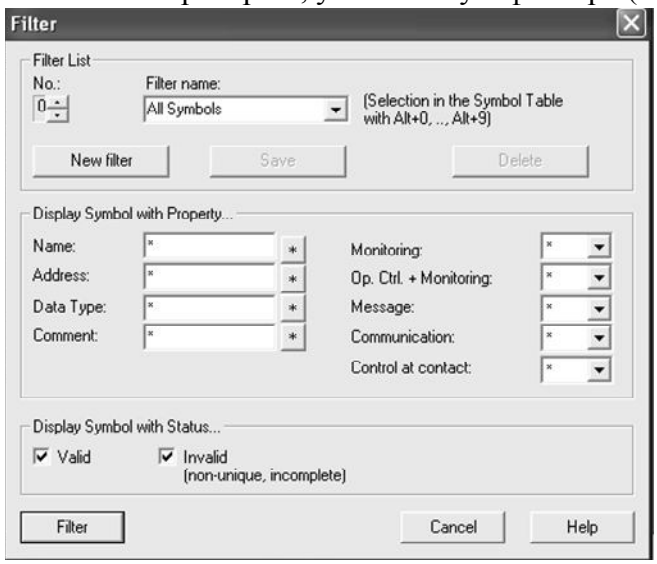


Рисунок 37 - Окно Filter

Можно применить различные критерии одновременно. Критерии, указанные в фильтре, объединяются. Также можно выбрать различные фильтры и объединить их согласно следующим свойствам: Name (имя), Address (адрес), Data type (тип данных), Comment (комментарий), Operator control and monitoring (управление и просмотр операторов), Communication (связь), Message (сообщение).

Допустимы сокращения символов: «*» и «?».

Например, если задать имя M*, то в таблице символов будут отображаться только те имена, которые начинаются с «M» и содержат любое число любых последующих символов.

Если задать имя SENSOR_?, то в таблице символов будут отображаться только те имена, которые начинаются с «SENSOR_» и содержат один любой символ в конце.

Метки «Valid» и «Invalid» позволяют отображать только уникальные или повторяющиеся символы. Если таблица символов длинная, то можно найти неоднозначные символы или адреса более быстро, воспользовавшись меню «View» -> «Filter» и установив атрибут «Invalid».

С помощью меню «View» -> «Sort» можно расположить данные в символьной таблице в алфавитном порядке. При этом можно задавать как возрастание или убывание как по именам, так и по адресам.

Команда меню «Symbol Table» -> «Export» позволяет преобразовать символьную таблицу в другой файловый формат, чтобы можно было работать с ней в других программах. Можно задавать следующие файловые форматы:

а) ASCII Format (*.ASC) – текстовый формат для Notepad и Word;

б) Data Interchange Format (*.DIF) – формат электронных таблиц для EXCEL;

в) System Data Format (*.SDF) – формат баз данных для ACCESS;

г) Assignment List (*.SEQ) – формат для STEP 5.

Аналогично с помощью команды меню «Symbol Table» -> «Import» можно импортировать символьные таблицы, подготовленные в других программах. Для этого необходимо выбрать меню «Symbol Table» -> «Import», затем выбрать файловый формат в диалоговом окне «Import», задать маршрут к директории в списке «Find in», ввести имя файла в поле «File Name» и нажать «OK».

С помощью команды меню «Options» -> «Edit Symbols» или щелчком правой кнопки мыши на адресе с последующим выбором пункта выпадающего меню «Edit Symbol» можно назначать символические имена для абсолютных адресов непосредственно при редактировании программы. Имена автоматически вводятся в символьную таблицу. Имена, которые уже есть в символьной таблице, отображаются другим цветом. Они не могут снова быть использованы в таблице.

При необходимости изменить присвоения в символьной таблице уже существующей программы нужно выбрать, что важнее – абсолютный или символический адрес. Чтобы сделать выбор в SIMATIC Manager нужно выбрать правой кнопкой мыши папку «Blocks», а затем выбрать пункт меню «Properties» и закладку «Blocks». Здесь можно выбрать «Absolute Value» (абсолютное значение) или «Symbol» (имя) в поле «Priority» (приоритет).

Пусть, например, старая запись в таблице символов была On = I 0.1, а новая On = I 1.0.

Если выбран приоритет адреса, то при изменении в символьной таблице абсолютный адрес операнда в программе не изменяется. В примере абсолютное имя I 0.0 было изменено на абсолютное имя I 1.0 в таблице символов. С установленным приоритетом по абсолютному значению, программа продолжает использовать вход I 0.0.

Если задан приоритет имени, то при смене абсолютного адреса операнда в символьной таблице, он изменяется во всей программе.

В примере выше, выход I 0.0 (имя символа «On») изменен на выход I 1.0 в символьной таблице. С установкой приоритета по имени, адрес I 0.0 изменяется на I 1.0 во

всей программе. Измененный адрес также сохраняется в символьной таблице. Таким способом можно изменять абсолютные адреса в программе пользователя, использующей символьные имена.

5.6 СИМУЛЯТОР КОНТРОЛЛЕРА PLCSIM

Проверку программной части без подключения реального оборудования можно проводить с помощью дополнительного пакета S7-PLCSIM. После того как проект готов, симулятор можно вызвать из главного окна SIMATIC Manager. Для этого в меню «Options» необходимо выбрать пункт «Simulate Modules», что приведет к запуску S7-PLCSIM, основное окно которого показано на рисунке 38.

С помощью значков, расположенных на панели инструментов симулятора S7-PLCSIM, можно добавлять для просмотра различные блоки и элементы контроллера:

а) IB – входная переменная;
б) QB – выходная переменная;
в) MB – биты памяти;
г) T – таймер;
д) C – счетчик;
е) Variable – переменная;
ж) Stacks – стек логических операций;
з) ACCUs – аккумуляторы и слово состояния;
и) Block Regs – блок регистров.

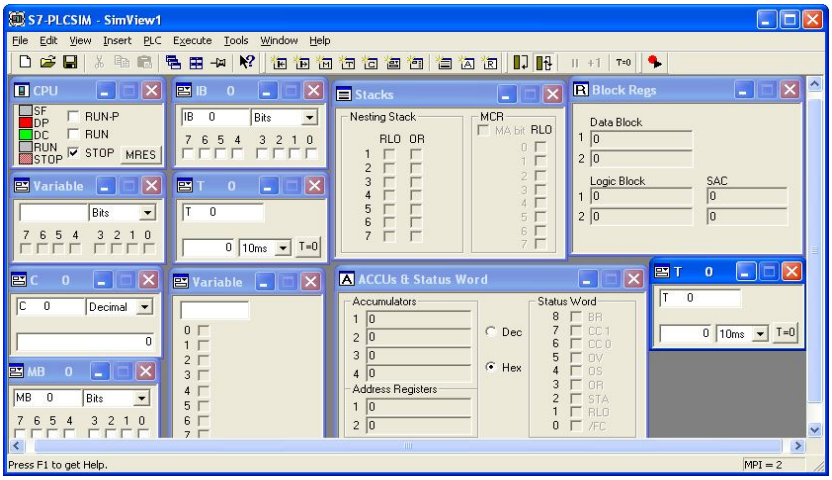


Рисунок 38 - Симулятор PLCSIM

В блоках а) – е) можно вводить свои адреса. Для того, чтобы можно было использовать символьную адресацию, нужно войти в меню «Tools» -> «Options» -> «Attach Symbols», в результате чего появится окно, показанное на рисунке 37. В этом окне нужно в разделе «Entry Point» указать вид блока, например проект или библиотека, имя проекта, в проекте выйти на уровень S7 Program и выбрать значок с именем Symbols.

После загрузки в эмулятор контроллера таблицы символов все переменные отображаются со своими именами как на рисунке 40.

Прежде чем проверять работу программы ее необходимо загрузить в контроллер. Это можно сделать либо из основного окна SIMATIC Manager, либо из редактора LAD/STL/FBD. В первом случае нужно выделить необходимые блоки, выбрать пункт меню «PLC» -> «Download». Во втором случае также используется меню «PLC» -> «Download», но загружается только текущий открытый блок. После этого нужно перейти в окно S7-PLCSIM и убедиться, что в его меню «PLC» установлен флаг «Power on». При загрузке блоком процессор симулятора должен находиться в режиме STOP.

Для того, чтобы запустить программу на выполнение, достаточно установить флажок RUN (циклическое выполнение) или RUN-P (однократное выполнение). При этом

можно мышкой менять входы и смотреть как изменяются выходы, отлаживая программу, записанную в контроллер.

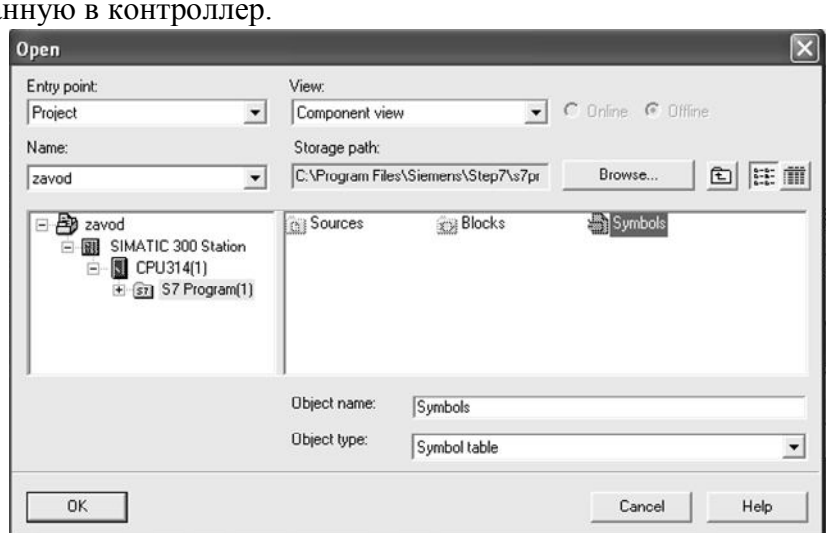


Рисунок 39 - Чтение таблицы символов в PLCSIM

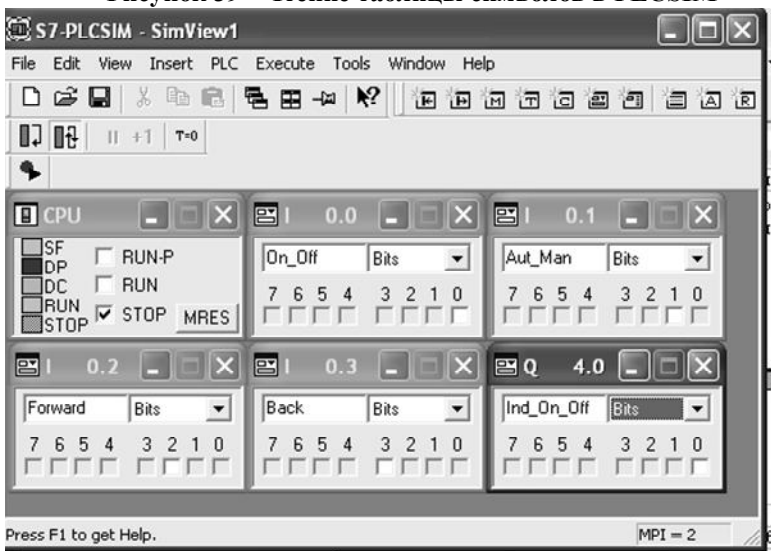


Рисунок 40 - Результат загрузки таблицы символов в PLCSIM

4. ПРИМЕР СОЗДАНИЯ ПРОЕКТА

5.1 Основная последовательность действий при планировании проекта автоматизации

Существует много способов планирования проекта автоматизации. Основная последовательность действий, которая можете быть использована для любого проекта, проиллюстрирована на следующем рисунке.

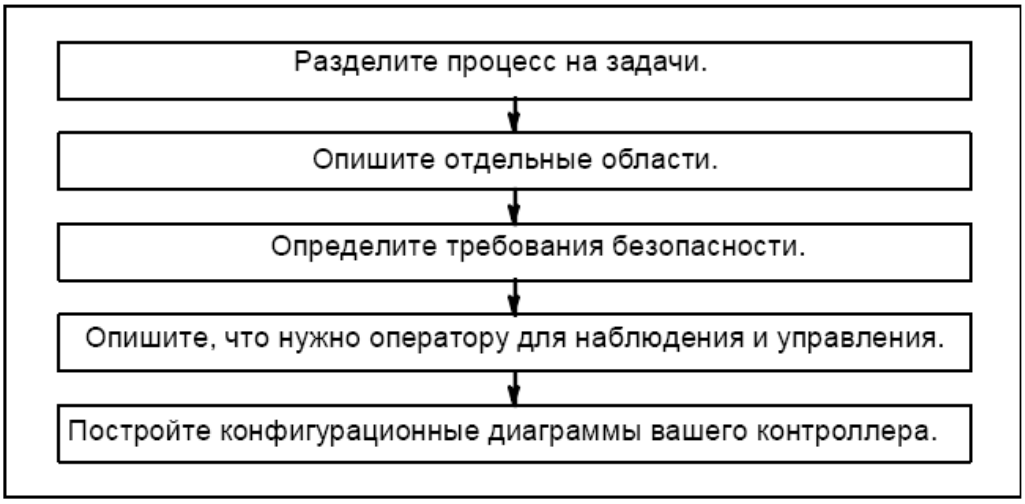


Рисунок 41 - Последовательность действий при составлении проекта

Деление процесса на задачи и области

Процесс автоматизации состоит из ряда отдельных задач. Путем выделения групп связанных задач внутри процесса и последующего разбиения этих групп на более мелкие задачи может быть определен даже самый сложный процесс.

Определение областей процесса

После определения процесса, подлежащего управлению, разделите процесс на связанные группы областей. Так как каждая группа разделена на более мелкие задачи, то задачи, необходимые для управления этой частью процесса, становятся менее сложными.

Описание отдельных функциональных областей

Описывая каждую область и задачу внутри Вашего процесса, Вы не только определяете функционирование каждой области, но и различные элементы, управляющие этой областью. Они включают в себя:

- электрические, механические и логические входы и выходы для каждой задачи
- блокировки и зависимости между отдельными задачами

Список входов, выходов и входов/выходов

Сделав физическое описание каждого устройства, подлежащего управлению, нарисуйте диаграммы входов и выходов для каждого устройства или группы задач.

Определение требований безопасности

Определите, какие дополнительные элементы необходимы для обеспечения безопасности процесса - на основе юридических требований и корпоративной политики в области охраны здоровья и безопасности. В свое описание Вам следует также включить все воздействия, которые элементы безопасности оказывают на области Вашего процесса.

Описание требуемых для оператора устройств отображения и управления

Каждый процесс требует интерфейса с оператором, который обеспечивает вмешательство человека в процесс. Часть спецификации проекта включает в себя проект пульта оператора.

Составление конфигурационной диаграммы

принять решение относительно типа управляющего оборудования, требующегося для проекта.

Принимая решение о том. какие модули Вы хотите использовать, Вы также определяете структуру программируемого контроллера. Составьте конфигурационную диаграмму, определяющую следующие аспекты:

- тип CPU
- количество и тип модулей ввода/вывода
- конфигурация физических входов и выходов

После выполнения всех выше изложенных этапов приступают к составлению программы в среде программирования.

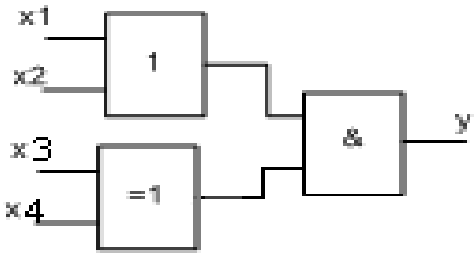


Рисунок 42 - Вид устройства, ревизующего функцию подлежащую программированию

Рассмотрим пример создания проекта реализующий функцию, представленную на рисунке 42 в среде программирования. Последовательность действий можно представить в следующем виде.

1. В основном окне SIMATIC Manager, показанном на рисунке 18, входим в меню «File» -> «New Project Wizard».
2. Нажимаем кнопку «Next».
3. Выбираем из списка: процессор CPU314, устанавливаем MPI адрес, равный 2, и нажимаем «Next».
4. Устанавливаем язык программирования STL, задаем тип организационного блока OB1 и нажимаем «Next».
5. Вводим имя проекта и нажимаем кнопку «Finish».
6. Переходим на уровень SIMATIC 300 STATION и запускаем программу Hardware двойным щелчком мыши.
7. В появившемся окне в стойке будет один процессор CPU314. Открываем справа библиотеку SIMATIC 300 и помещаем в стойку следующие элементы: блок питания PS 307 2A из папки PS-300; модуль цифрового ввода DI32xDC24V из папки SM-300\DI-300; модуль цифрового вывода DO32xDC24V/05A из папки SM-300\\DO300.
8. Входим в меню «Station» –> «Save and Compile». При отсутствии ошибок в папке блоков появляется объект «System Data».
9. Возвращаемся в SIMATIC Manager и переходим на уровень S7 Program(1) и запускаем редактор символов двойным щелчком на значок Symbols.
10. В окне Symbol Editor вносим в таблицу переменные, показанные в таблице 6. Сохраняем таблицу.

Таблица 6 - Переменные программы

Symbol	Address	Data Type	Comment
Main Program	OB 1	OB 1	
X1	I 0.0	BOOL	
X2	I 0.1	BOOL	
X3	I 0.2	BOOL	
X4	I 0.3	BOOL	
Y	Q 4.0	BOOL	

11. Переходим в окно SIMATIC Manager на уровень Blocks и через меню «Insert» - > «S7 Block» -> «Function» добавляем функцию с именем FC1 в список блоков, установив язык STL и нажав OK в появившемся окне.
12. Переходим в окно SIMATIC Manager на уровень Blocks и двойным щелчком на OB 1 запускаем редактор LAD/STL/FBD.
13. Набираем программу CALL FC 1 и сохраняем ее.
14. Переходим в окно SIMATIC Manager на уровень Blocks и двойным щелчком на FC1 открываем ее в редакторе LAD/STL/FBD.
15. Набираем в редакторе программу, реализующую функцию (X1 ИЛИ X2) И (X3 ИСКЛЮЧАЮЩЕЕ ИЛИ X4):

```
A(
    O    "X1"
    O    "X2"
    )
    A(
    X    "X3"
    X    "X4"
    )
    =    "Y"
```

и сохраняем ее.

Отметим, что логические операции обозначаются следующим образом: A – И; O – ИЛИ; X – ИСКЛЮЧАЮЩЕЕ ИЛИ, NOT – инверсия. Можно комбинировать логические операции с инверсией, например AN – И-НЕ.

16. Запускает симулятор S7-PLCSIM. Для этого в меню «Options» основного окна SIMATIC Manager выбираем пункт «Simulate Modules», что приводит к запуску S7-PLCSIM.
17. В основном окне SIMATIC Manager выделаем блоки OB1 и FC1 и выбираем пункт меню «PLC» вкладку «Download».
18. В окне утилиты Hadware входим в меню «PLC» -> «Download» и загружаем текущую конфигурацию в симулятор.
19. Переходим в окно S7-PLCSIM в меню «PLC» устанавливаем флаг «Power on». Загружаем таблицу символов в S7-PLCSIM, выбрав меню «Tools–>Options–>Attach Symbols». В появившемся окне нужно выбрать свой проект, перейти на уровень Program, выбрать Symbols и нажать OK.
20. В окне S7-PLCSIM добавляем пять окон: четыре входа и один выход. В редактируемых заголовках указываем имена переменных: X1, X2, X3, X4, Y или их абсолютные адреса. Во втором случае символьные имена добавятся автоматически. Если таблица символов не загружена, то процессор будет работать с той лишь разницей, что символьные адреса не будут отображаться.
21. Запускам программу, записанную в симулятор, на выполнение, установив флажок CPU в положение RUN. Это означает, что наша программа будет выполняться циклически.

Изменяя значения входов, измеряем значения выхода, заполняем таблицу истинности логической функции и проверяем выполнения логики работы устройства.

5. ПРАКТИЧЕСКИЕ ЗАДАНИЯ

По заданиям 1-6 составить программы и запуская в режиме эмуляции ПЛК проверить работоспособность программы и правильное выполнение заданного алгоритма работы.

Задания:

1. Создать программу обеспечивающая выдачу сигнала в течении 80 секунд после поступления управляющей команды.

2. Разработать программу для подачи звонков в техникуме по расписанию, с учетом особенностей субботнего расписания. Предусмотреть возможность ручного управления.
3. Разработать программу для управления воротами въезда на территорию завода. Ворота управляются вручную. Требования к системе управления воротами:
   - Ворота открываются и закрываются нажатием на кнопку в помещении вахты. Одновременно вахтер может контролировать работу ворот по сигнальным лампочкам. Одновременное нажатие кнопок не изменяет состояния ворот.
   - Перемещение ворот может быть остановлено в любое время.
   - Мигающий предупреждающий сигнал включается за 5 секунд до начала перемещения ворот, и остается включенным, пока ворота находятся в движении (до срабатывания датчика конечного положения).
   - Предохранительная планка гарантирует, что при закрытии ворот никто не получит травму и ничто не будет зажато или повреждено при срабатывании ее контактов обеспечивается остановка движения ворот.

Рисунок 43 - Внешний вид въездных ворот

4. Разработать программу для управления системой вентиляции цеха.

Назначение вентиляционной системы состоит в том, чтобы подавать свежий воздух в помещение цеха и вытягивать застоявшийся воздух из помещения. Требования к системе вентиляции:

В помещении имеется вытяжной вентилятор и приточный вентилятор свежего воздуха Рис. 1. Оба вентилятора контролируются датчиком потока. В помещении никогда не должно возникать избыточное давление. Приточный вентилятор должен включаться только при условии, что датчик потока сигнализирует о надежной работе вытяжного вентилятора. Если после короткой задержки воздушный поток не регистрируется, то система выключается и выдается сообщение о неисправности. Если один из вентиляторов выходит из строя, то загорается предупреждающая лампа.

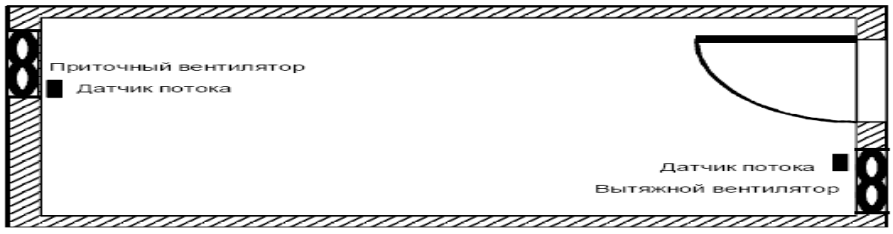


Рисунок 44 - Внешний вид системы вентиляции цеха

5. Разработать программу реализующее устройство подсчета количества деталей прошедших по поточной линии. Прохождение деталей фиксируется по сигналу оптического датчика. Предусмотреть сброс количества деталей в начале каждой смены (9.00).
6. Разработать программу для управления установкой непрерывного литья (Рис.1). Установка представляет собой ленту транспортера с установленными на ней формами. При доставке формы в зону разлива в течение 60секунд осуществляется заливка в форму расплавленного материала.

Требования к системе управления установкой:

- В начале транспортера имеются две кнопки: S1 для запуска и S2 для останова. В конце транспортера тоже имеются две кнопки: S3 для запуска и S4 для останова. Транспортер можно запускать или останавливать с любого конца. Данный блок программы оформить в виде подпрограммы;
- поступление формы в зону разлива фиксируется фотоэлектрическим датчиком.
- на время разлива транспортер останавливается;
- при завершении процесса разлива движение транспортера продолжается до поступления следующей формы;
- при отсутствии поступления форм в зону разлива более 40секунд транспортер останавливается и выдается сигнал тревоги.

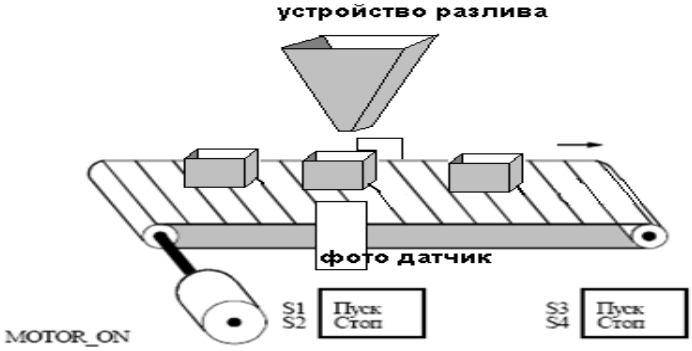


Рисунок 45 - Внешний вид установки непрерывного литья

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите основные утилиты STEP 7, которые использованы в работе для создания проекта. Для чего они предназначены?
2. Какие уровни содержит иерархическая структура проекта в SIMATIC Manager? Какие элементы они содержат?
3. Опишите этапы создания проекта с помощью мастера «New Project Wizard».
4. Какие типы организационных блоков могут быть реализованы в проекте SIMATIC Manager?
5. Опишите процесс конфигурации оборудования посредством утилиты Hardware Configuration. Каким образом распределены слоты станции S7-300?
6. Опишите систему приоритетов прерываний, используемую в контроллерах S7-300 и S7-400.
7. Какие языки программирования существуют в STEP 7? Чем они отличаются и каковы их особенности?
8. Что такое абсолютная и символьная адресация? Перечислите элементы таблицы символов.
9. Расскажите, какие настройки делаются при тестировании программы в программе S7-PLCSIM.
10. Какие типы переменных можно просматривать в программном симуляторе S7-PLCSIM?

7. ЛИТЕРАТУРА

1. Бергер Г. Автоматизация посредством STEP 7 с использованием STL и SCL и программируемых контроллеров SIMATIC S7-300/400. Siemens AG, Нюрнберг, 2001.
2. Программирование с помощью STEP 7 V5.3. Руководство 6ES7810-4CA07-8BW1. Siemens AG, Нюрнберг, 2004.