

Лекция 2

Среда разработки STEP 7-Micro/WIN 32. Установка коммуникационного соединения. Языки программирования. Редакторы LAD /STL/ FBD. Символьная и абсолютная адресация. Работа над проектом в STEP 7-Micro/WIN 32 (управление входами-выходами). Конфигурирование ЦПУ. Логические операции.

После изучения этой темы студенты:

- Будут знать и будут способны выбрать язык программирования LAD /STL/ FBD для требуемой задачи
- Будут уметь выполнять настройку редактора LAD /STL/ FBD
- Будут уметь редактировать, сохранять и загружать программу с использованием редактора LAD /STL/ FBD
- Будут уметь редактировать символы в редакторе LAD /STL/ FBD
- Будут уметь использовать глобальную таблицу символов.
- Способны описать различные виды используемой памяти и способы сохранения

Запуск Micro/WIN

Рабочий стол Windows имеет иконку "STEP 7-Micro/WIN 32" и пункт "STEP 7-Micro/WIN 32" в разделе SIMATIC стартового меню. Запустить эту программу, как и любое приложение Windows, можно двойным щелчком мышью на иконке или выбором пункта в стартовом меню.

Для открытия STEP 7-Micro/WIN дважды щелкните на символе STEP 7-Micro/WIN или выберите команду меню **Start > SIMATIC > STEP 7 MicroWIN 32 V4.0** [Пуск > SIMATIC > STEP 7 MicroWIN 32 V4.0 (Рис.1)]

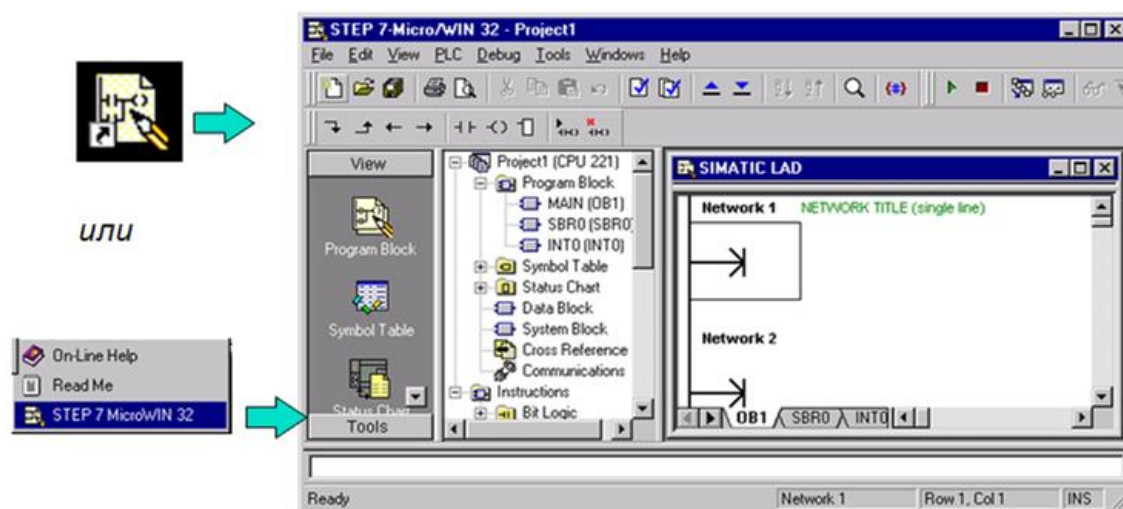


Рис.1 Окно STEP 7-Micro/WIN

Элементы окна

Панель заголовка Панель заголовка содержит имя окна и кнопки управления окном (Рис.2).

Строка меню Содержит все меню, доступные для активного окна.

Панель инструментов Содержит наиболее часто используемые команды меню в форме кнопок с изображениями.

Панель навигации Содержит иконки для активации функций программы.

Дерево команд Показывает все элементы проекта и все команды, доступные в активном редакторе программ (LAD, FBD или STL).

Окно вывода Когда программа компилируется, в выходном окне появляются информационные сообщения.

Строка состояния Показывает текущее состояние и другую информацию (Рис.2).

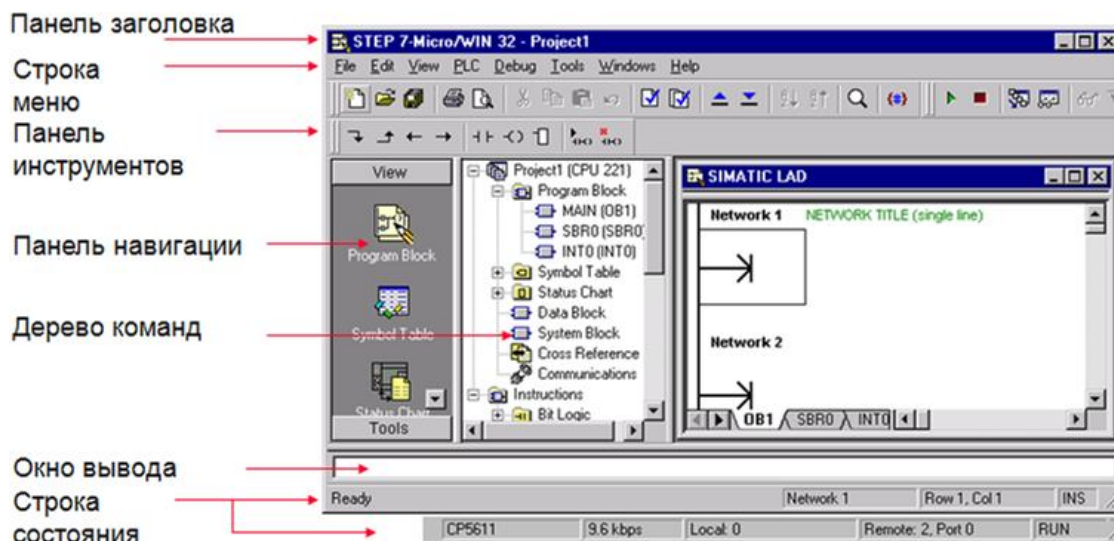


Рис.2 Элементы окна

Рабочая область для создания программы управления

Навигационная панель (Рис.3) предлагает группы символов для доступа к различным функциям программирования STEP 7-Micro/WIN.

Дерево команд отображает все объекты проекта и команды, необходимые для создания программы управления. Отдельные команды из этого дерева можно тащить в свою программу или вставлять команду двойным щелчком в текущее положение курсора в редакторе программ.

Редактор программ содержит логику программы и таблицу локальных переменных, в которой можно назначить символические имена для временных локальных переменных.

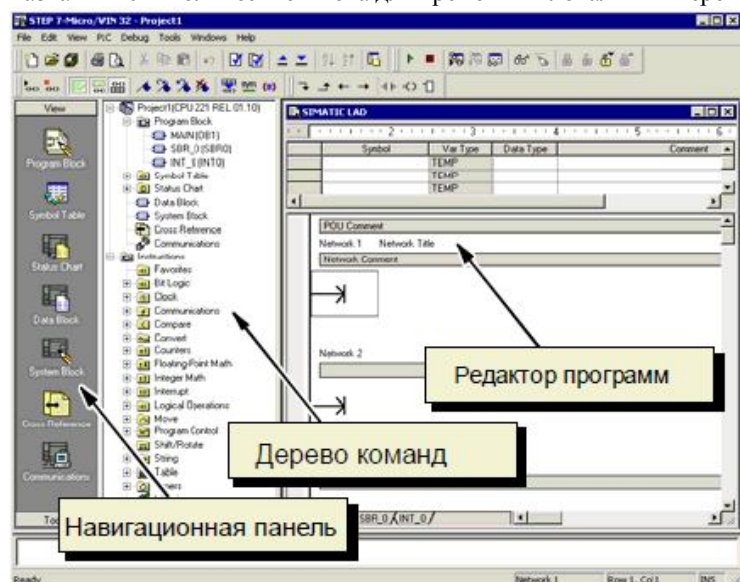


Рис. 3 Рабочая область для создания программы управления

Настройка панели инструментов в STEP 7-Micro/WIN 32.

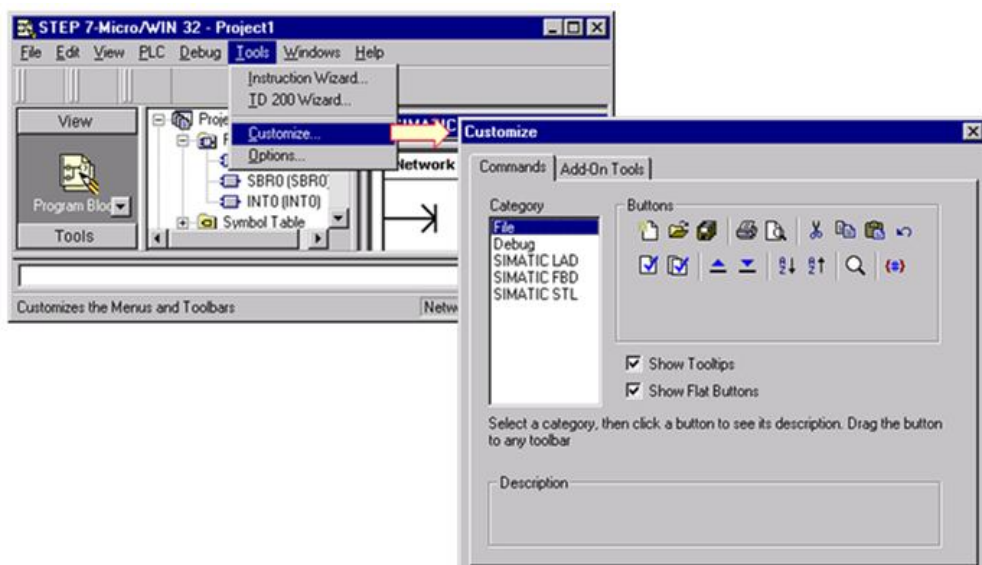


Рис. 4 Настройка панели инструментов в STEP 7-Micro/WIN 32.

Изменение вида: Активируйте бокс выбора Show Tooltips (Показать советы по инструментам), если Вы хотите увидеть краткое объяснение, когда Вы держите указатель мыши над кнопкой. Активизируйте бокс выбора Show Flat Buttons (показывать плоские кнопки), если Вы хотите, чтобы кнопки выглядели плоскими, а не трехмерными.

Перемещение кнопок: Выберите в списке Category (категории) toolbar (панель инструментов), чтобы показать кнопки панели инструментов. Для того, чтобы переместить кнопку из панели, где она находится по умолчанию, в другую панель, выберите в списке категорий имя панели, содержащей кнопку к настоящему времени. Перетаскивайте кнопку из области "Buttons (кнопки)" на желаемую панель рабочей области STEP 7-Micro/WIN 32, чтобы добавить ее к этой панели. Чтобы удалить кнопку из панели инструментов STEP 7-Micro/WIN 32, тащите кнопку из панели STEP 7-Micro/WIN 32 в область "Buttons (кнопки)" диалогового окна Customize.

Структура проекта

Проект создает пять собственных компонентов:

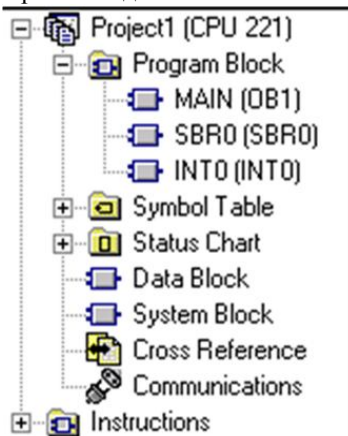


Рис. 5 Компоненты проекта

Загрузка компонентов проекта в CPU и выгрузка из CPU

Для загрузки проекта в CPU S7-200 нужно:

- Выбрать команду меню File >Download.
- Щелкнуть на элементе проекта, который необходимо загрузить.
- Щелкнуть на кнопке Download

Установка режима работы CPU S7-200

S7-200 имеет два режима работы: STOP и RUN. В состоянии STOP S7-200 не выполняет программы, и можно загрузить в CPU программу или конфигурацию CPU. В режиме RUN S7-200 исполняет программу. Для изменения режима работы S7-200 снабжен переключателем режимов. С помощью переключателя режимов можно установить режим работы вручную:

- установка переключателя режимов в STOP прекращает исполнение программы;
- установка переключателя режимов в RUN запускает исполнение программы;
- установка переключателя режимов в режим TERM (терминал) не изменяет режима работы.

Если питание прерывается, когда переключатель режимов находится в положении STOP или TERM, S7-200 при восстановлении питания автоматически переходит в состояние STOP. Если питание прерывается, когда переключатель режимов находится в положении RUN, S7-200 при восстановлении питания переходит в режим RUN.

STEP 7-Micro/WIN в режиме **online** дает возможность изменить режим работы S7-200. Чтобы это программное обеспечение могло управлять режимом работы, нужно вручную перевести переключатель режимов работы на S7-200 в положение TERM или RUN. Для изменения режима работы можно использовать команды меню PLC > STOP или PLC > RUN или соответствующие кнопки на панели инструментов.

Для перевода S7-200 в состояние STOP можно использовать в программе команду STOP. Это позволяет прекратить исполнение программы в зависимости от логики обработки.

Программный блок Программный блок содержит исполняемый код и комментарии. Исполняемый код состоит из основной программы (OB1) и некоторых подпрограмм или программ прерываний. Код компилируется и загружается в PLC. Комментарии не компилируются и не загружаются.

Блок данных DB. Блок данных содержит данные (начальные значения переменных, значения констант) и комментарии. Данные компилируются и загружаются в PLC. Комментарии не компилируются и не загружаются.

Системный блок System Block. Системный блок содержит данные конфигурации, такие как параметры коммуникации, области сохраняемых данных, аналоговые и цифровые входные фильтры, значения выходов в случае перехода в STOP (и информацию о пароле). Системный блок загружается в PLC.

Таблица символов "Symbol Name". Таблица символов позволяет Вам использовать символическую адресацию. Символика часто делает программирование более простым и облегчают чтение программ. Скомпилированная программа, которая загружается в PLC преобразует все символы в абсолютные адреса. Информация таблицы символов не загружается в PLC.

Диаграмма состояний Status Chart. Информация диаграммы состояний не загружается в PLC. На диаграмме состояния можно ввести адреса программы для мониторинга и модификации. Величины таймеров или счетчиков могут быть отображены, как биты или слова. Если выбирается битовый формат, то отображается состояние выхода (ON или OFF). Если выбирается формат слова, то отображается текущая величина таймера или счетчика.

Использование таблицы символов для символической адресации переменных

Таблица символов используется для присвоения символических имён входам, выходам и адресам.

Символическое имя:

- Максимум 23 символа
- Большая, маленькая буква имеет смысл
- Пробел заменяется знаком подчёркивания.
- Повторяющиеся символьные имена подчёркиваются, не компилируются и не могут быть использованы в программе.



Таблица символов дает возможность определять и редактировать символы, к которым можно обращаться во всей программе через символические имена. Таблица символов называется также таблицей глобальных переменных.

Можно указывать операнды команд в программе абсолютно или символически. При абсолютной адресации задается область памяти, а также бит или байт адреса. При символической адресации для указания адреса используются комбинации алфавитно-цифровых символов.

Для присвоения адресу символического имени необходимо:

1. Щелкнуть в навигационной панели на кнопке таблицы символов, чтобы вызвать таблицу.
2. Ввести символическое имя (например, Pump1Limit) в столбце "Symbol Name". Максимальная длина символического имени составляет 23 символа.
3. В столбце Address ввести адрес (например, I1.1).

		2	3	4	5
		Symbol	Address	Comment	
1		AlwaysOn	SM0.0	Always on contact	
2		Pump1	Q2.3	Pump 1 on/off	
3		Pump1Limit	I1.1	Pump 1 pressure limit switch	
4		Pump1Pressure	VD100	Pump 1 current pressure (real)	
5		Pump1Rpm	Vw200	Pump1 PRMs (integer)	
6					

Рис. 6 Таблица символов

Открыть таблицу символов можно с помощью щелчка правой кнопкой "мыши" на символе модуля. После этого во всплывающем окне выбирается пункт **Edit Symbolic Names** (Редактирование символьных имен). После этого открывается таблица символов с соответствующими адресами.

Для программы создается только одна таблица символов, независимо от того, какой язык программирования выбран. Нельзя использовать одну и ту же строку более одного раза в качестве глобального символического имени ни в единственной таблице, ни в нескольких различных таблицах!

Использование таблицы состояний (Status Chart)

С помощью таблицы состояний (**Status Chart**) можно наблюдать и изменять переменные процесса, когда S7-200 выполняет программу управления. Можно отслеживать состояние входов, выходов или переменных программы, отображая их текущие значения. В таблице состояний можно также принудительно задавать или изменять значения переменных процесса.

Для вызова таблицы состояний необходимо выбрать команду меню **View > Component > Status Chart**



или щелкнуть на пиктограмме таблицы состояний на навигационной панели.

	Address	Format	Current Value	New Value
1	Pump1	Bit	2#0	
2	Pump1Limit	Bit	2#0	
3	Pump1Pressure	Signed	+0	
4	Pump1Rpm	Signed	+0	
5	M3.7	Bit	2#0	
6	VD100	Hexadecimal	10#00	
7	VD200	Floating Point	0.0	
8		Signed		

Рис. 7 Таблица состояний


Для создания таблицы состояний и контроля переменных:

1. Введите в поле адресов адреса желаемых величин.
2. В столбце Format выберите тип данных.
3. Для отображения состояния переменных процесса в своем S7-200 выберите команду меню **Debug > Chart Status**.
4. Если вы хотите опрашивать эти величины непрерывно или хотите однократно считать состояние, щелкните на соответствующем символе на панели инструментов.

В таблице состояний можно также принудительно устанавливать или изменять значения различных переменных процесса. В таблицу состояний можно вставлять дополнительные строки, выбрав команду меню **Edit > Insert > Row**.

Невозможно отобразить состояния констант, аккумуляторов и локальных переменных. Значения таймеров и счетчиков можно отображать в виде бита или слова. Если значение отображается в виде бита, то оно представляет состояние бита таймера или счетчика; если значение отображается в виде слова, то оно является значением таймера или счетчика.

Создание проекта

Для создания нового проекта выберите команду меню **File -> New** или нажмите в панели инструментов на кнопку  (Рис.8).

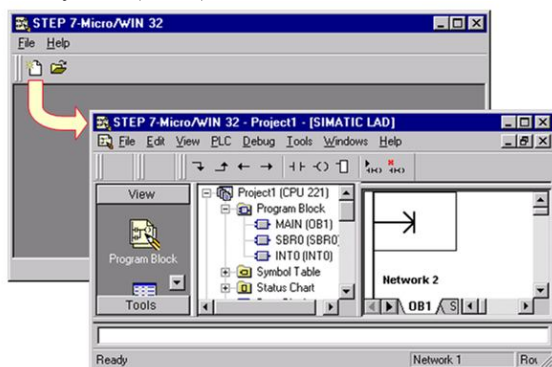


Рис. 8 Создание нового проекта

Структура создаваемой программы

Программа состоит из логических блоков и блоков данных. Логические блоки - это блоки, содержащие кодовую часть, например, организационные блоки, функциональные блоки и функции.

Данные

Операционная система делает доступными следующие данные:

- Периферийные входы и выходы
- Образ процесса на входах и выходах
- Меркеры
- Таймеры
- Счетчики

Основные элементы программы

Блок программы составлен из выполняемого кода и комментариев. Исполняемый код состоит из

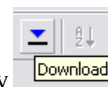
- основной программы (OB1),
- подпрограмм (Subroutine)
- программ обработки прерываний (Interrupt).

Код компилируется и загружается в S7-200. С помощью этих организационных элементов (основной программы, подпрограмм и программ обработки прерываний) можно структурировать свою программу.

Комментарий. Существует 2 вида комментариев, которые можно добавить в программу. Короткий комментарий содержит до 36 символов. Длинный комментарий вводится 2-мя кавычками. Комментарии не компилируются и не загружаются.

Подпрограммы и программы обработки прерываний появляются как закладки в нижней части окна редактора программ. Для перемещения между подпрограммами, программами обработки прерываний и основной программой необходимо щелкать по этим закладкам.

Загрузка программы



Вы выбираете пункт меню: **PLC ->Download** или соответствующую кнопку на панели инструментов.



Для загрузки выбранной конфигурации в PLC. PLC должен быть в режиме "STOP"!

Редакторы для создания программ

STEP 7-Micro/WIN имеет в своем распоряжении три редактора для создания программ:

- контактный план (LAD)
- список операторов (STL)
- функциональный план функциональная блок-схема (FBD).

LAD - это графический язык, здесь в качестве команд используются коммутационная схема, которая очень похожа на электротехническую схему. Данный язык легко позволяет проследить идущий сигнал между токовыми шинами, входами, выходами и командами.

FBD - это графический язык, использующий логические блоки, известные из булевой алгебры для представления логических операций.

STL - это текстовый язык программирования. Его операторы очень похожи на язык ассемблера, за которыми следуют адреса (операнды).

Для каждого созданного блока, можно выбирать, какой язык программирования использовать. С некоторыми ограничениями, программы, написанные в любом из этих редакторов программ, могут отображаться и редактироваться с помощью других редакторов программ.

Чтобы переключиться между FBD/LAD/STL, из меню нужно выбрать соответствующий редактор (Рис. 9). FBD и LAD всегда можно переключить в представление STL. В случае переключения из LAD в FBD или наоборот, операторы, которые не могут быть представлены на данном языке, будут отображаться на языке STL.

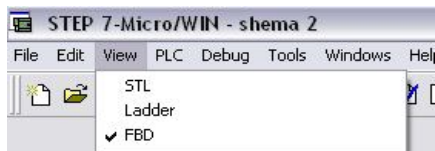


Рис. 9 Выбор соответствующего редактора

Язык программирования LDR (LAD) основан на изображении контактно-релейных схем.

- Все нагрузки, лампы и другие выходы изображаются справа.
- Входной сигнал может управлять несколькими выходами. В этом случае выходы изображаются параллельно
- Ключи, контакты, датчики и другие управляющие элементы изображаются в LDR-диаграммах слева.
- Ключи, контакты, датчики и другие управляющие элементы могут включаться и изображаться последовательно, параллельно, последовательно- параллельно.
- Каждая строчка LDR-диаграммы нумеруется и считывается контроллером сверху вниз.
- Каждому элементу LDR-диаграммы присваивается один, отличный от других идентификационный номер.

Отношение между дигитальными блоками и контактными схемами. Создание LDR диаграмм из контактно-релейных схем.

Элементы и блоки

Команды LAD состоят из элементов и блоков, графически объединяемых в сегменты. Элементы и блоки можно разделить на следующие группы:

Команды как элементы - эти команды **LAD** представляются в виде отдельных элементов, которым не нужны ни адреса, ни параметры (Рис.10).

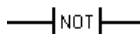


Рис.10 Элемент

Команды как элементы с адресом - эти команды **LAD** представляются как отдельные элементы, для которых нужно вводить адрес (Рис.11).

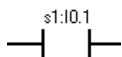


Рис.11 Элемент с адресом

Команды как элементы с адресом и значением - эти команды контактного плана представлены как отдельные элементы, для которых нужно вводить адрес и значение (Рис.12).

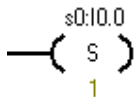


Рис. 12 Элемент с адресом и значением

Команды в виде блоков с параметрами – эти команды в виде блоков с линиями входов и выходов (Рис.13). Входы находятся с левой стороны блока; выходы – с правой стороны блока. Заполняются входные параметры. Для выходных параметров указываются места, куда программное обеспечение STEP 7 может поместить выходную информацию.

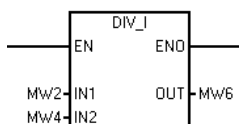


Рис. 13 Команды в виде блоков с параметрами

Передача энергии на разрешающий вход (EN) блока **LAD** (активизация) приводит к тому, что блок выполняет заданную функцию. Если блок способен выполнить свою функцию без ошибок, то разрешающий выход (ENO) передает энергию по цепи.



Ограничения для блоков и катушек

Вы не можете размещать блоки и коннекторы в цепи **LAD**, которая не начинается на левой питающей шине. Исключением являются операции сравнения.

FBD - это графический язык программирования, использующий для представления логических операций логические блоки, известные в булевой алгебре. Сложные функции (например, математические) тоже могут быть представлены непосредственно соединенными с логическими блоками.

Команды **FBD** состоят из элементов и блоков, графически объединяемых в сегменты. Элементы и блоки можно разделить на следующие группы:

Команды как элементы

Часть команд **FBD** представляется в виде отдельных элементов, которые не нуждаются ни в адресах, ни в параметрах (Рис. 14).



Рис. 14 Отрицание двоичного ввода

Команда как блок с адресом

Некоторые из команд **FBD** представляются в виде блоков, для которых необходимо указать адрес (Рис. 15).

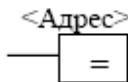


Рис. 15 Команда Присвоить

Команда как блок с адресом и значением

Некоторые из команд **FBD** представляются в виде блоков, для которых нужно указать адрес и значение (например, значение таймера или счетчика).

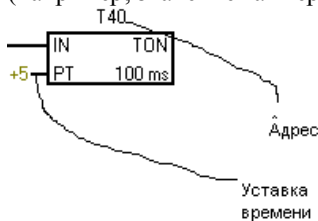


Рис. 16 Таймер с адресом и значением времени

Команда как блок с параметрами

Некоторые из команд **FBD** представляются в виде блоков с входами и выходами (Рис. 17). Входы расположены слева от блока, а выходы справа. Указываются входные параметры и некоторые из выходных параметров. Для назначения параметров необходимо использовать специальную запись типов данных. Параметры EN (деблокировать вход) и ENO (деблокировать выход) описаны ниже.

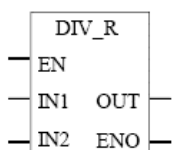


Рис. 17 Блок с параметрами: деление вещественных чисел

Параметры EN и ENO

Если параметр EN (деблокировать вход) блока активизирован, то блок выполняет определенную функцию. Если эта функция выполняется блоком без ошибок, то активизируется параметр ENO (деблокировать выход). Параметры EN и ENO относятся к типу данных BOOL.

EN и ENO действуют так:

- Если EN не активизирован (его сигнальное состояние равно 0), то блок не выполняет свою функцию и ENO не активизируется (его сигнальное состояние тоже равно 0).
- Если EN активизирован (его сигнальное состояние равно 1) и если блок выполняет свою функцию без ошибок, то ENO тоже активизируется (его сигнальное состояние тоже равно 1).
- Если EN активизирован (его сигнальное состояние равно 1) и если при исполнении блоком функции возникает ошибка, то ENO не активизируется (его сигнальное состояние остается равным 0).

Редактирование

После ввода элемента редактор проводит синтаксический контроль и показывает, были ли вводы неверными (ошибки отображаются красным цветом). Неправильно расположенные элементы отвергаются с сообщением об ошибке.

В разделе операторов можно редактировать заголовок блока, названия сегментов, комментарии к блоку, комментарии к сегментам и, конечно, команды внутри сегментов.



Программа STEP7 обладает интерфейсом Windows. Копирование объектов и прочие элементарные операции выполняются также, как в Windows.

Запуск SIMATIC Manager и создание проекта

При запуске STEP7 по умолчанию запускается мастер который поможет создать проект **step7**. Структура проекта используется для надлежащего хранения и размещения всех данных и программ.

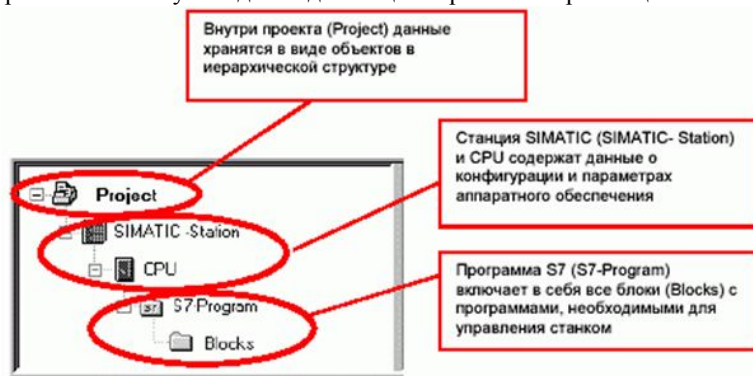


Рис. 18 Структура проекта STEP7

Каждый CPU обладает определенными свойствами относительно конфигурации его памяти или адресных областей. Вот почему вы должны выбрать CPU, прежде чем начать программирование. Адрес MPI (многоточечный интерфейс) нужен, чтобы CPU мог обмениваться информацией с вашим устройством программирования или PC. Установка по умолчанию для адреса MPI равна 2.

Структура проекта в SIMATIC Manager

Как только мастер STEP 7 закрывается, появляется SIMATIC Manager с открытым окном проекта "Getting Started". Отсюда вы можете запускать все функции и **окна STEP 7**.



Рис.19. Структура проекта в SIMATIC Manager

Папка SIMATIC 300Station содержит все данные проекта относящиеся к аппаратуре. Компонент Hardware [Аппаратура] используется для указания параметров программируемого контроллера.

Папка **Program1** содержит все необходимые компоненты. Компонент **Symbols (Символы)** используется для того чтобы дать адресам символические имена. Компонент **Source Files** [Исходные файлы] используется для хранения программ в виде исходных файлов.



Рис. 20 Папка **Program1**

Папка Blocks [Блоки] содержит OB1 и другие блоки которые потом создаются. Отсюда можно запускать программирование в LAD, STL, FBD.

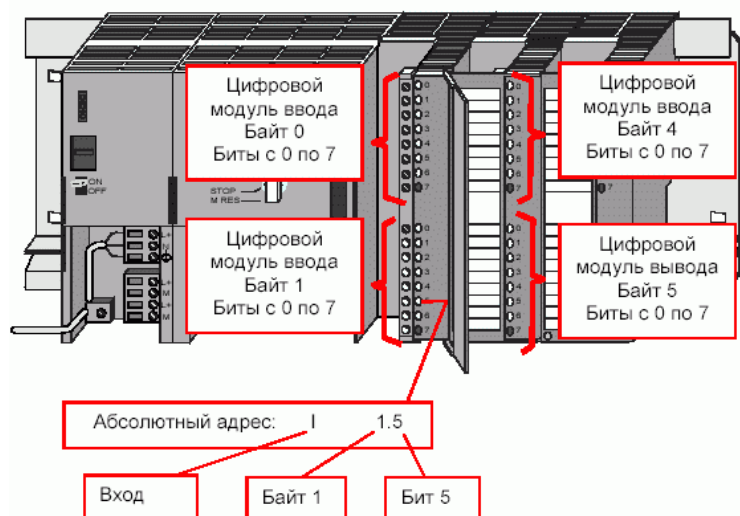


Рис.21 Адресация цифровых модулей

Обращение к данным в областях памяти. Адреса цифровых модулей

Для обращения к биту в некоторой области памяти необходимо указать адрес бита. Этот адрес состоит из идентификатора области памяти, адреса байта и номера бита. На рис.21 и 22 показан пример обращения к биту (адресация в формате «байт.бит»). В этом примере за областью памяти и адресом байта (I = input [вход], I= байт следует точка (.), чтобы отделить адрес бита (бит 2).



Рис. 22 Адрес входа или выхода цифрового модуля складывается из адреса байта и адреса бита



Точка используется, чтобы отделить адрес бита.

Адрес байта связан с начальным адресом модуля. Адрес бита считывается на модуле.

Регистр входов образа процесса: I

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

Бит: I[адрес байта].[адрес бита] **I0.1**

Байт, слово или двойное слово: I[длина][начальный адрес байта] **IB4**

Регистр 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:

Бит: Q[адрес байта].[адрес бита] **Q1.1**

Байт, слово или двойное слово: Q[длина][начальный адрес байта] **QB5**

Основы выполнения программы, различные виды используемой памяти

Исполнение программы

На этом участке цикла контроллер обрабатывает программу с первой команды до последней. Можно непосредственно управлять входами и выходами и получать, таким образом, доступ к ним во время исполнения основной программы или программы обработки прерываний.

Самодиагностика CPU

На этом участке цикла контроллер проверяет надлежащую работу CPU, области памяти и состояние модулей расширения.

Чтение входов

В начале цикла текущие значения цифровых входов считываются, а затем записываются в регистр входов образа процесса.

Запись в цифровые выходы

В конце каждого цикла контроллер записывает значения, хранящиеся в регистре выходов образа процесса, в цифровые выходы. (Аналоговые выходы обновляются немедленно, независимо от цикла.)

Управление входами.

Двоичные сигналы от нескольких входов логически объединяются центральном блоке в соответствии с командами программы.

Логические блоки.

Обозначения всех логических блоков AND, OR, NOT, NOR, NAND, Exclusive OR описываются стандартом. Рассмотрим их операции и внутреннюю электронику.



Все блоки имеют один выход.

На выходе блоков или 0 или 1 в зависимости от логических сигналов на входе. Открытое состояние блоков – «1» на выходе, когда +5V DC поступает с источника питания. На выходе «0», когда блок в закрытом состоянии.



NOT блок имеет один вход, Exclusive OR и Exclusive NOR - только два входа. Все другие блоки имеют до 8 входов, а иногда и больше. Вход включается, когда +5V DC подаётся, выключается, когда 0.

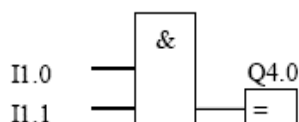
Булева логика и таблицы истинности

В двоичной логике булевой алгебры переменные могут принимать значения “истина” (1) или “ложь” (0). Каждая логическая команда проверяет состояние сигнала переменной на равенство 1 (истина, удовлетворяется) или 0 (ложь, не удовлетворяется) и генерирует результат. Затем команда или сохраняет результат, или использует его для выполнения булевой логической операции. Результат логической операции называется RLO. Для представления логики используются логические блоки, известные из булевой алгебры. Результаты логических операций для всех возможных комбинаций логических переменных перечисляются в таблицах истинности.

Правила булевой логики иллюстрируются ниже на примере логических операций **И**, **ИЛИ** и **исключающее ИЛИ**.

Логическая операция И

В логической операции И опрашиваются сигнальные состояния двух или более указанных адресов. Если на всех входах сигнал равен 1 то на выходе будет 1. Если хотя бы на одном входе будет ноль, то на выходе тоже будет ноль.



Условие удовлетворяется, когда сигнальное состояние равно 1 на входах I1.0 **И** I1.1.

Возможные результаты логической операции И могут быть представлены в таблице истинности (табл.1). Здесь 1 означает “удовлетворяется”, а 0 означает “не удовлетворяется”.

Таблица 1

Результат опроса сигнального состояния по адресу I1.0 равен	Результат опроса сигнального состояния по адресу I1.1 равен	Результат логической операции имеет следующее значение:
1	1	1
0	1	0
1	0	0
0	0	0

Логическая операция ИЛИ

В логической операции ИЛИ опрашиваются сигнальные состояния двух или более указанных адресов. Если хотя бы на 1 входе сигнал равен 1 то и на выходе будет единица.



Условие удовлетворяется, когда равно 1 сигнальное состояние на входе I1.0 **ИЛИ** I1.1.

Результаты логической операции ИЛИ представлены в таблице истинности (табл.2).

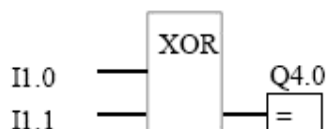
Таблица 2

Результат опроса сигнального состояния по адресу I1.0 равен	Результат опроса сигнального состояния по адресу I1.1 равен	Результат логической операции имеет следующее значение:
1	1	1

0	1	1
1	0	1
0	0	0

Логическая операция исключающее ИЛИ

В логической операции исключающее ИЛИ опрашиваются сигнальные состояния двух или более указанных адресов. Если сигнальное состояние одного из адресов равно 1, то условие удовлетворяется, и команда дает результат 1. Если сигнальные состояния всех адресов равны 0 или 1, то условие не удовлетворяется, и операция результат равен 0 (табл.3).



Условие удовлетворяется, когда сигнальное состояние равно 1 только на входе I1.0 или I1.1 (т.е. не на обоих одновременно).

Результаты логической операции исключающее ИЛИ представлены в таблице 3.

Таблица 3

Результат опроса сигнального состояния по адресу I1.0 равен	Результат опроса сигнального состояния по адресу I1.1 равен	Результат логической операции имеет следующее значение:
1	0	1
0	1	1
1	1	0
0	0	0

Упражнение. Проект в STEP 7-Micro/WIN 32 (управление входами-выходами).

Упражнение: Создание проекта

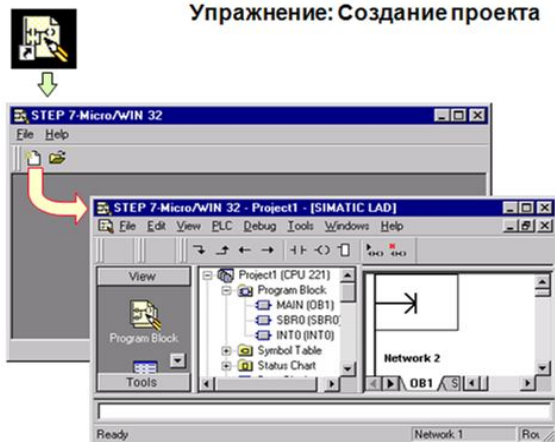


Рис. 19 Создание нового проекта

Как делать:

1. Запустите Micro/WIN.
2. Создайте новый проект выбором команды меню **File -> New**.
3. Вставьте новую программу в проект.



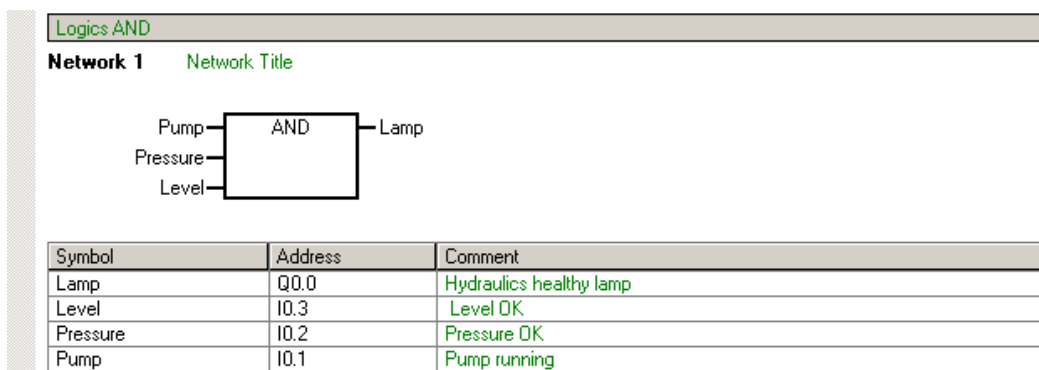


Рис. 20 Примеры применения AND функции в редакторах LAD и FBD

Символическая адресация

В редакторе можно вводить абсолютные адреса, параметры и имена блоков или использовать символы. С помощью команды меню **View > Symbolic Representation** можно переключаться между абсолютной и символической адресацией.

Для использования глобальных символов нужно внести их в таблицу символов следующим образом:


- Откройте таблицу символов командой меню **Options > Symbol Table**.
- Используя команду меню **Options > Edit Symbols**, откройте диалоговое окно, в котором можно определять и модифицировать отдельные символы.



Сохранить сделанные изменения в таблице.

Синдром двойной катушки

Синдром двойной катушки происходит, когда мы используем ту же самую выходную катушку больше, чем 1 раз в нашей программе.

 Рассмотрим пример. У нас есть два выключателя (toggle switches). Если один из них включают, то двигатель включится. Просто. Мы только создаем две линии диаграммы:

```

I0.0      Q0.1
--| |----- ( ) ---

```

```

I0.1      Q0.1
--| |----- ( ) ---

```

Так, если или вход I0.0 или I0.1 физически включается, то выход Q0.1 включится, и наш двигатель будет вращаться. Хорошо. Давайте идти дальше другому. Включаем сначала I0.0, затем I0.1. Проверяем. Ждем. Это будет работать? Правильно. Это не будет работать способом, которым мы спроектировали. Почему? Из-за ДВОЙНОГО СИНДРОМА КАТУШКИ.

Страшная программная болезнь появляется здесь, потому что мы использовали ту же самую катушку на выходе (Q0.1) дважды в той же самой программе на двух отдельных ветках (rungs).

После проверки ключей и двигателя выясняется, что двигатель хорош, и ключи работают очень хорошо. Так, назад к программе.

Причина болезни - способ, которым наш PLC просматривает программу. Помните, что он просматривает от слева направо, от начала до конца. Наблюдайте это.

```

I0.0      Q0.1
--|a |----- (b) ---

```

```

I0.1      Q0.1

```


--|c|------(d)---

Первоначально:

a и **c** физически выключены. Так, **b** и **d** также выключены.

Просмотр по шагам:

1 - Мы включаем ключ I0.0, так теперь **a** включен.

2 - Так как **a** включен, **b** (Q0.1) включится.

3 - **c** все еще физически выключен, таким образом **d** (Q0.1) также выключен.



Последняя вещь, которую мы сказали, была то, что Q0.1 будет выключено. Так, когда PLC сработает, чтобы обновить выходы, он оставит Q0.1 выключенным!!

Другая ситуация:

Первоначально, **a** и **c** физически выключены. Так, **b** и **d** также выключены. Теперь мы включаем ключ **I0.1**, таким образом **c** теперь включен.

Просмотр по шагам:

1 - **a** выключен, и **b** (Q0.1) также выключен.

2 - **c** включен, так и **d** (Q0.1) включится.



Последняя вещь, которую мы сказали, была то, что Q0.1 будет включен. Так, когда PLC обновляет выходы, он собирается включить Q0.1!!

Если Вы следовали за проверкой, Вы заметите это - независимо от того, ключ I0.0 включен или выключен, выход двигателя Q0.1, реагирует только на состояние (вкл\выкл) переключателя **I0.1**. Если **I0.1** включен, Q0.1 включен. Если **I0.1** выключен, Q0.1 выключен. Даже не имеет значения, что статус ключа I0.0 переключается. I0.0 в значительной степени проигнорирован!

Как мы "вылечиваем" эту болезнь и устраняем синдром?? Просто. Только использованием ИЛИ функции. В конце концов, мы хотим, чтобы двигатель включился, если ключи I0.0 *OR* I0.1 включаются. Вот решение:

I0.0 Q0.1
--| |------()---
 |
 I0.1 |
--| |--|

Теперь, если включаются I0.0 **ИЛИ** I0.1, то Q0.1 включится. Синдром Двойной Катушки вылечен.



В одно и то же время та же самая выходная катушка может благополучно использоваться в пределах подпрограмм и блоках перехода (jump, и т.д.).



Вопрос.

1. Как выгрузить проект из контроллера и сохранить в памяти компьютера?

2. Как сделать очистку памяти при изменении программы?

Необходимо, после выкачивания 3-х составляющих проекта, сделать очистку памяти CPU.

Очистка делается так:

1. Выбираем в меню PLC -> Clear PLC

2. Ставим галочки на очистку всех трёх блоков.

3. Жмём ОК.
4. Опять идёт запрос пароля. На этот запрос отвечаем "CLEARPLC"
5. Всё должно затереться.

После этого, вносим необходимые изменения в проект.

Затем заходим в System Block - > Password и решаем, ставить пароль на изменённый проект или нет.
Затем делаем Download, выделив все 3 блока для загрузки.