Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования
«Пермский государственный технический университет»
Кафедра электрификации и автоматизации горных предприятий

ИССЛЕДОВАНИЕ РАБОТЫ ПРОГРАММИРУЕМОГО ЛОГИЧЕСКОГО КОНТРОЛЛЕРА SIMATIC S7-313C

Методические указания к лабораторной работе по курсу
«Элементы систем автоматики»

Издательство
Пермского государственного технического университета
2008

Составители: ст. преподаватель *Е.В. Аристов*, ассистент *Р.А. Хузин*

УДК 680.51
И88

Рецензент
канд. техн. наук *А.В. Ромодин*
(Пермский государственный технический университет)

Исследование работы программируемого логического контроллера
И88 SIMATIC S7-313C: метод. указания к лаб. работе по курсу «Элементы систем автоматики» / сост. Е.В. Аристов, Р.А. Хузин. – Пермь: Изд-во Перм. гос. техн. ун-та, 2008. – 32 с.

Рассмотрены отличительные особенности S7-313C, подключение к контроллеру входных и выходных сигналов, а также принцип работы с программным пакетом Step7 и принципы программирования на языке LAD.

Предназначено для студентов специальности ЭАГП, изучающих курс «Элементы систем автоматики».

УДК 680.51

Подготовлено по программе «Гранты ТНК-ВР для профильных вузов РФ».

ИССЛЕДОВАНИЕ РАБОТЫ ПРОГРАММИРУЕМОГО ЛОГИЧЕСКОГО КОНТРОЛЛЕРА SIMATIC S7-313C

Введение

Цели работы:

1. Изучение устройства и способа подключения программируемого логического контроллера (ПЛК) Simatic S7-313C.
2. Изучение основ создания программ для ПЛК посредством программного пакета Step 7 на языке «контактный план» (КОР).

В настоящее время появилась возможность автоматизировать достаточно сложные процессы, автоматизация которых стала возможной с развитием микропроцессорной техники и персональных компьютеров. Широкий ассортимент оборудования для автоматизации промышленных производств предоставляет фирма SIEMENS, начиная от простых логических реле типа LOGO и заканчивая современными модульными промышленными компьютерами с операционными системами реального времени. Широчайшее распространение нашли программируемые логические контроллеры серии S7-300. Они представляют быстрые, высокопроизводительные и разнообразные компактные ПЛК. На нашем стенде установлен ПЛК S7-313C (з.н. 6ES7 313-5BE01-0AB0), предназначенный для решения относительно простых задач автоматического управления, в которых необходима скоростная обработка информации и малое время реакции системы. Основными отличительными особенностями S7-313C являются:

- Микропроцессор: 100нс на выполнение логической операции с битами.
- Загружаемая память в виде микрокарты памяти NVFlash-EEPROM емкостью до 8 Мбайт: сохранение программы и данных, опциональное сохранение архива полного проекта Step 7 с символьными таблицами и комментариями, опциональное сохранение архивов данных и рецептурных данных.
- Необслуживаемое сохранение резервной копии данных: при перебоях в питании в микро карту памяти записываются состояния флагов, таймеров, счетчиков и содержимое блоков данных.
- Встроенный MPI интерфейс: программирование/ диагностика/ обслуживание/ построение простейших сетевых структур, скорость передачи данных 187.5 Кбит/с. Объединение до 16 центральных процессоров SIMATIC S7/C7, поддержка механизма передачи глобальных данных.
- Встроенный переключатель режимов работы (RUN/ STOP/ MRES).
- Парольная защита: обеспечивает защиту программы от несанкционированного доступа.
- Диагностический буфер: в буфере сохраняется 100 последних сообщений об отказах и прерываниях. Содержимое буфера используется для анализа причин, вызвавших остановку центрального процессора.

• Часы реального времени: все диагностические сообщения могут снабжаться отметками даты и времени.

• Встроенные коммуникационные функции: PG/OP функции связи, стандартные и расширенные (клиент и сервер) *S7* – функции связи.

• Работа без буферной батареи.

• 24 встроенных дискретных входа =24В. Все входы могут использоваться для приема сигналов аппаратных прерываний.

• 16 встроенных дискретных выходов =24В/0.5А.

• 4 аналоговых входа для измерения сигналов напряжения или силы тока, 1 аналоговый вход для измерения сопротивления или подключения датчика температуры Pt100.

• 2 аналоговых выхода.

• Гибкое расширение: система локального ввода-вывода, обслуживающая до 31 сигнального, функционального и коммуникационного модуля S7-300 (4-рядная конфигурация).

• Возможность построения ПИД-регуляторов с импульсными или аналоговыми выходными сигналами.

• Рабочая память: RAM емкостью 32 Кбайт/ 10 К-инструкций.

• До 8 логических соединений в промышленных сетях с программмируемыми контроллерами S7-300/ S7-400/ C7/ программаторами/ компьютерами/ панелями оператора. Одно статическое соединение зарезервировано для связи с программатором и панелью оператора.

• 12 из 24 встроенных дискретных входов могут использоваться встроенными технологическими функциями.

• 3 из 16 встроенных дискретных выходов могут использоваться в импульсном режиме (до 2.5 кГц).

• ПИД-регулирование с формированием импульсных или аналоговых выходных сигналов.

• Измеритель частоты: CPU 313C – три канала для измерения частот до 30 кГц, CPU 314C-2 – четыре канала для измерения частот до 60 кГц. Принцип действия: подсчет количества импульсов за опорный промежуток времени.

• Скоростной счет: CPU 313C – три счетчика (30 кГц) с входами для подключения 24В инкрементальных датчиков и встроенными компараторами.

• Аналоговые каналы ввода-вывода:
  - общее количество – до 253;
  - из них в системе локального ввода-вывода – до 248.

• Дискретные каналы ввода-вывода:
  - общее количество – до 1016;
  - из них в системе локального ввода-вывода – до 992.

Контроллер имеет модульную структуру с возможностью расширения различными функциональными модулями.

Подключение контроллера

Подключение входных и выходных сигналов к контроллеру поясняет рис. 1.

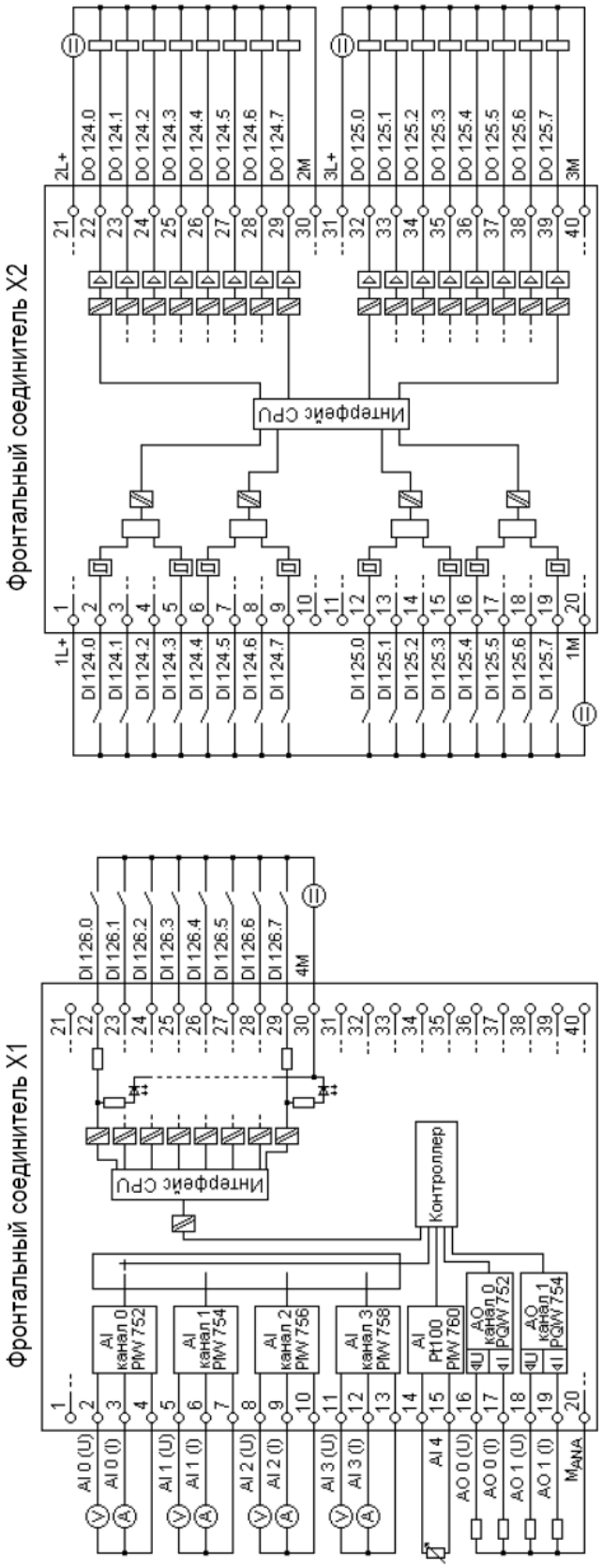


Рис. 1. Подключение цепей управления к контроллеру

Питание контроллера осуществляется от блока питания

стабилизированным напряжением 24 В.

Соединение контроллера с компьютером осуществляется посредством адаптера MPI(DP)/USB (з.н. A5E00166353-02).

Схема стенда и его внешний вид

Принципиальная схема стенда приведена на рис. 2. Внешний вид стенда приведен на рис. 3. Стенд позволяет имитировать технологический процесс

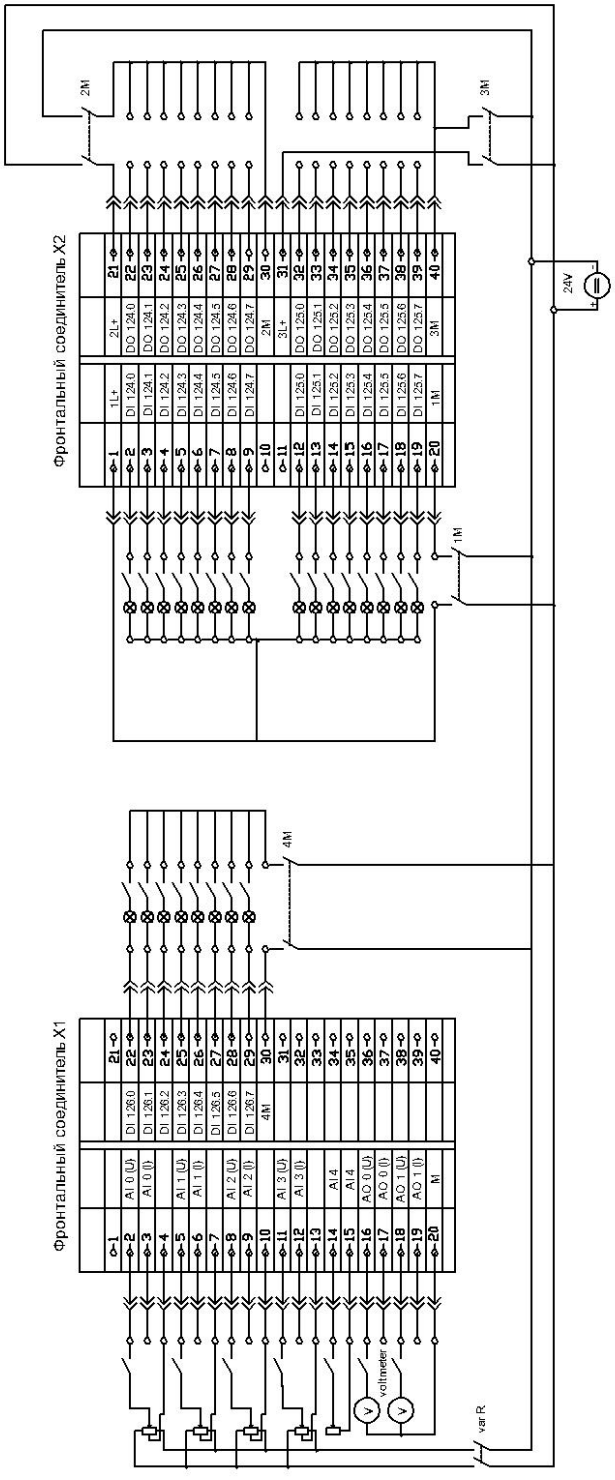


Рис. 2. Принципиальная схема стенда

путем создания входных сигналов для контроллера с помощью ключей и по-

тенциометров и регистрации выходных сигналов при помощи светодиодной индикации и цифровых вольтметров. Кроме того, стенд позволяет производить подключение внешних устройств (генераторов напряжений различной формы, исполнительной аппаратуры) посредством выведенных на лицевую панель разъемов. На стенде приведена абсолютная адресация каналов ввода-вывода.

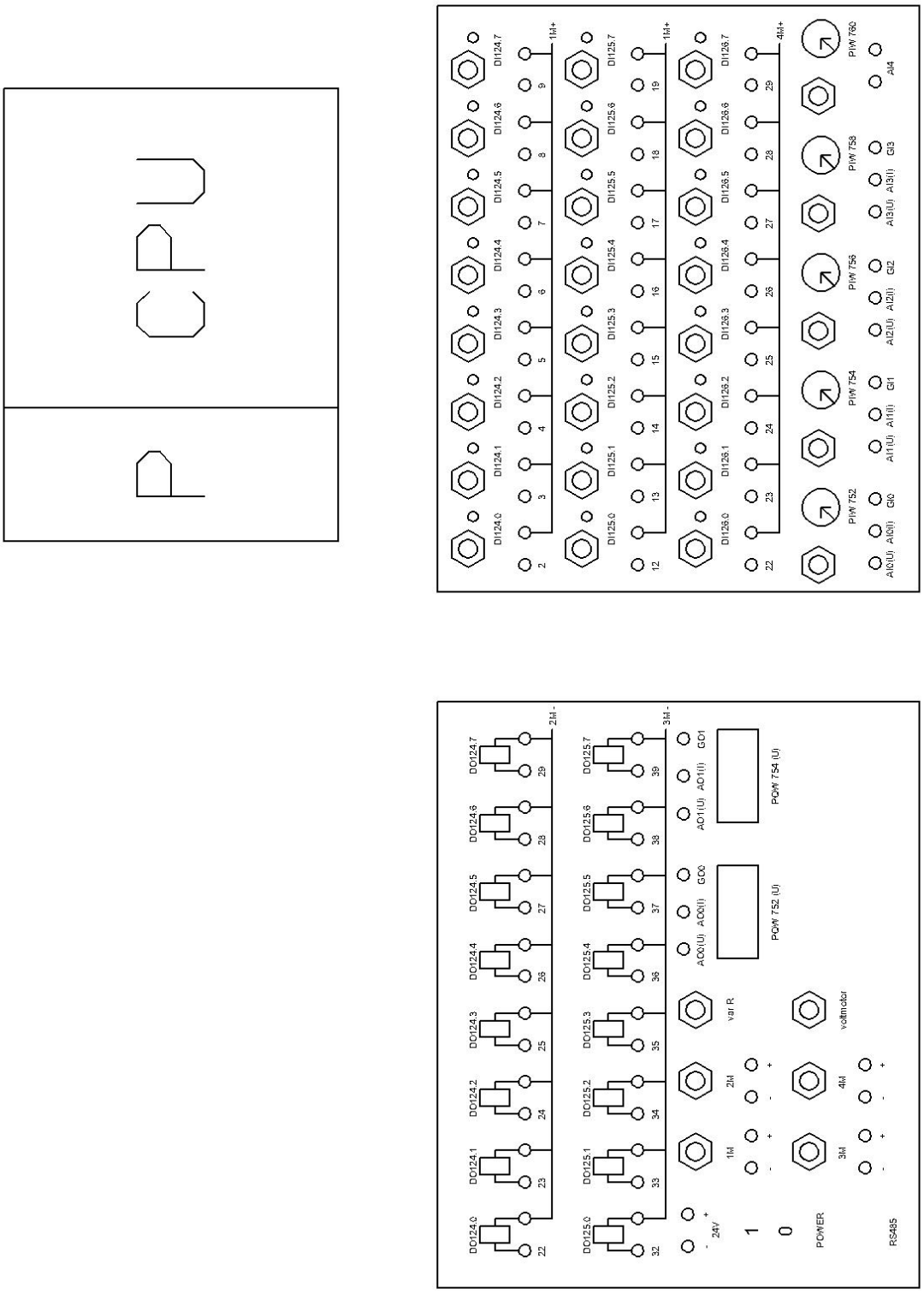


Рис. 3. Внешний вид стенда

Техника безопасности при выполнении лабораторной работы

Работа выполняется в присутствии преподавателя (лаборанта).

К работе допускаются студенты, прошедшие инструктаж по технике безопасности и ознакомившиеся с методическим пособием по проведению работ на стенде «Исследование работы программируемого логического контроллера SIMATIC S7-313C».

Выполнение лабораторной работы производится строго в соответствии с методическим указанием к работе. В процессе работы соединительные провода и клеммы могут находиться под опасным для жизни напряжением! Подключение внешних устройств к панелям стенда производится только преподавателем (лаборантом).

Запрещается извлечение карты памяти из контроллера.

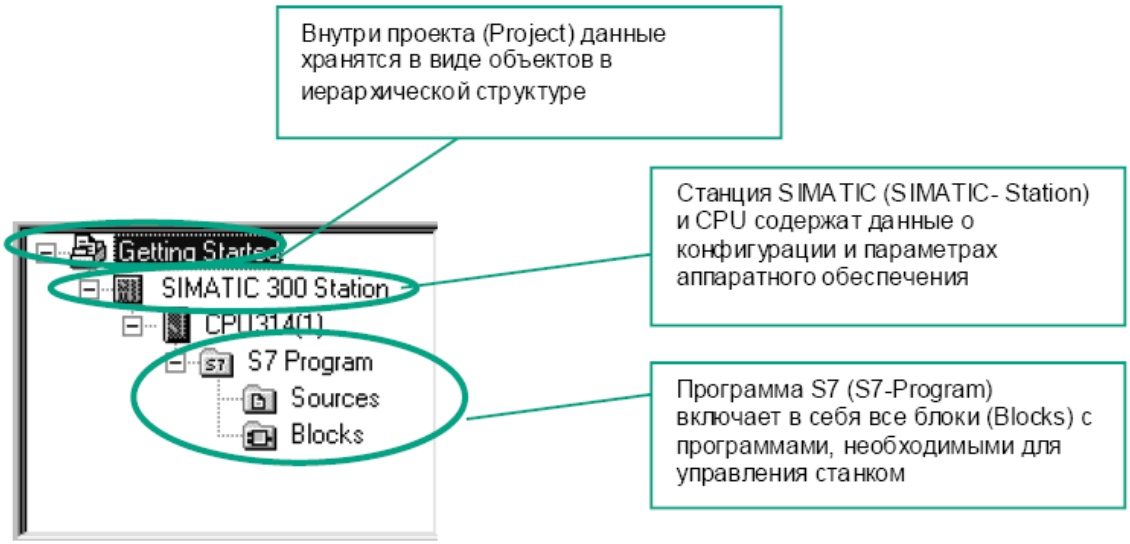
В случае наличия неисправностей выполнение работы прекратить и сообщить об этом преподавателю.

Программирование ПЛК

Данный раздел методического пособия написан на основе руководства «Step 7. Первые шаги» (Step 7 v.5.3. First steps. З.н. 6ES7810-4CA07-8BW0).

Программирование ПЛК осуществляется в программном пакете Step 7. На компьютере установлен пакет Step 7 v.5.2 Professional. Данный пакет позволяет писать программы на языках программирования: контактный план, функциональный план или список операторов для станций SIMATIC S7-300/400.

Для активации основного окна программы необходимо запустить программу «SIMATIC Manager». **SIMATIC Manager [Администратор SIMATIC]** – это центральное окно, которое становится активным при запуске Step 7. По умолчанию запускается мастер Step 7 Wizard, который оказывает помощь при создании проекта. Структура проекта используется для надлежащего хранения и размещения всех данных и программ.



После запуска программы вы попадете в окно, показанное на рис. 4.

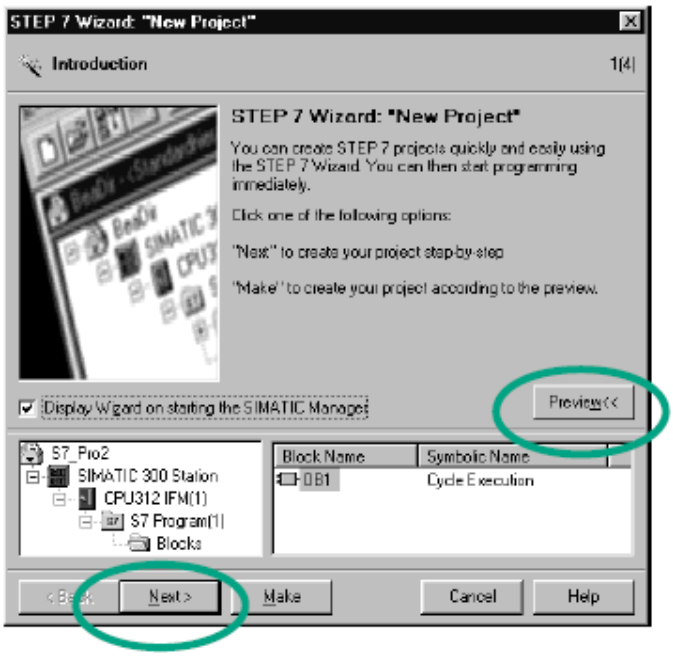
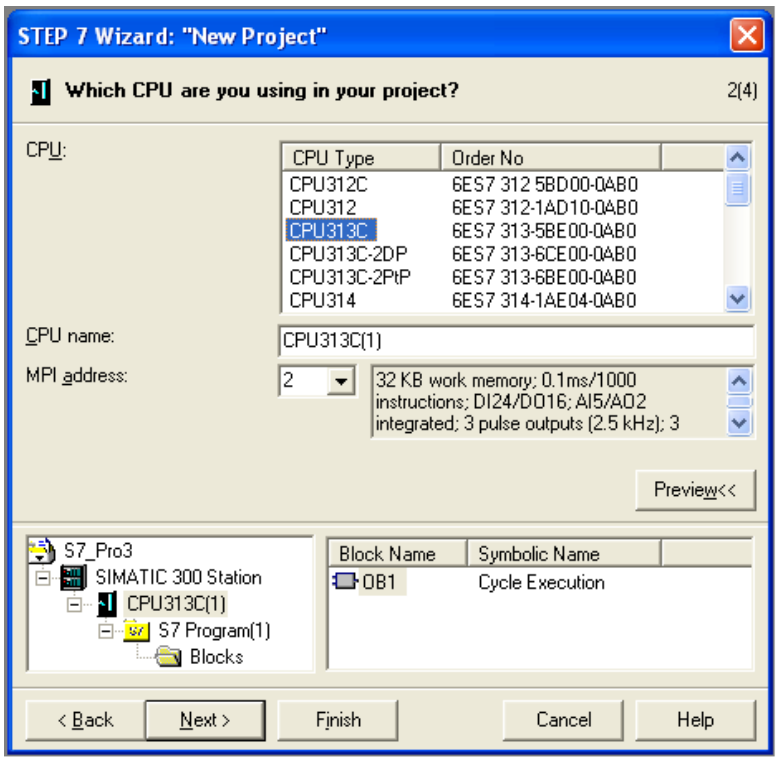


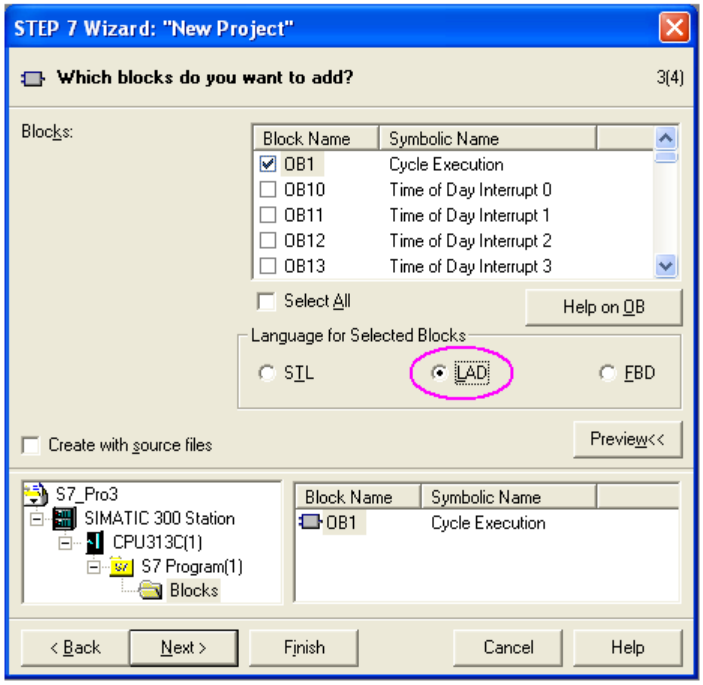
Рис. 4. *Step 7 Wizard*

Чтобы перейти к следующему диалоговому окну, щелкните на кнопке **Next [Дальше].**

На следующем этапе необходимо выбрать процессор (CPU), для которого создается программа. В нашем стенде установлен процессор CPU313C. Выберите его и нажмите **Next.**



Теперь необходимо выбрать язык, на котором будет создаваться программа, мы выберем **LAD** (от ladder – 'лестница' – язык контактный план). В данном программном обеспечении можно производить программирование еще на двух языках STL – расширенный ассемблер и FDB – функциональный план. Переключение между языками возможно в любое время, при этом программа будет переконвертирована в другой язык в большинстве случаев автоматически. Выбрав язык, снова нажмите **Next.**



На последнем этапе необходимо дать название своему проекту в поле **Project name** и нажать кнопку **Finish**. После этого Simatic Manager создаст проект, внешний вид которого показан на рис. 5.

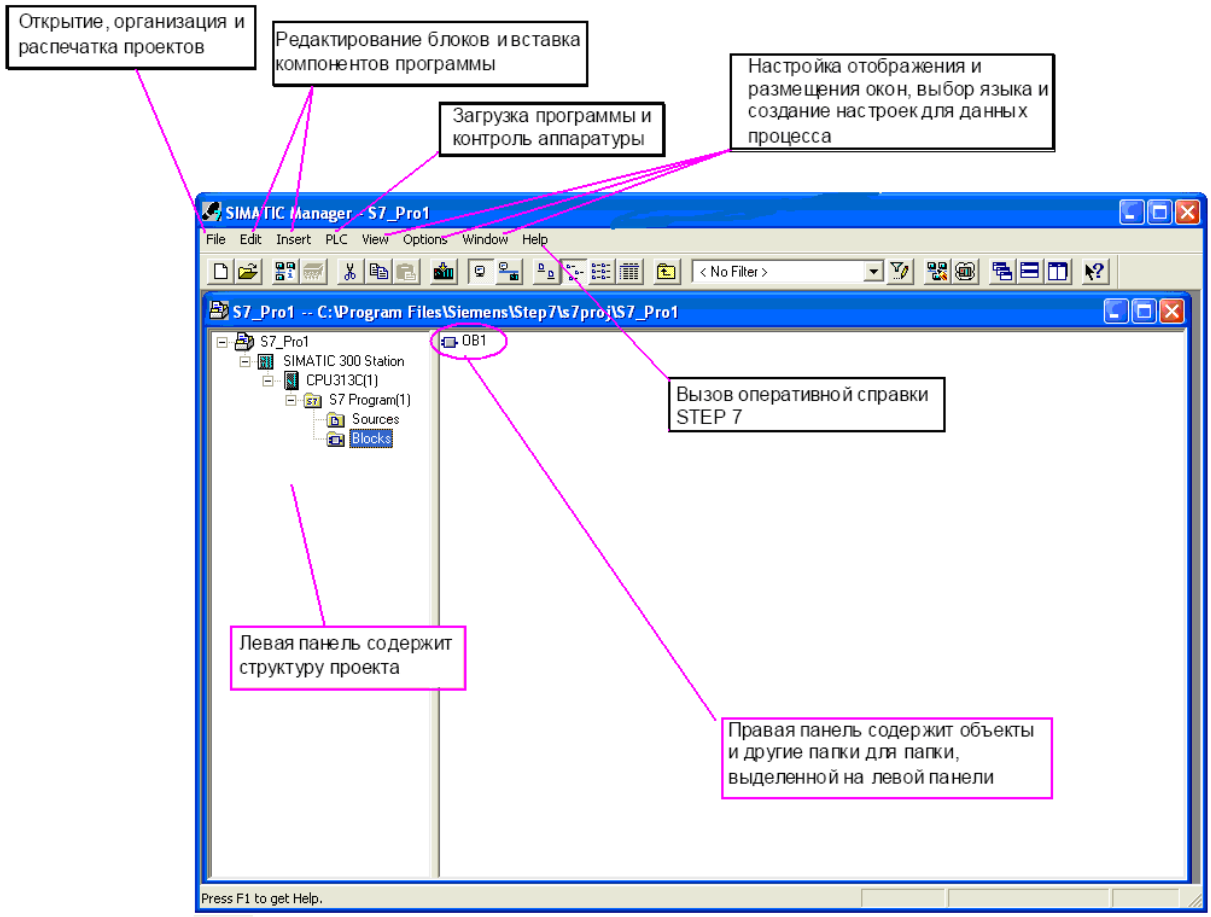


Рис. 5. Внешний вид проекта

В Simatic существует отличная система справочной информации, в т.ч. и контекстно-зависимая. Для получения справки необходимо выделить объект и нажать **F1**.

Теперь необходимо сказать об адресации входов-выходов, используемой в Simatic. Структура адресации поясняется на рис. 6.

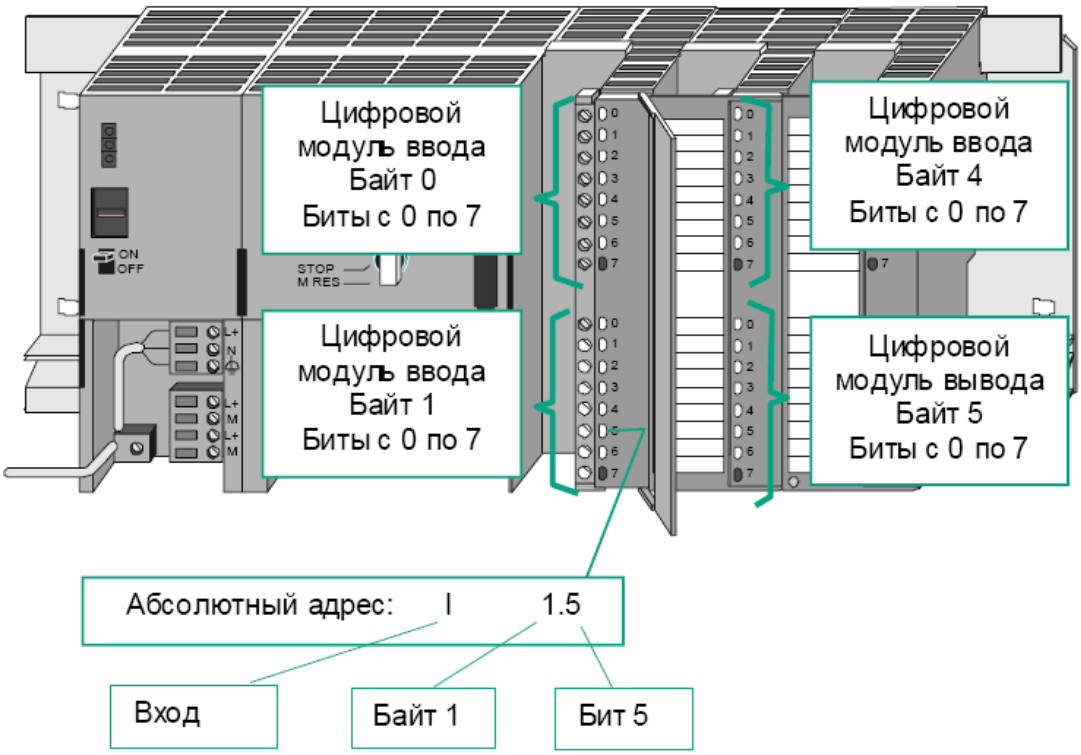


Рис. 6. Адресация в Simatic

Как видно, к группе входов можно обращаться как байту – обращение сразу к группе входов, также возможно обращение к отдельному входу – биту. Это относится к цифровым (дискретным) входам. Аналоговые входы представляются в виде 11-битного слова.

Для того чтобы посмотреть абсолютные адреса, которые допустимы для данного контроллера, необходимо нажать левую клавишу мыши в структуре проекта на раздел **SIMATIC 300 Station** и дважды нажать левую клавишу мыши в правой панели на папку **Hardware**. Перед вами откроется окно, представленное на рис. 7.

В этом окне можно производить конфигурирование стойки, т.е. производить выбор аппаратуры. Программа сама поможет верно поместить необходимое оборудование на стойку. Нас интересуют данные процессора, которые можно посмотреть в нижней части окна. Итак, у нашего процессора есть 24 цифровых входа и 16 цифровых выходов (DI24/DO16) с адресами 124.x–126.x (I address) и 124.x–125.x (Q address) соответственно, а также 5 аналоговых входов с адресами 752–761 и 2 выхода (752–755). Count – это адреса, используемые технологическими функциями для скоростного счета, измерения частоты, ШИМ. Данные технологические функции используют цифровые входы и выходы, а в указанных адресах содержатся результаты их работы. Если нажать левую клавишу мыши на соответствующие входы/выходы, можно произвести их конфигурирование, но нас устраивают существующие значения по умолчанию. Закройте окно конфигурирования аппаратуры и вернитесь обратно в главное окно программы.

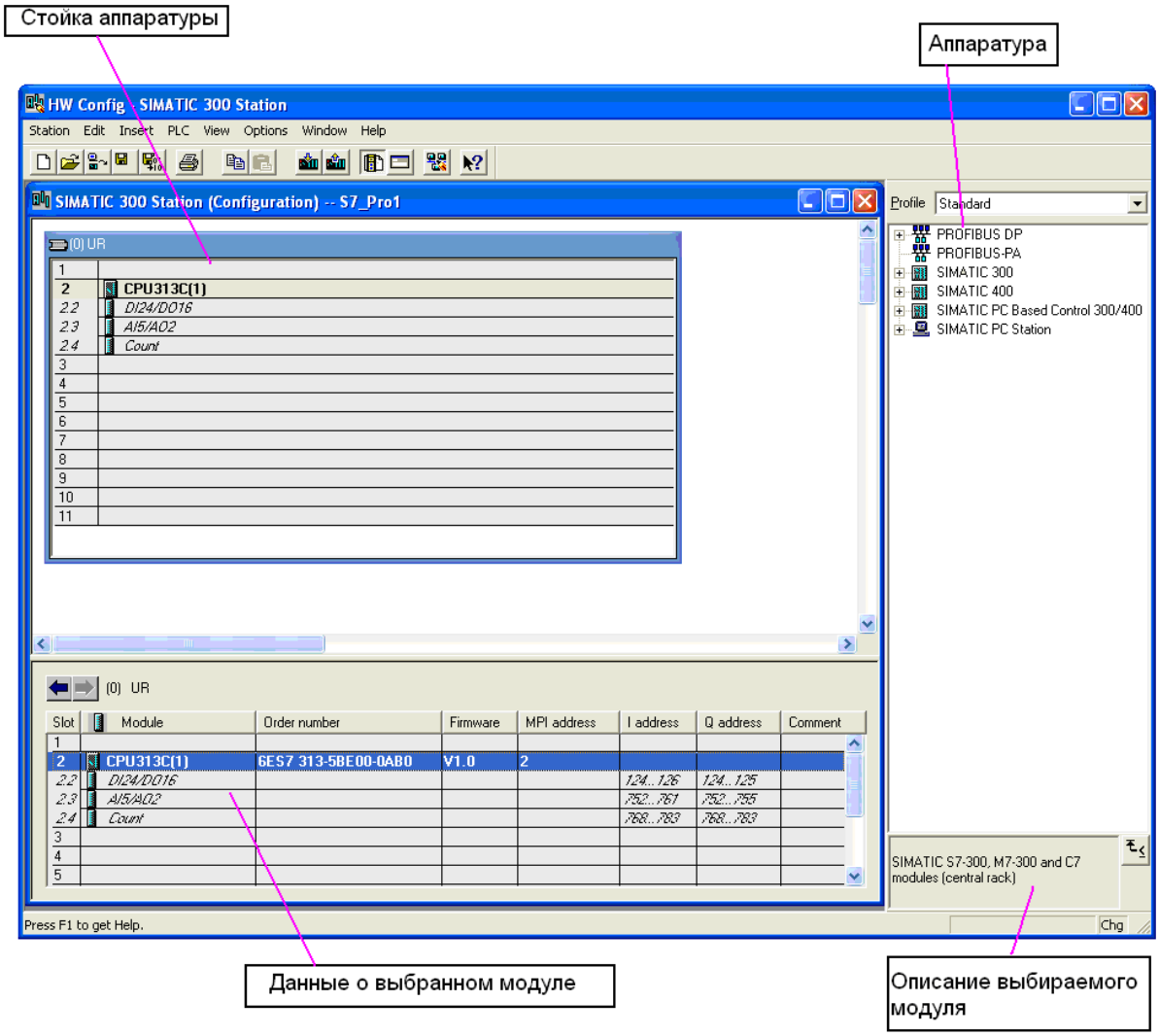


Рис. 7. Конфигурирование аппаратуры

При написании программы удобнее заменить абсолютные адреса символическими именами. Для присвоения символических имен необходимо активизировать таблицу символов для этого в древе проектов необходимо нажать левую клавишу мыши на **S7 Program** и в правой панели дважды нажать левую клавишу мыши по папке **Symbols** (рис. 8). После активации таблицы символов вы увидите окно, показанное на рис. 9.

В начале в таблице будет только одна переменная OB 1. Заполните таблицу символов согласно рисунку и сохраните ее. Посмотрите, как обозначаются цифровые входы и выходы, а также аналоговые. Следует отметить, что максимальный адрес для входа одной группы естественно будет XXX.7, а аналоговые входы и выходы имеют длину в два слова, т.е. первый аналоговый вход занимает адреса 752 и 753.

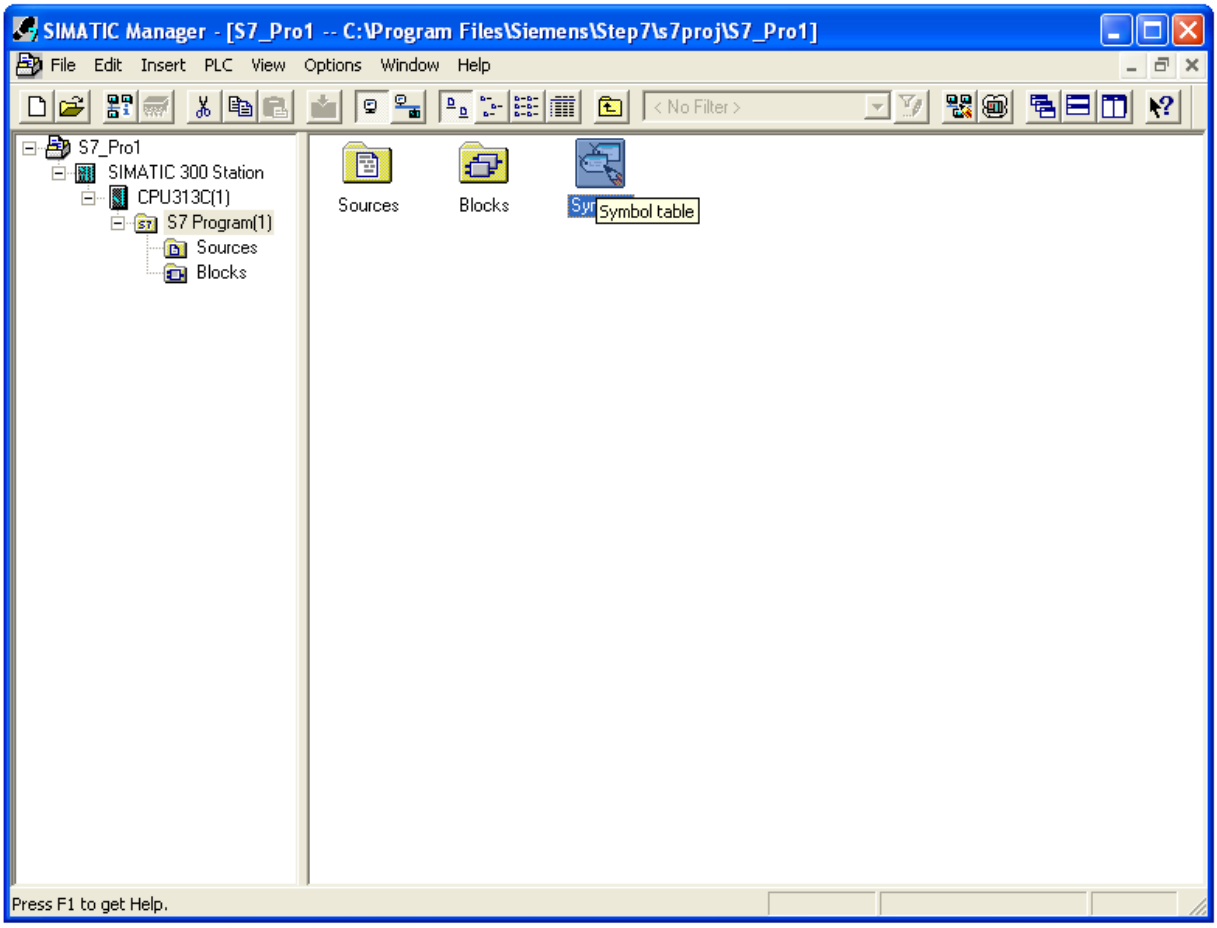
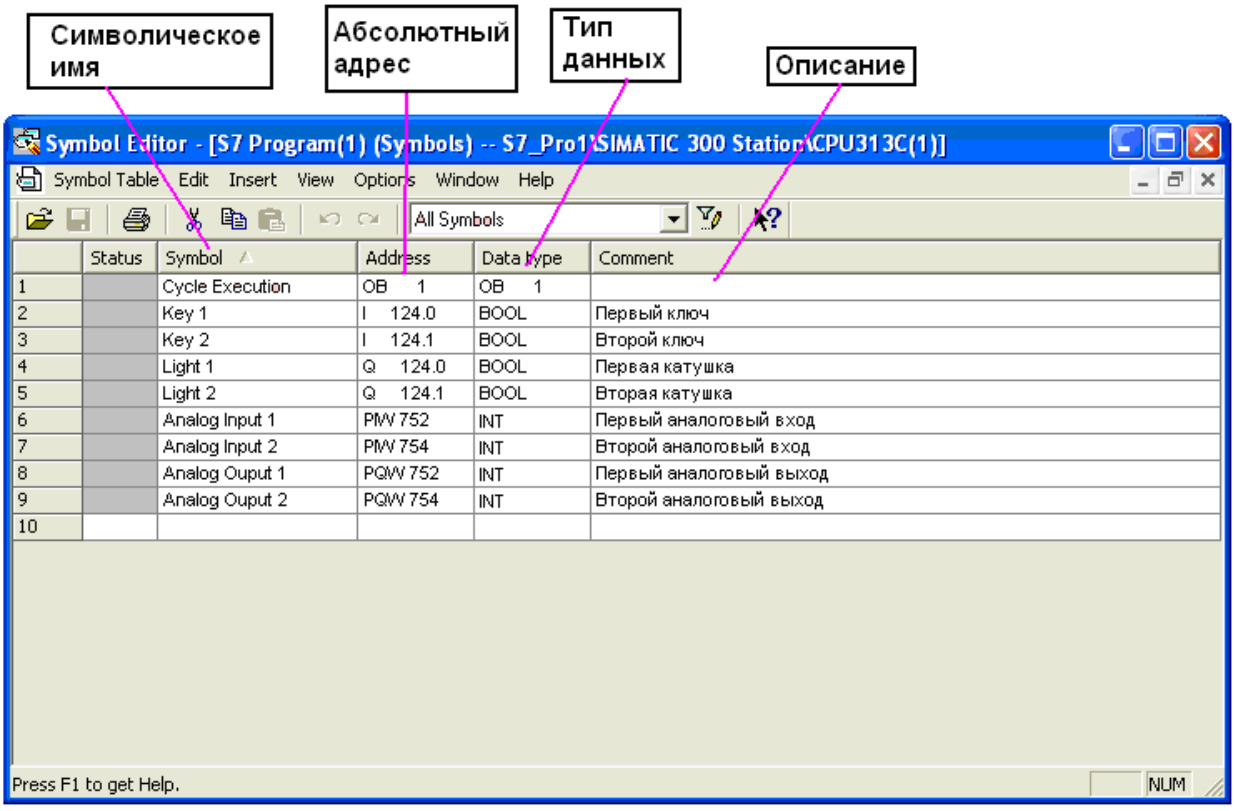


Рис. 8. Активизация таблицы символов



	Status	Symbol	Address	Data type	Comment
1		Cycle Execution	OB 1	OB 1	
2		Key 1	I 124.0	BOOL	Первый ключ
3		Key 2	I 124.1	BOOL	Второй ключ
4		Light 1	Q 124.0	BOOL	Первая катушка
5		Light 2	Q 124.1	BOOL	Вторая катушка
6		Analog Input 1	PIW 752	INT	Первый аналоговый вход
7		Analog Input 2	PIW 754	INT	Второй аналоговый вход
8		Analog Ouput 1	PQW 752	INT	Первый аналоговый выход
9		Analog Ouput 2	PQW 754	INT	Второй аналоговый выход
10					

Рис. 9. Таблица символов

Среди возможных данных могут выступать:

BOOL BYTE WORD DWORD	Данные этого типа являются комбинациями битов. От 1 бита (тип BOOL) до 32 битов (DWORD).
CHAR	Данные этого типа занимают ровно один символ из набора символов ASCII.
INT DINT REAL	Эти данные доступны для обработки числовых величин (например, для расчета арифметических выражений).
S5TIME TIME DATE TIME_OF_DAY	Данные этого типа представляют различные значения времени и даты внутри Step 7 (например, чтобы установить дату или ввести значение времени для таймера).

Более подробные данные вы сможете найти в справочной системе программы.

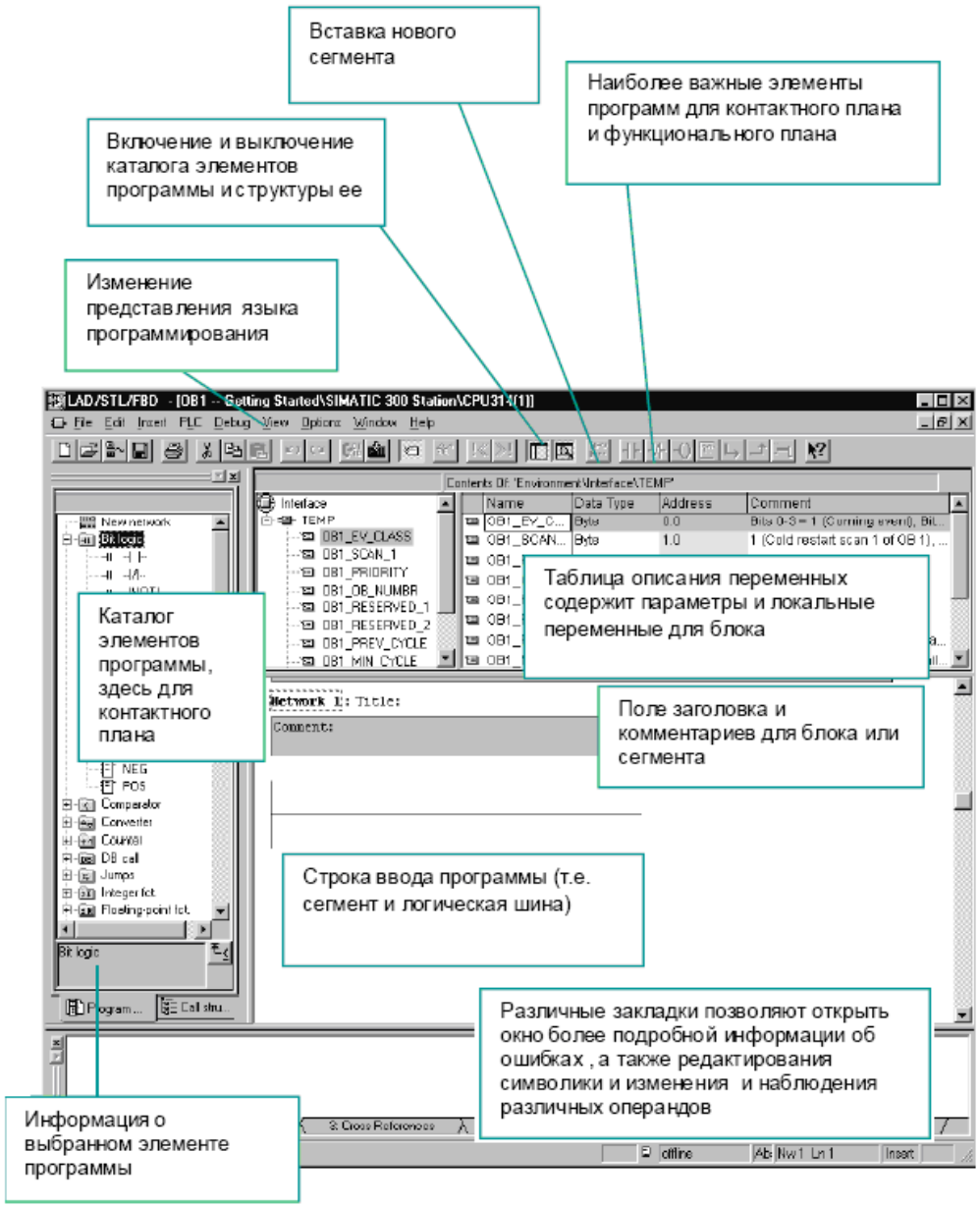
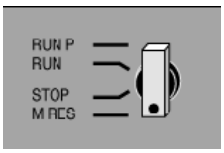
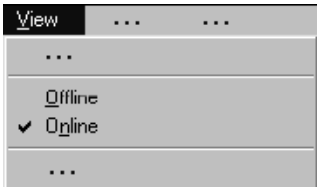


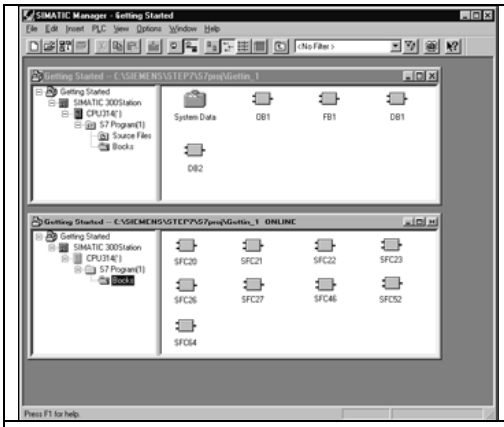
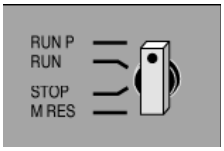
Рис. 10. Окно редактора блоков

Создадим простейшую программу и посмотрим, как создаются программы в Step 7. Для начала кликните в древе проектов на папку **Blocks**, а затем в правой панели дважды кликните на блок OB1, перед вами откроется окно, показанное на рис. 10. Кнопки программы имеют интуитивно понятное значение и всплывающие подсказки. Создайте программу, внешний вид которой показан на рис. 11. Кнопки, которые вам понадобятся при создании программы, обведены. Сохраните проект и произведите загрузку программы в ПЛК.

Загрузка программы в программируемый контроллер

Порядок включения контроллера и загрузки программы показан ниже.

	Включите питание стенда, включив выключатель Power на панели выходных сигналов (левая панель)
	Включите блок питания с помощью переключателя ON/OFF [ВКЛ\ВЫКЛ]. На CPU загорится диод «DC 5V» [5 В пост. тока].
	Поверните переключатель режимов работы в положение STOP (если он еще не находится в этом положении). Загорится красный светодиод «STOP» LED.
	Поверните переключатель режимов работы в положение MRES и удерживайте его там, в течение не менее 3 секунд, пока желтый светодиод «STOP» не начнет медленно мигать. Отпустите переключатель и спустя не более 3 секунд снова поверните его в положение MRES. Когда светодиод «STOP» замигает быстро, CPU сброшен. Если светодиод «STOP» не начинает быстро мигать, повторите эту процедуру (при этом вы стерли существующую программу в CPU).
	Чтобы загрузить программу, теперь снова поверните переключатель режимов работы в «STOP».
	Кроме окна «Ваш проект» откройте окно «Ваш проект ONLINE». Состояние online или offline показывается с помощью заголовков различного цвета.

	Переместитесь в обоих окнах к папке **Blocks [Блоки]**. Окно offline показывает содержимое программатора; окно online показывает блоки в CPU.
	Выделите папку **Blocks [Блоки]** в окне offline, а затем загрузите программу в CPU с помощью команды меню **PLC > Download [ПЛК > Загрузить]**. После запроса о перезаписи подтвердите команду с помощью кнопки OK.
	Поверните переключатель режимов работы в **RUN-P**. Загорится зеленый светодиод «**RUN**», а желтый светодиод «**STOP**» погаснет. CPU готов к работе. Когда загорается зеленый светодиод, вы можете начинать тестирование программы. Если продолжает гореть желтый светодиод, это значит, что произошла ошибка.

Для возможности наблюдения и отладки программы нажмите кнопку  в окне открытого блока OB1. Включите ключ I 124.0, и на экране программы вы увидите картинку, похожую на рис. 12. В данный момент времени включен один ключ и соответственно логике программы горит лампочка на выходе Q 124.0. Поэкспериментируйте с программой, чтобы понять, как работает Step 7. Следует отметить, что присвоение имени входам/выходам в таблице символов вовсе не обязательно, можно вводить непосредственно абсолютный адрес. Нажмите на кнопку  редактора блоков, и имена заменятся абсолютными адресами. И еще одно удобство, которое поможет вам быстрее создавать программы: если при вводе адреса нажать **Ctrl+Space** Step 7 сам предложит вам выбрать имена, имеющиеся в таблице символов (при этом отфильтрует те, которые в данном случае неуместны).

Программа Step 7 имеет богатые возможности по созданию программ, начиная с разобранной нами и программами с операциями над числами с плавающей точкой и заканчивая построением промышленных сетей и их конфигурирования (в зависимости от возможностей CPU). Наиболее удобно изучать программирование, приступая к созданию какого-нибудь проекта, а не простых конструкций, чем мы и займемся в дальнейшем.

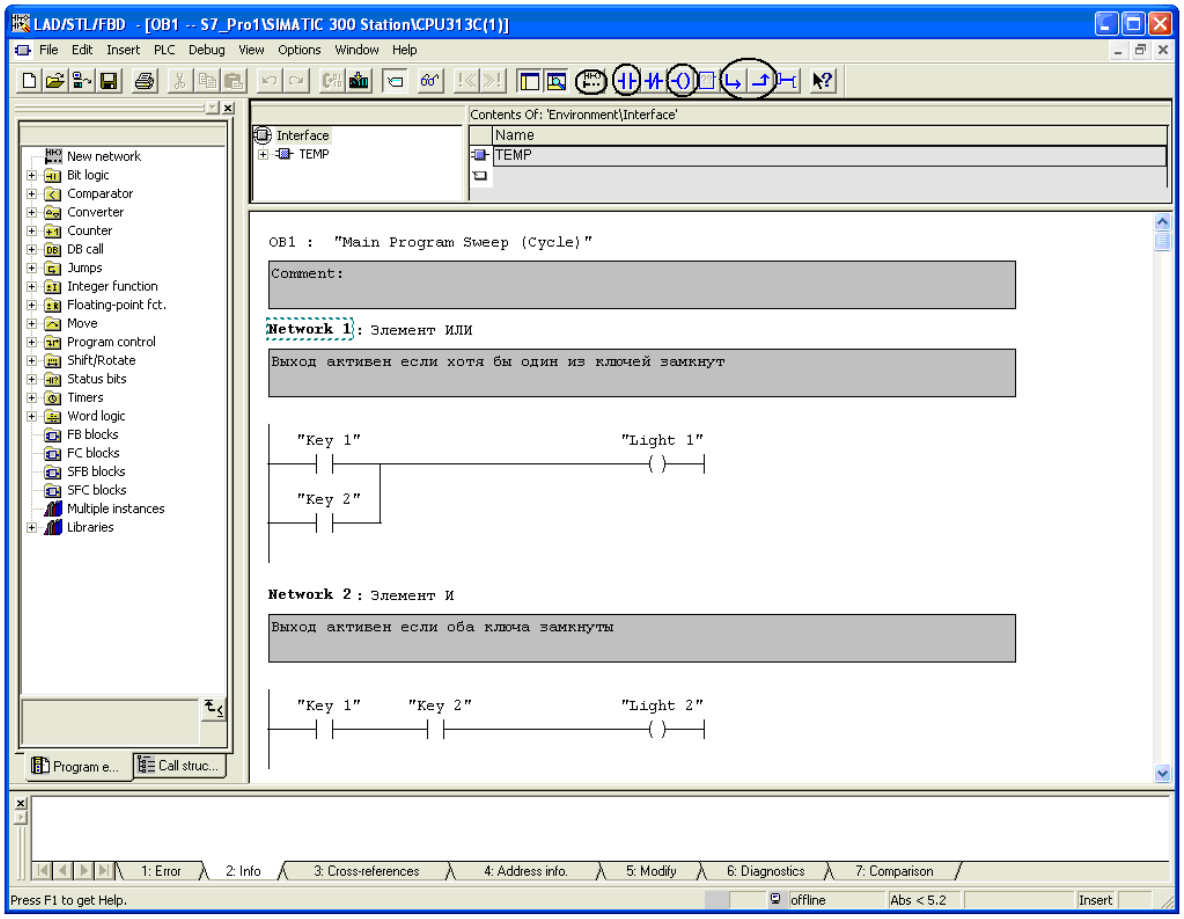


Рис. 11. Внешний вид программы

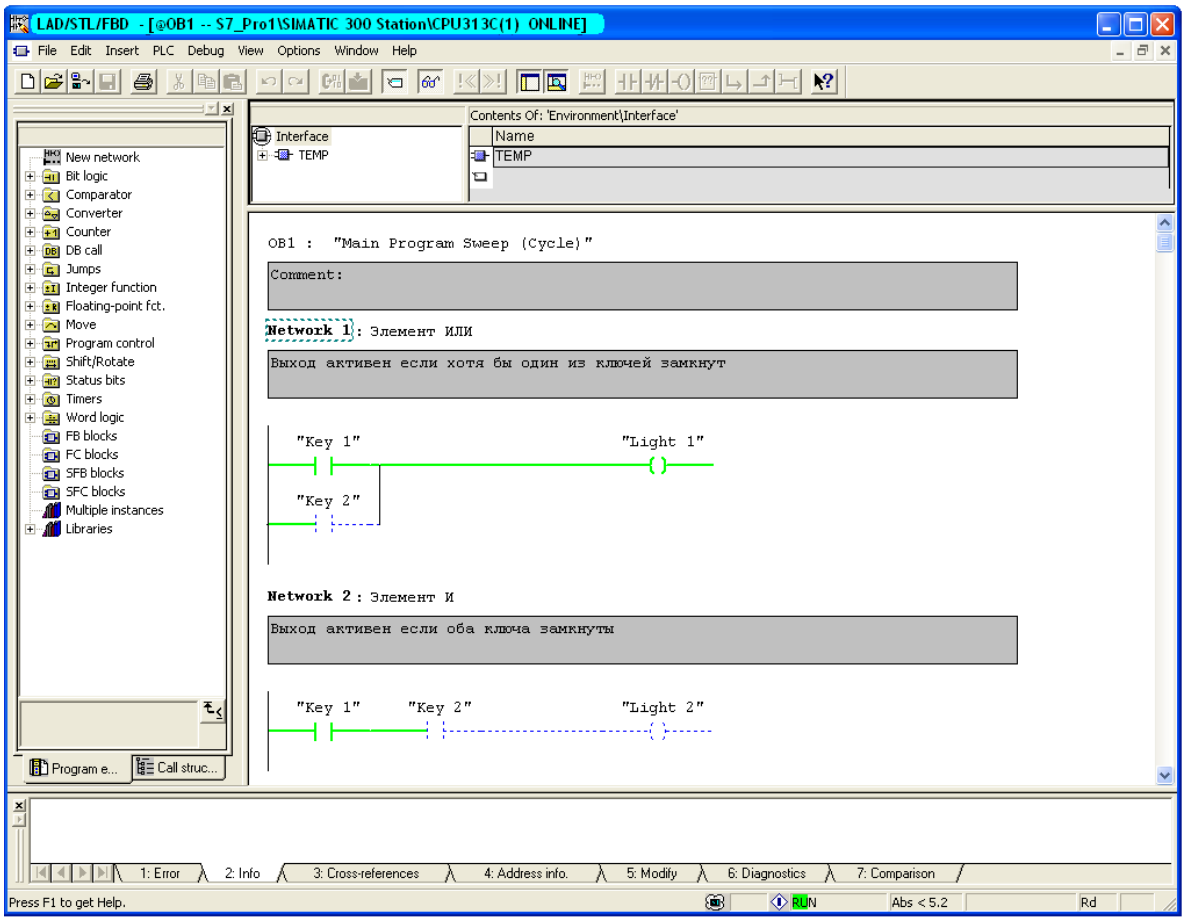


Рис. 12. Отладка программы

Использование аналоговых входов и выходов аналогично. Создайте программу, показанную ниже на рис. 13, для связи аналоговых входов и выходов (пример – аналоговый калькулятор). Необходимые при создании данной программы блоки выделены на нем. Обратите внимание, что аналоговый вход (10 В) представляется огромным числом, т.к. вход многобитный, впрочем его разрешение всего 2^{12}=4096, т.е. 10 В (реально чуть больше) делится на 4096 частей. Если вы работаете непосредственно с контроллером, то необходимо взять блоки их вкладки Integer function. Запустите программу, и на первом вольтметре будет показываться сумма падений напряжений на потенциометрах, а на втором – их разность.

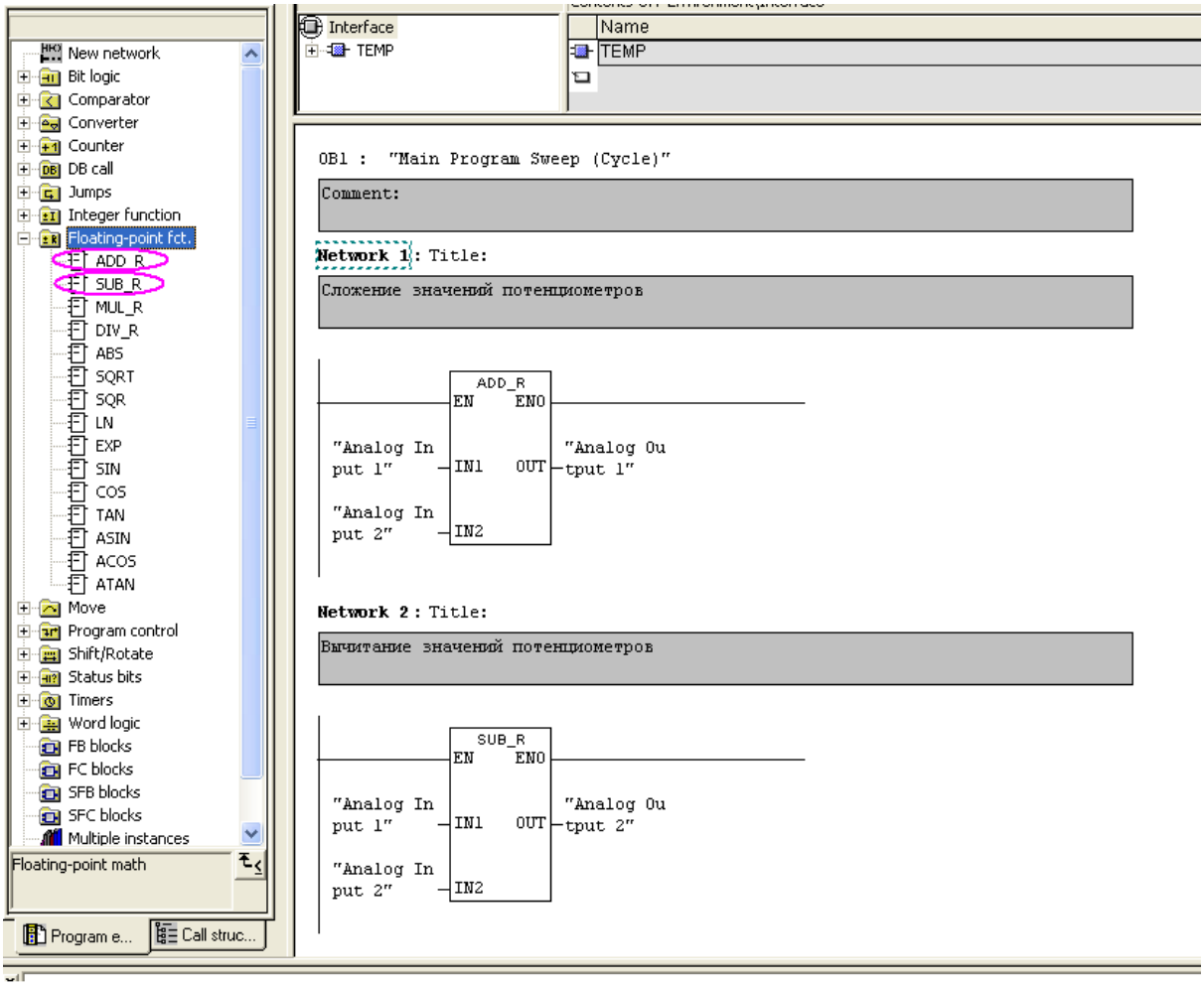
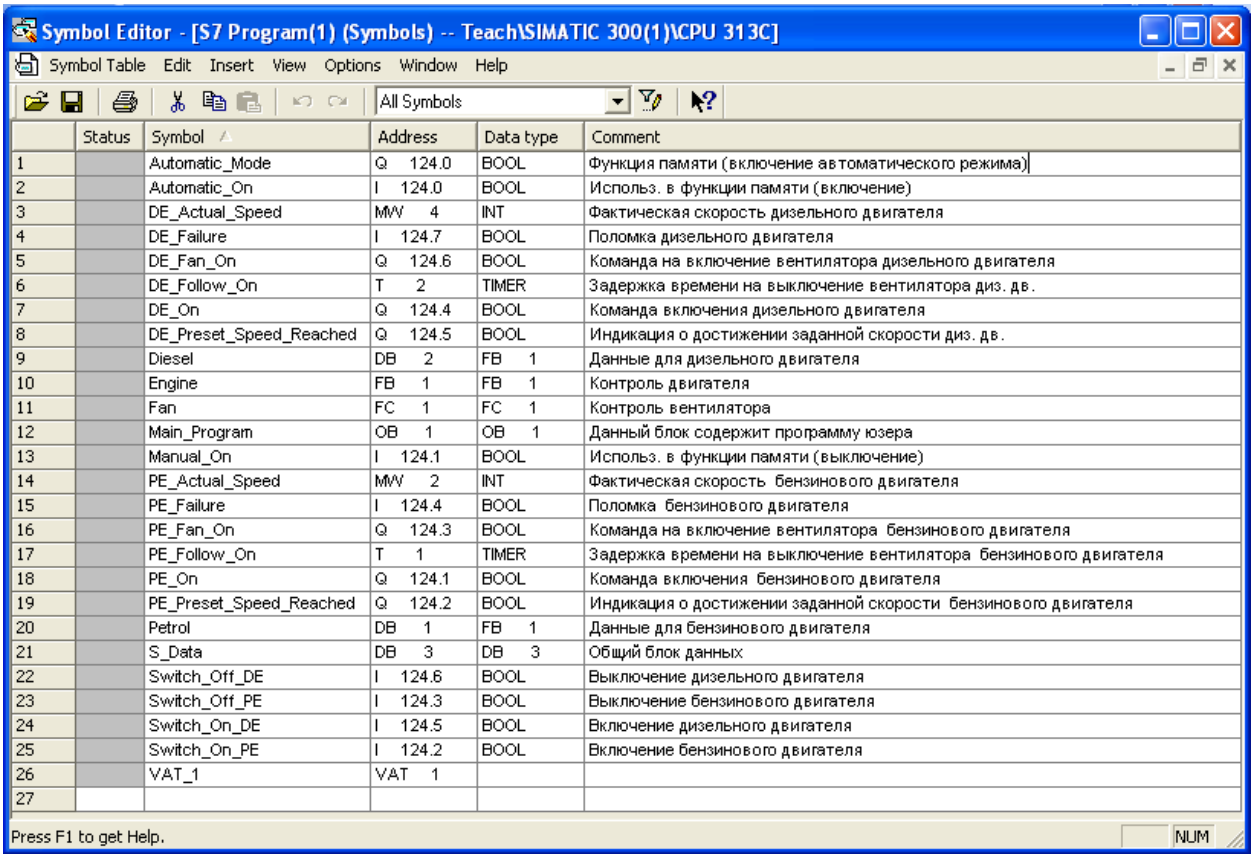


Рис. 13. Аналоговый калькулятор

В нашей работе мы попробуем не просто дать основы создания программ, но и способы их оптимизации. Создадим простую программу для управления двумя двигателями: бензиновым и дизельным. Естественно, любой двигатель необходимо включать и выключать, также необходимо следить за его исправностью двигателя (выключать его при аварии) и, конечно, необходимо следить за его скоростью. Программу для такого управления мы и будем создавать.

Для удобства написания программы создайте таблицу символов, показанную на рис. 14, при этом вы можете оставить уже созданный проект.



	Status	Symbol	Address	Data type	Comment
1		Automatic_Mode	Q 124.0	BOOL	Функция памяти (включение автоматического режима)
2		Automatic_On	I 124.0	BOOL	Использ. в функции памяти (включение)
3		DE_Actual_Speed	MW 4	INT	Фактическая скорость дизельного двигателя
4		DE_Failure	I 124.7	BOOL	Поломка дизельного двигателя
5		DE_Fan_On	Q 124.6	BOOL	Команда на включение вентилятора дизельного двигателя
6		DE_Follow_On	T 2	TIMER	Задержка времени на выключение вентилятора диз. дв.
7		DE_On	Q 124.4	BOOL	Команда включения дизельного двигателя
8		DE_Preset_Speed_Reached	Q 124.5	BOOL	Индикация о достижении заданной скорости диз. дв.
9		Diesel	DB 2	FB 1	Данные для дизельного двигателя
10		Engine	FB 1	FB 1	Контроль двигателя
11		Fan	FC 1	FC 1	Контроль вентилятора
12		Main_Program	OB 1	OB 1	Данный блок содержит программу юзера
13		Manual_On	I 124.1	BOOL	Использ. в функции памяти (выключение)
14		PE_Actual_Speed	MW 2	INT	Фактическая скорость бензинового двигателя
15		PE_Failure	I 124.4	BOOL	Поломка бензинового двигателя
16		PE_Fan_On	Q 124.3	BOOL	Команда на включение вентилятора бензинового двигателя
17		PE_Follow_On	T 1	TIMER	Задержка времени на выключение вентилятора бензинового двигателя
18		PE_On	Q 124.1	BOOL	Команда включения бензинового двигателя
19		PE_Preset_Speed_Reached	Q 124.2	BOOL	Индикация о достижении заданной скорости бензинового двигателя
20		Petrol	DB 1	FB 1	Данные для бензинового двигателя
21		S_Data	DB 3	DB 3	Общий блок данных
22		Switch_Off_DE	I 124.6	BOOL	Выключение дизельного двигателя
23		Switch_Off_PE	I 124.3	BOOL	Выключение бензинового двигателя
24		Switch_On_DE	I 124.5	BOOL	Включение дизельного двигателя
25		Switch_On_PE	I 124.2	BOOL	Включение бензинового двигателя
26		VAT_1	VAT 1		
27					

Рис. 14. Таблица символов для создания проекта

Создание функционального блока

Функциональный блок (FB) расположен в иерархии программы ниже организационного блока. Он содержит часть программы, которая может многократно вызываться в OB1. Все формальные параметры и статические данные функционального блока сохраняются в отдельном блоке данных (DB), назначаемом функциональному блоку. Вы будете программировать функциональный блок (FB1, символическое имя **Engine [Двигатель]**; см. таблицу символов) в окне для программирования LAD, с которым вы теперь знакомы. Для этого вам следует использовать тот же язык программирования, что и в главе 4 (программирование OB1).

Откройте в проекте папку **Blocks [Блоки]**. Щелкните в правой половине окна правой кнопкой мыши. Всплывающее меню для правой кнопки мыши содержит наиболее важные команды из строки меню. Вставьте в качестве нового объекта **Function Block [Функциональный блок] (New Object ->Function Block)**.

В диалоговом окне **Properties – Function Block [Свойства – Функциональный блок]** – выберите язык, на котором вы хотите создавать этот блок, активизируйте триггерную кнопку **Multiple instance FB** [Мультиэкземплярный FB] и подтвердите остальные параметры настройки, щелкнув на OK. Функциональный блок FB1 вставлен в папку блоков (**Blocks**). Дважды щелкните на значке FB1, чтобы открыть его.

Теперь посмотрим, как программировать функциональный блок. Все

данные, управления для двигателей передаются функциональному блоку из организационного блока как параметры блока и поэтому должны быть определены в таблице описания переменных как входные и выходные параметры (описание **in** и **out**).

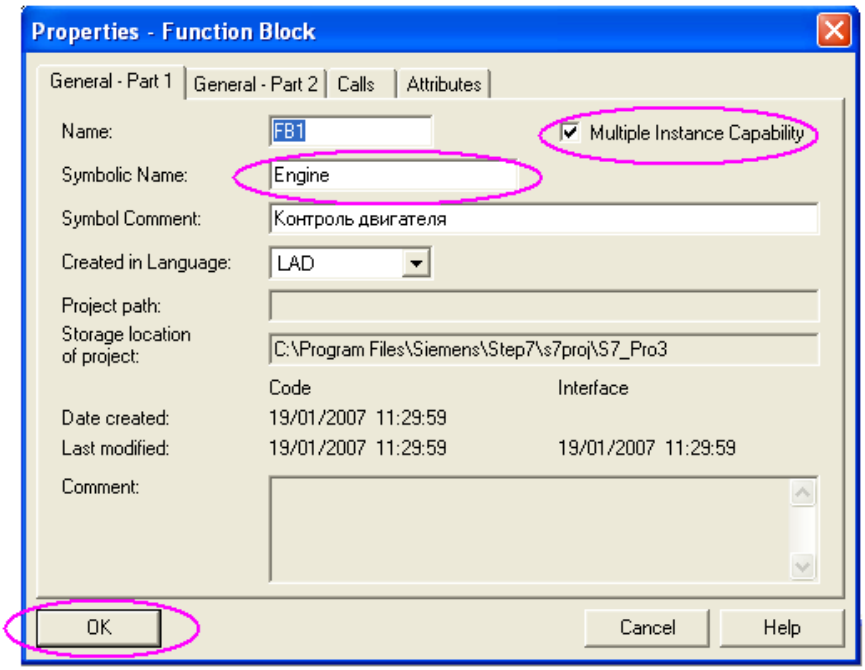


Рис. 15. Создание функционального блока

Таблица деклараций состоит из вида переменных (левая часть) и детального отображения задаваемых переменных (правая часть таблицы). Выбрав в левой части тип переменных IN, OUT, IN_OUT или STAT, введите необходимые имена переменных, тип данных и необходимые комментарии в правой части таблицы описания переменных (комментарии лучше взять в кавычки). Вы можете использовать выпадающее меню для задания типа переменных. В конце концов, у вас должно получиться таблица деклараций, показанная на рис. 16.

В таблице описания переменных для имен параметров блока разрешенными символами являются только буквы, цифры и знаки подчеркивания, допускаются только латинские буквы при описании переменных.

Вставьте нормально открытый контакт, нормально замкнутый контакт и элемент SR последовательно в сегменте 1 (Network 1), используя соответствующие кнопки на панели инструментов или каталог элементов программы. Затем выделите непосредственно вход R. Вставьте еще один нормально открытый контакт. Выделите шину непосредственно перед этим контактом (см. рис. 17). Вставьте нормально замкнутый контакт параллельно нормально открытому контакту. Результат всех действий показан на рис. 18.

Проверьте, активизировано ли символьное представление (Symbolic Representation) (View->Display with). Выделите вопросительные знаки и введите соответствующие имена из таблицы описания переменных (символ

появится автоматически). Результат показан на рис. 18.

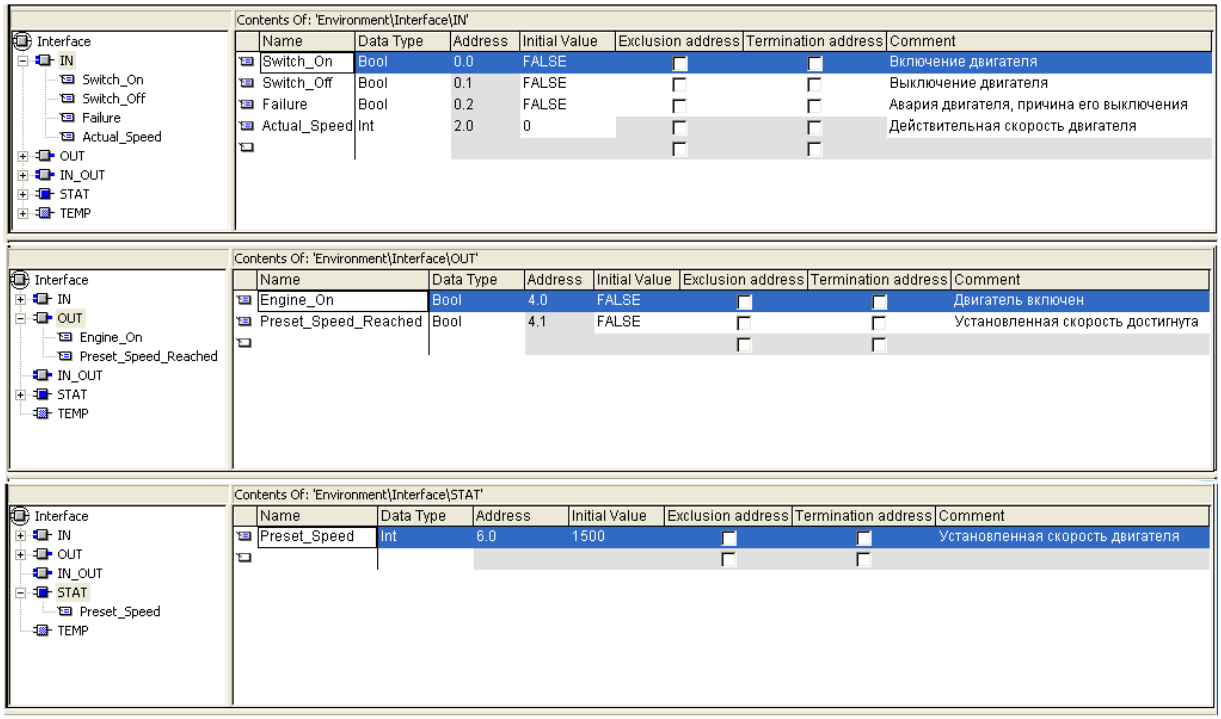


Рис. 16. Таблица деклараций

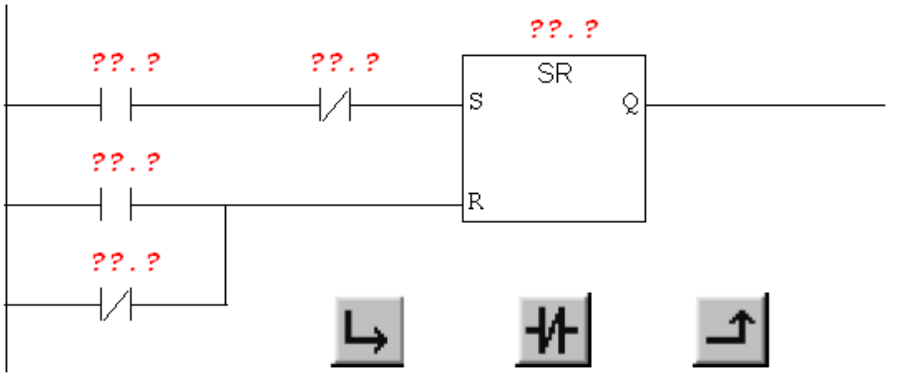


Рис. 17. Порядок действий

```
FB1 : Функциональный блок для контроля двигателя
Comment:
Network 1: Включение двигателя, н. открытый и н. закрытый контакт
Comment:
```

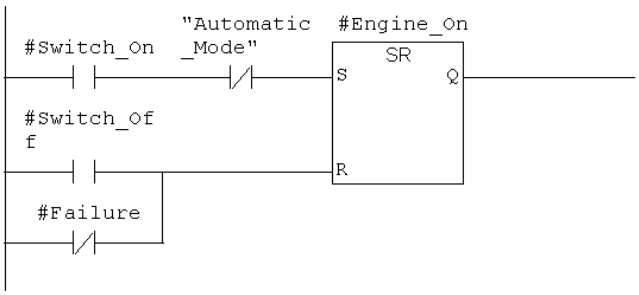


Рис. 18. Сегмент

Вставьте новый сегмент и выделите шину. Затем перемещайтесь в каталоге элементов программы, пока не достигнете функции **Compare [Сравнение]**, и вставьте CMP>=I. На выходе элемента сравнения установите катушку. Снова выделите вопросительные знаки и обозначьте катушку и блок сравнения именами из таблицы описания переменных. Затем сохраните свою программу. Вид получившегося сегмента показан на рис. 19.

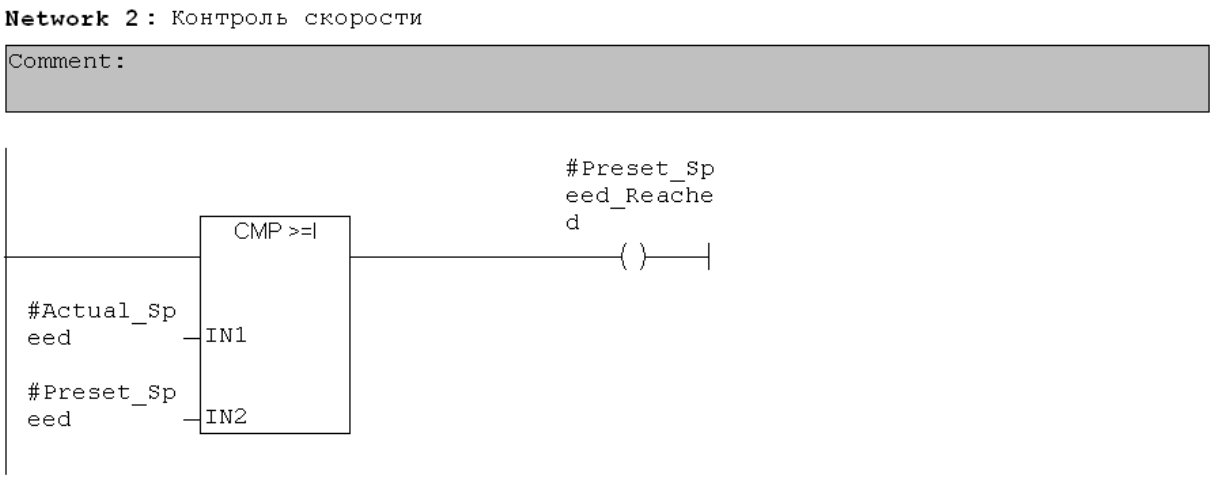


Рис. 19. Вид получившегося сегмента

Итак, что получилось. Когда переменная #Switch_On [включить] имеет состояние «1» и переменная Automatic_Mode [Автоматический_режим] имеет состояние «0», двигатель включается. Эта функция не разрешена, пока автоматический режим не выключен (отрицание Automatic_Mode, нормально замкнутый контакт). Когда переменная #Switch_Off [выключить] имеет состояние «1» или переменная #Fault [неисправность] имеет состояние «0», двигатель выключается. Эта функция снова реализуется путем отрицания переменной #Fault (#Fault – это «нуль-активный» сигнал, он равен «1» в нормальном состоянии и «0», если возникает неисправность). Блок сравнения сравнивает переменные #Actual_Speed [фактическая_скорость] и #Preset_Speed [заданная_скорость] и присваивает результат сравнения переменной #Preset_Speed_Reached [заданная_скорость_достигнута] (состояние сигнала «1»).

Генерация экземплярных блоков данных и изменение фактических значений

Вы только что создали функциональный блок FB1 (Engine [Двигатель]) и определили среди прочего специфические для двигателя параметры в таблице описания переменных. Чтобы в будущем получить возможность вызова функционального блока в OB1, вы должны сгенерировать соответствующий блок данных. Экземплярный блок данных (DB) всегда ставится в соответствие функциональному блоку. Функциональный блок должен управлять

и контролировать работу бензинового или дизельного двигателя. Различные заданные скорости двигателей хранятся в двух отдельных блоках данных, в которых изменяется задаваемое значение скорости (#Preset_Speed [заданная_скорость]). Централизованно программируя функциональный блок один раз, вы можете сократить объем программирования.

Переместитесь в папку **Blocks [Блоки]** и щелкните в правой половине окна правой кнопкой мыши. Вставьте блок данных (data block), используя всплывающее меню. Задайте имя блоку данных DB1, выберите тип блока «Instance DB» (блок-экземпляр) и поставьте ему в соответствие функциональный блок FB1 (рис. 21). Примите все параметры настройки, отображаемые в диалоговом окне **Properties [Свойства]** щелкнув на OK.

Блок данных DB1 добавляется к проекту. Дважды щелкните на значке блока DB1, чтобы открыть его.

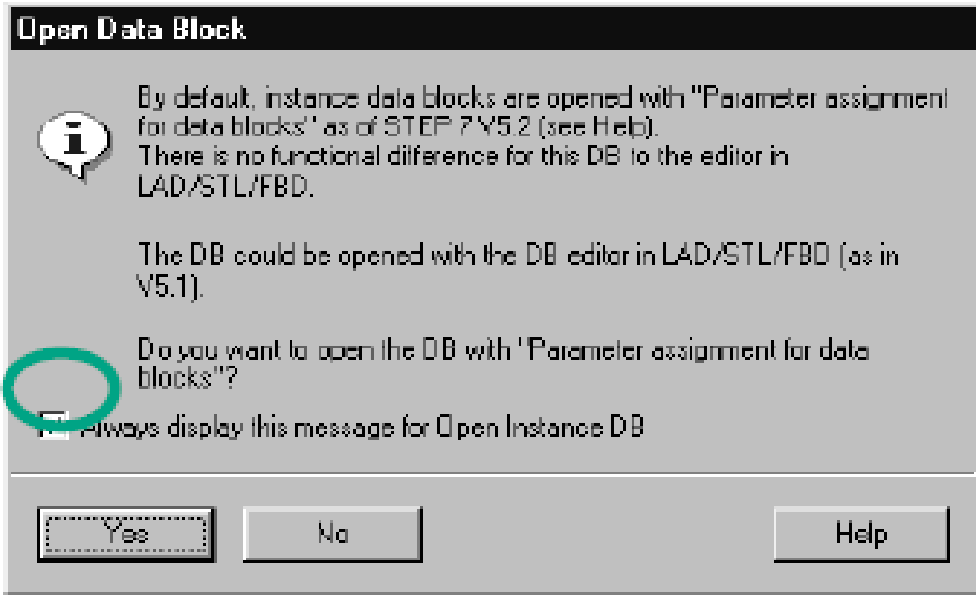


Рис. 20. Назначение блока- экземпляра

Подтвердите назначение блока-экземпляра для FB1, **Engine [Двигатель]** с помощью OK (рис. 20). Затем введите значение «1500» для бензинового двигателя в столбец **Actual Value [Фактическое значение]** (в строке Preset_Speed) (рис. 22). Теперь вы определили максимальную скорость для этого двигателя. Сохраните DB1 и закройте блок. Как и в случае с DB1, сгенерируйте еще один блок данных, DB2, для FB1. Теперь введите значение «1200» для дизельного двигателя.

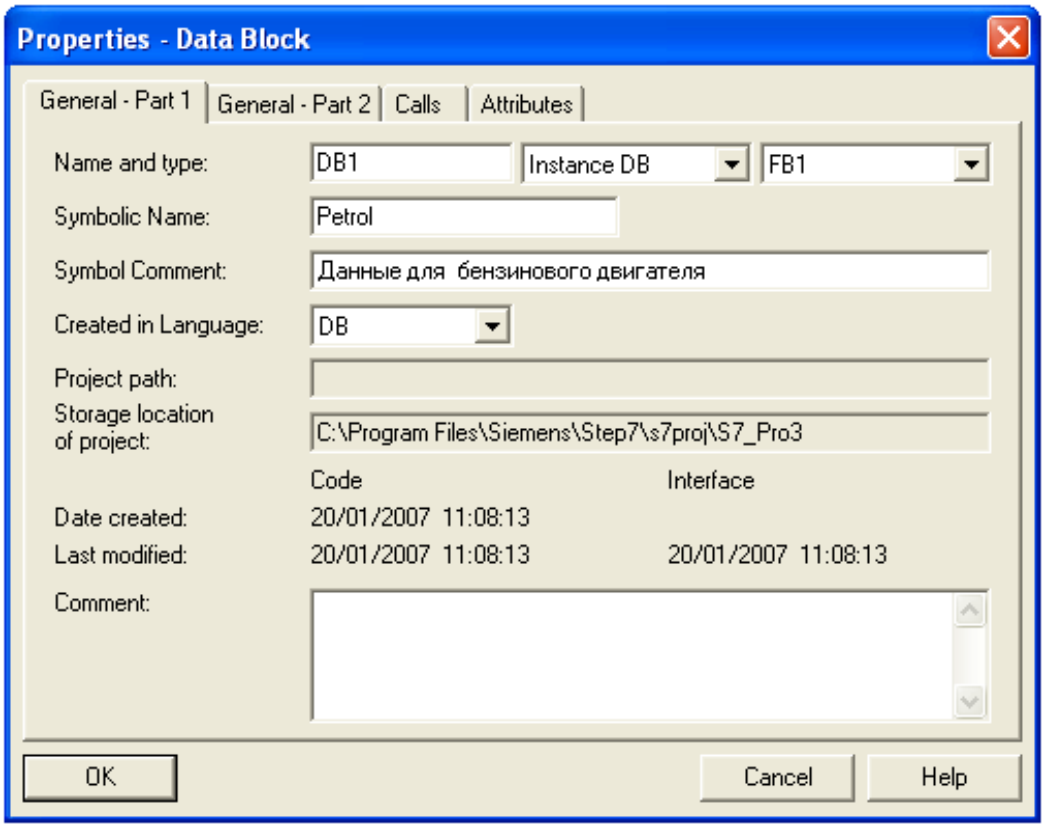


Рис. 21. Выбор типа блока

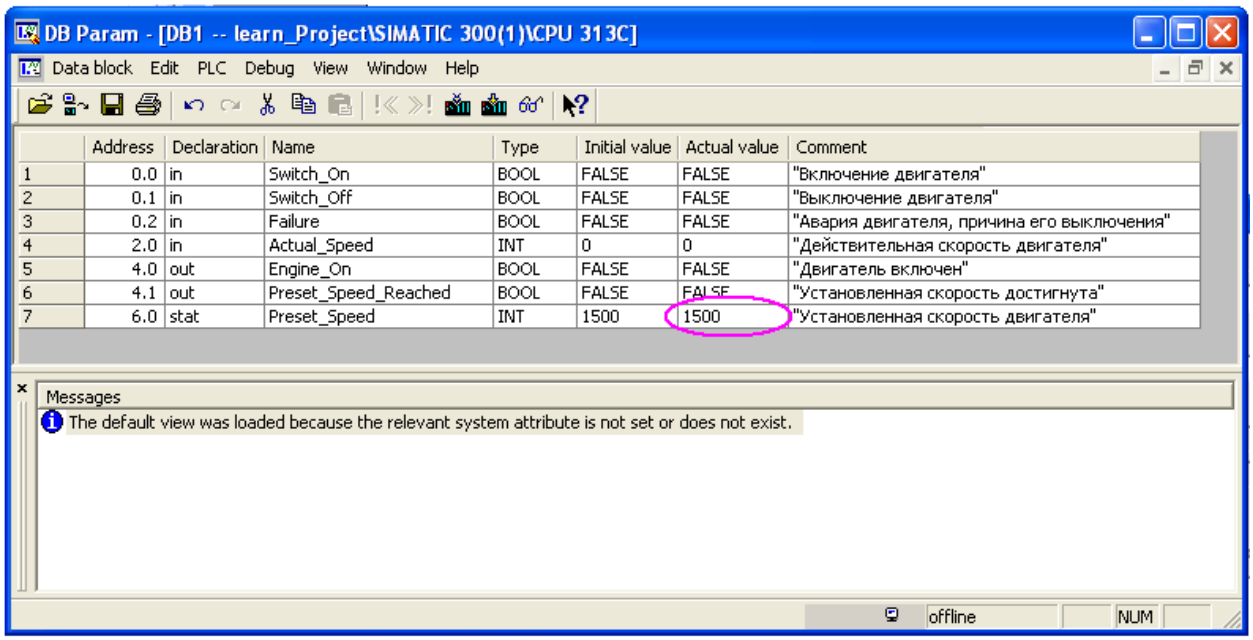


Рис. 22. Задание переменных

Изменив фактические значения, вы закончили приготовления к управлению двумя двигателями с помощью только одного функционального блока. Для управления большим количеством двигателей единственное, что вам нужно сделать, это сгенерировать дополнительные блоки данных. Следующий шаг, который вы должны выполнить, это запрограммировать вызов функционального блока в OB1.

Дважды щелкните на OB1. Затем перемещайтесь в каталоге элементов

программы, пока не достигнете папки функциональных блоков. Выделите FB1 и вставьте этот блок в OB1. Вставьте нормально открытый контакт перед каждым из следующих входов: Switch_On [Включить], Switch_Off [Выключить] и Fault [Неисправность]. Щелкните на знаке ??? над блоком Engine [Двигатель], а затем, удерживая курсор в том же положении, щелкните правой кнопкой мыши в рамке ввода. Используя левую кнопку мыши, выберите во всплывающем меню Insert Symbol [Вставить символ] или нажмите **CTRL+SHIFT**. Появится прокручиваемый список. Когда вы это делаете в первый раз, эта процедура может занять некоторое время. OB1 – вызов DB1 – данные бензинового двигателя, DB2 – данные дизельного двигателя FB1 Engine. Результат показан на рис. 23.

Когда вы создаете структуры программ с организационными блоками, функциональными блоками и блоками данных, вы должны программировать вызов для подчиненных блоков (таких, как FB1) в блоке, расположенном в иерархии более высоко (например, OB1). Эта процедура всегда одна и та же. Можно также давать различным блокам символьные имена в таблице символов (например, FB1 имеет имя Engine [Двигатель], а DB1 – имя Petrol [Бензиновый]).

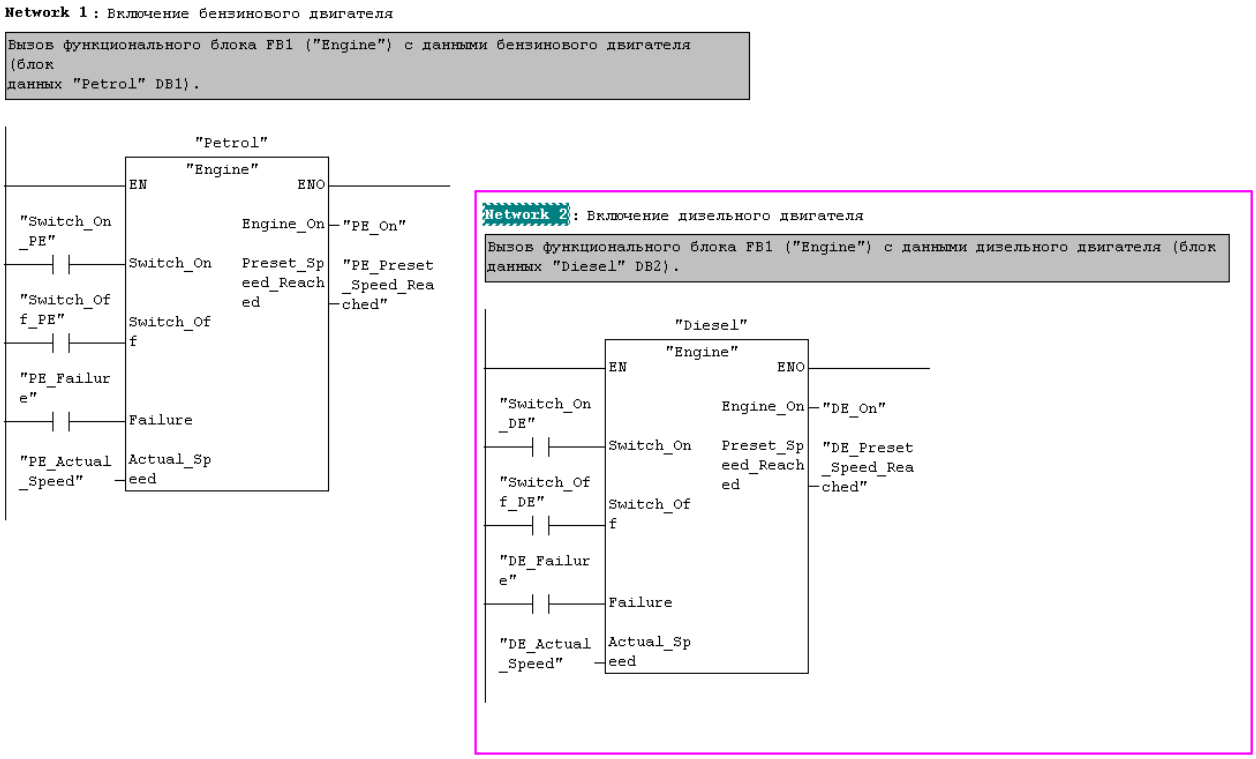


Рис. 23. Результат ввода данных

Загрузите программу в ПЛК, протестируйте программу, проверьте, запускаются ли двигатели. Если не помните значения переключателей (входов и выходов), то можете посмотреть их в таблице переменных.

Тестирование программы

Используя функцию **Статус программы**, вы можете тестировать программу в блоке. Предпосылкой для этого является установление соединения online с CPU. CPU должен находиться в режиме RUN или RUN-P, а программа должна быть загружена. Откройте OB1 в окне проекта **Getting Started ONLINE.** Открывается окно для программирования LAD/STL/FBD. Активизируйте функцию **Debug > Monitor [Отладка > Наблюдение]**. Вы увидите, «как проходят сигналы», например, замкните ключ I 124.4 (включение этого ключа соответствует исправности бензинового двигателя – такие функции обычно всегда используют нормально открытый контакт, и его замыкание свидетельствует об исправности системы, как правило, это несколько последовательно соединенных контактов защит) и включите контакт I 124.2 (включение бензинового двигателя). При этом вы увидите, что двигатель заработал – переменная PE_On приняла значение 1 (рис. 24).

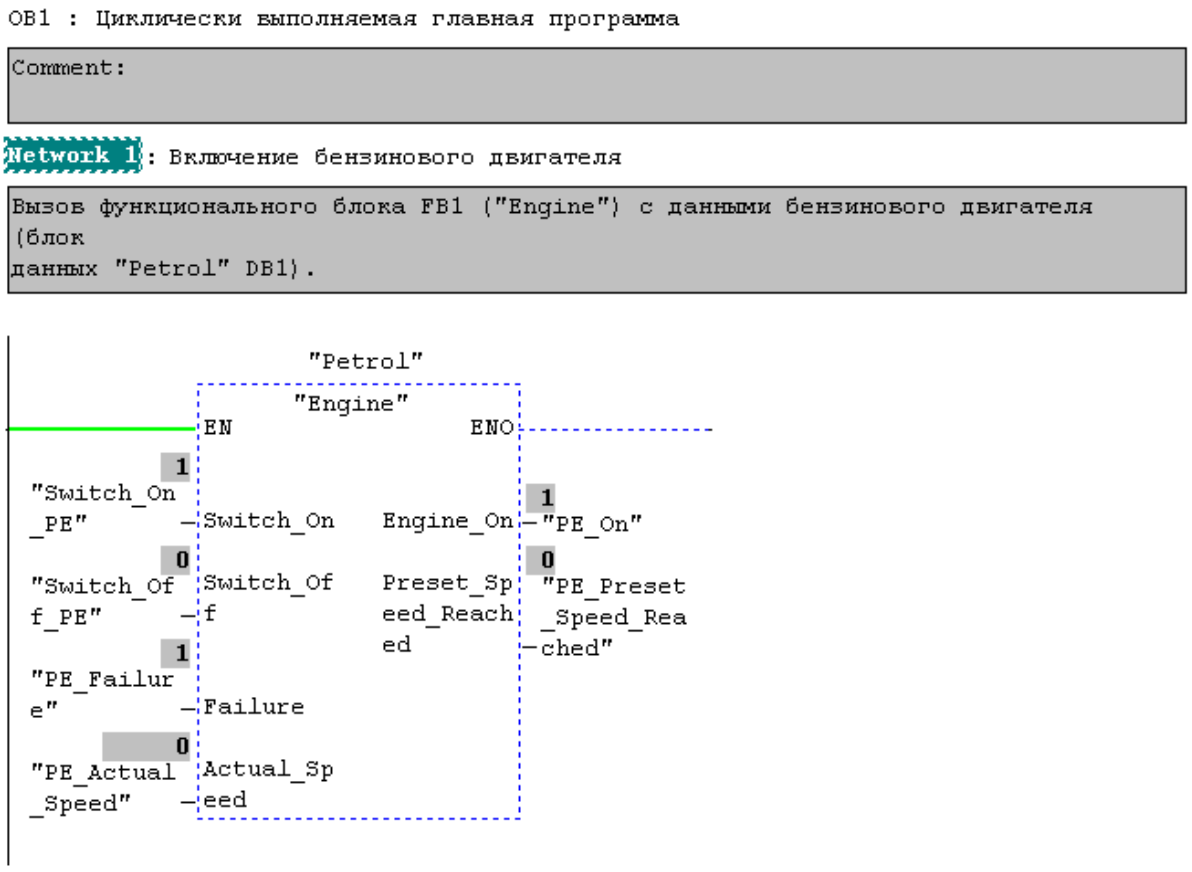


Рис. 24. Программа в работе

Вы можете тестировать отдельные переменные программы путем их изменения и наблюдения. Предпосылкой для этого является установление соединения online с CPU. CPU находится в режиме RUN-P, а программа – загружена. Как и при тестировании с помощью статуса программы, вы можете отображать входы и выходы в таблице переменных. Вы можете также тестировать блок сравнения для скорости двигателя в FB1 путем

предварительного задания фактической скорости.

Переместитесь к папке Blocks [Блоки] и щелкните правой кнопкой мыши в правой половине окна. Используйте правую кнопку мыши, чтобы вставить Variable Table [Таблицу переменных] из всплывающего меню. Примите параметры настройки по умолчанию, закрыв диалоговое окно **Properties [Свойства]** щелчком на OK.

VAT1 (таблица переменных) создается в папке блоков (рис. 25). Дважды щелкните на VAT1, чтобы открыть таблицу; откроется окно Monitoring and Modifying Variables [Наблюдение и изменение переменных] (рис. 26).

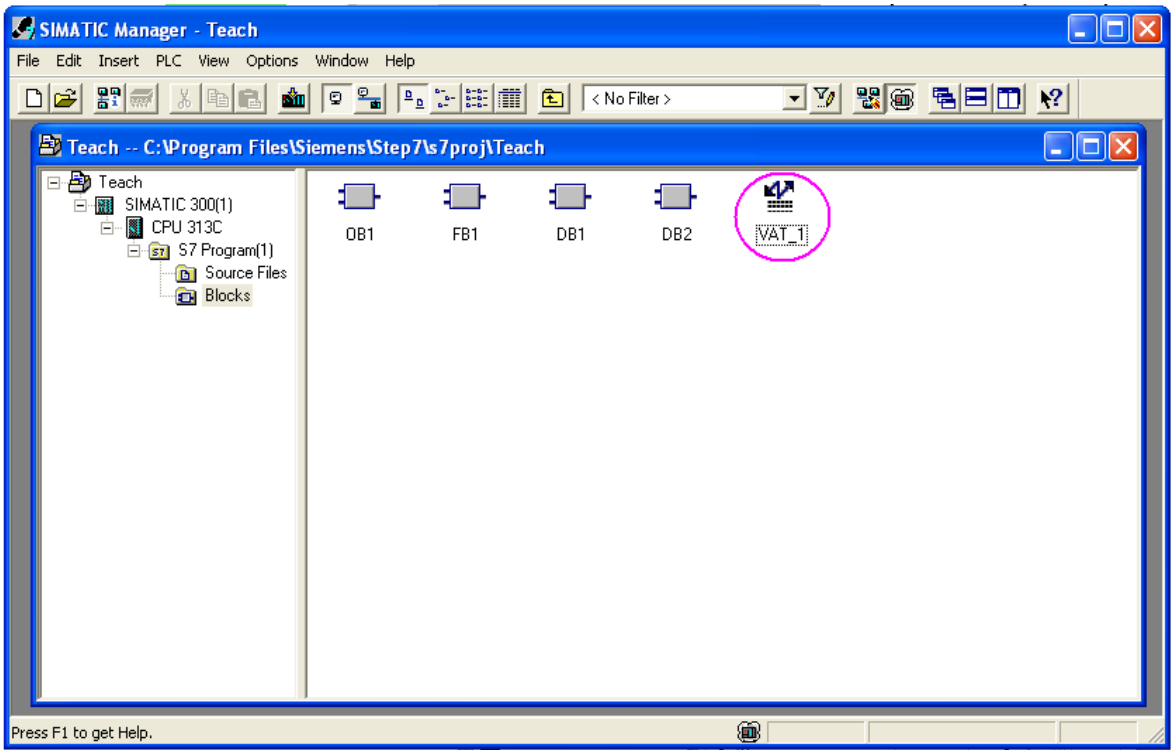


Рис. 25. Создание таблицы переменных

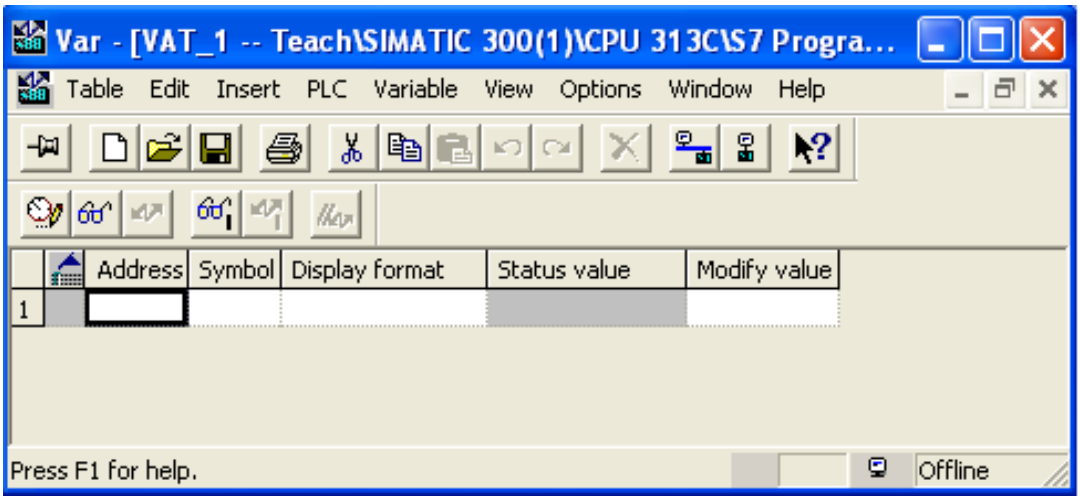


Рис. 26. Создание таблицы переменных

Сначала таблица переменных пуста. Введите символические имена или

адреса, как показано на рис. 27. Остальные элементы будут добавлены, когда вы завершите ввод нажатием Enter. Замените формат наблюдения (Monitior Format) всех значений скорости форматом DEC (десятичный). Для этого щелкните на соответствующей ячейке в заголовке (курсор превращается в стрелку над столбцом Monitior Format) и, используя правую кнопку мыши, выберите формат DEC.

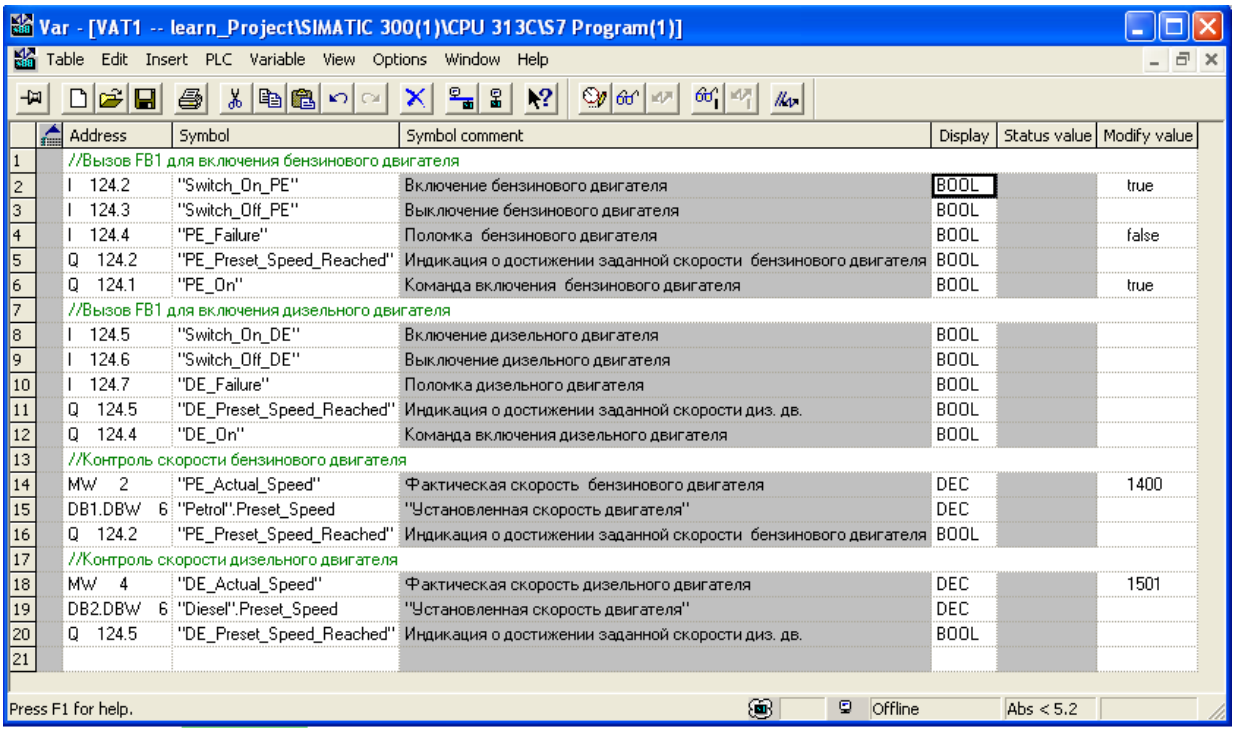


Рис. 27. Мониторинг переменных

Сохраните свою таблицу переменных  .

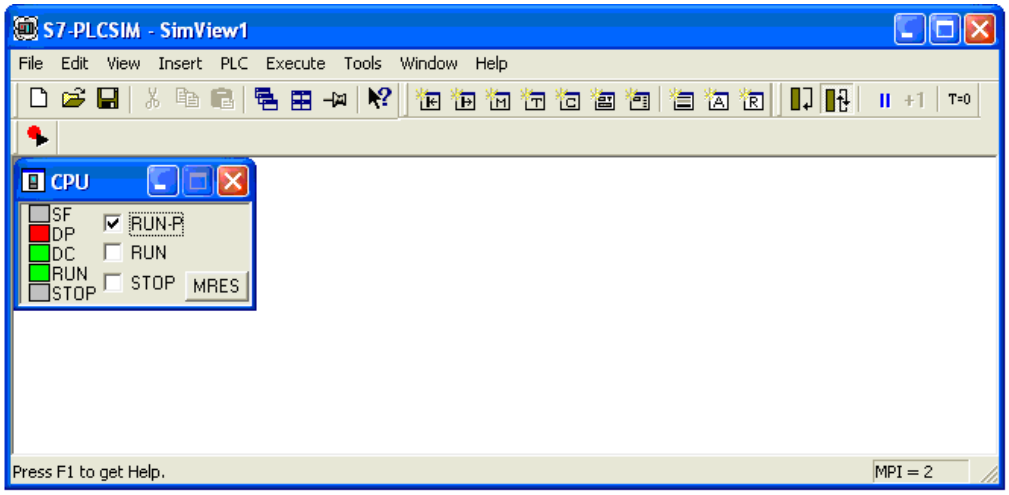


Рис. 28. Вид отладчика программы

Щелкните на кнопке Monitor  Variables на панели инструментов. Нажмите ключи 1 и 2 в своей тестовой конфигурации и наблюдайте за результатом в таблице переменных (если контроллер включен). Значения состояния в таблице переменных изменятся с false [ложь] на true [истина].

Введите значение «1500» для адреса MW2 в столбце Modify Value [Изменение значений] и «1300» – для адреса MW4. Передайте измененные значения в свой CPU .

Отладка программ без контроллера

При выполнении программирования для отладки программы не обязательно иметь контроллер, программа может быть отлажена и без него. Для отладки программы можно использовать имеющийся в Step 7 симулятор.

Для вызова симулятора в главном окне Simatic Manager необходимо выбрать Options->Simulate Modules, при этом контроллер стенда должен быть выключен. Появится окно, показанное на рис. 28.

Используя меню Insert, вставьте Inputs Variable (или нажмите F2). Вставьте еще один Inputs Variable и вставьте два окна Vertical Bits. В соответствующих полях введите значения, показанные на рис. 29. Загрузите программу в симулятор, точно так же, как вы загружали ее в реальный контроллер.

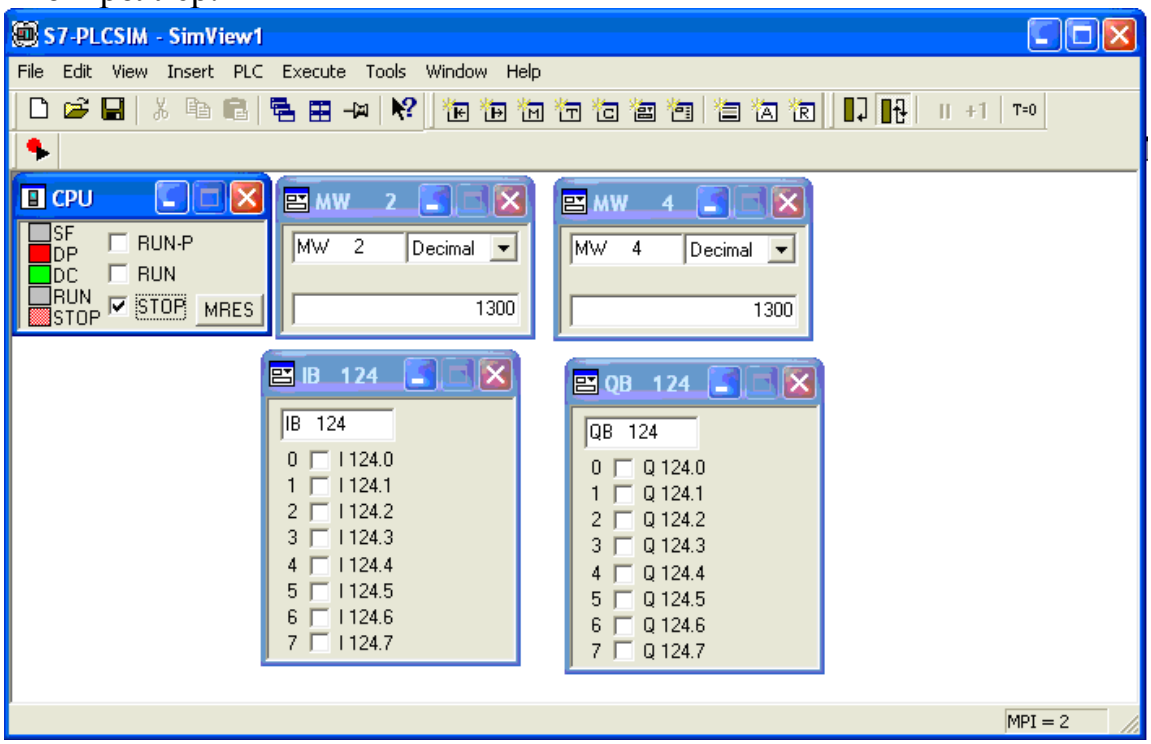


Рис. 29. Отладка программы

Запустите симулятор, поставив галочку в окне CPU в поле RUN или RUN-P. Поставив галочки в соответствующие поля IB (Input Bits), вы сможете сымитировать входные сигналы контроллера и увидеть значения выходов в QB. Для большего удобства измените малоговорящие абсолютные адреса на псевдонимы. Для этого выберите в меню Tools->Options->Attach Symbols и выберите таблицу переменных созданного проекта. При необходимости в этом же подпункте меню поставьте галочку в Show Symbols. Результат показан на рис. 30. В симуляторе можно изменять выходы контроллера без изменения входов, что невозможно в реальной

системе, но иногда необходимо при отладке. Все способы отладки, описанные выше, применимы при использовании имитатора.

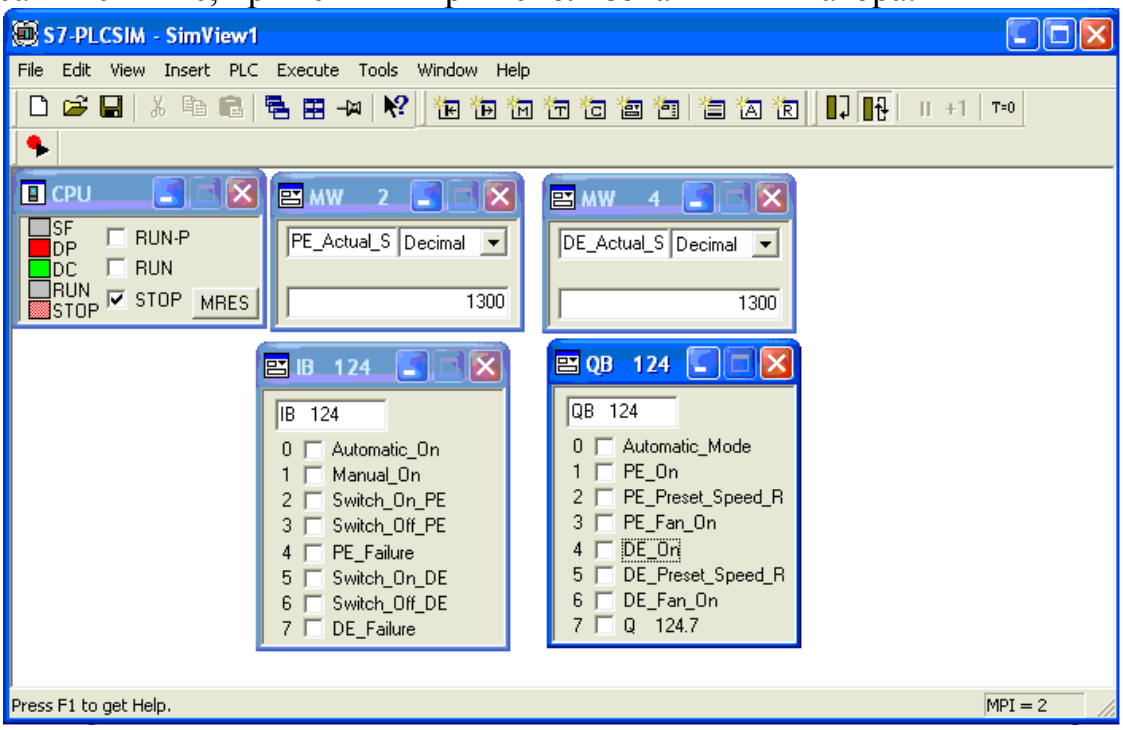


Рис. 30. Добавление подписей к переменным

На этом наше методическое пособие заканчивается. В ходе выполнения практического задания по лабораторной вам будет необходимо воспользоваться справкой по программе или русифицированной документацией по Step 7, т.к. весь объем информации охватить невозможно.

Форма отчетности

Письменного отчета по лабораторной работе не делается. Защита лабораторной работы заключается в выполнении индивидуального практического задания, выдаваемого преподавателем, по программированию ПЛК.

Учебное издание

ИССЛЕДОВАНИЕ РАБОТЫ ПРОГРАММИРУЕМОГО ЛОГИЧЕСКОГО КОНТРОЛЛЕРА SIMATIC S7-313C

Методические указания к лабораторной работе по курсу
«Элементы систем автоматики»

Составители: Аристов Евгений Валерьевич,
Хузин Руслан Альвертович

Редактор и корректор *Е.В. Копытова*

Подписано в печать 22.01.08 Формат 60x90/16.
Уч. печ. л. 2,0.
Тираж 100 экз. Заказ № 7/2008.

Издательство
Пермского государственного технического университета.
Адрес: 614990, г. Пермь, Комсомольский пр., 29, к. 113.
Тел. (342) 219-80-33.