ТЕХНОЛОГИЯ, ОБОРУДОВАНИЕ И АВТОМАТИЗАЦИЯ
АШИНОСТРОИТЕЛЬНЫХ
РОИЗВОДСТВ
СЕРИЯ ОСНОВАНА В 1996 ГОДУ

Митин Г.П., Хазанова О.В.

СИСТЕМЫ АВТОМАТИЗАЦИИ
с использованием программируемых логических контроллеров

Москва 2005

ТЕХНОЛОГИЯ, ОБОРУДОВАНИЕ И АВТОМАТИЗАЦИЯ
МАШИНОСТРОИТЕЛЬНЫХ ПРОИЗВОДСТВ
СЕРИЯ ОСНОВАНА В 1996 ГОДУ

Министерство образования и науки Российской Федерации

Федеральное агентство по образованию

Московский государственный технологический университет «СТАНКИН»

Учебно-методическое объединение по образованию в области автоматизированного машиностроения (УМО АМ)

Митин Г.П., Хазанова О.В.

СИСТЕМЫ АВТОМАТИЗАЦИИ
с использованием программируемых логических контроллеров

Учебное пособие

Рекомендовано Учебно-методическим объединением вузов по образованию в области автоматизированного машиностроения (УМО АМ) в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлениям «Конструкторско-технологическое обеспечение машиностроительных производств»; «Автоматизированные технологии и производства»

Москва 2005

УДК 681.5.001.2 (075)
М 66

Рецензенты: доктор технических наук, профессор
Радкевич Я.М. (МГГУ)

кандидат технических наук, доцент
Комаров Ю.Ю. (МАИ)

Митин Г.П., Хазанова О.В. **Системы автоматизации с использованием программируемых логических контроллеров**: Учебное пособие.– М.: ИЦ МГТУ «Станкин», 2005. – 136 с.

Рис. 53. Табл.16. Библ. 8 назв.

В учебном пособии рассмотрены вопросы решения задач автоматизации с использованием микроконтроллеров S7-200. Приведены основные технические характеристики, режимы работы и система команд микроконтроллеров этой серии. Особое внимание уделено типовым алгоритмам решения отдельных задач.

Пособие предназначено для студентов и инженерно-технических работников, занимающихся изучением и разработкой систем автоматизации на базе микроконтроллеров S7-200.

Митин Г.П., Хазанова О.В.

Системы автоматизации с использованием программируемых логических контроллеров

Учебное пособие

Лицензия на издательскую деятельность ЛР №01741 от 11.05.2000
Подписано в печать 26.04.2005. Формат 60x90$^{1}/_{16}$
Уч. изд. л. 8,5. Тираж 250 экз. Заказ № 77

Отпечатано в Издательском Центре ГОУ МГТУ «СТАНКИН»
103055, Москва, Вадковский пер., д.3а

ПРЕДИСЛОВИЕ

Темпы появления новых аппаратных и программных средств для решения задач автоматизации выдвигают требования к быстрому освоению их студентами ВУЗов. Чтобы не отстать от прогресса, необходимо регулярно обновлять учебную базу, что сегодня очень непросто. Поэтому появление на кафедре компьютерных систем управления МГТУ «Станкин» новой учебной лаборатории программируемых логических контроллеров (ПЛК) явилось долгожданным и важным событием.

Оснащение учебной лаборатории современными средствами позволило включить в учебный процесс вопросы проектирования систем управления различным оборудованием на базе программируемых логических контроллеров. Решены вопросы проведения лабораторных работ по нескольким учебным курсам: «Управление процессами, объектами и системами», «Аппаратные средства систем управления» и «Управление цикловой автоматикой».

В лаборатории используются микро-ПЛК SIMATIC S7-200 фирмы Siemens (Германия) с процессором CPU-214, которые являются типичными представителями программируемых логических контроллеров своего класса.

Состав и сложность лабораторных работ зависят от количества часов и уровня подготовки студентов. Задачи, решаемые студентами, охватывают широкий круг объектов: подъемно-транспортное оборудование (лифты, транспортеры, эскалаторы и т.д.), технологическое оборудование (магазины инструментов, тележки, роботы и т.д.), охранную сигнализацию, освещение, раздаточные автоматы, турникеты и др.

Несмотря на большое количество статей о программируемых логических контроллерах – все они представляют собой или рекламу или техническое описание отдельных моделей, а учебная литература, в первую очередь необходимая для студентов и специалистов, осваивающих работу с ПЛК, практически отсутствует.

Все вышесказанное и явилось предпосылкой для написания данной книги.

ВВЕДЕНИЕ

Первые программируемые логические контроллеры (ПЛК) появились в 1967 г. и были предназначены для локальной автоматизации наиболее часто встречающихся в промышленности технологических задач, которые часто описывались преимущественно логическими уравнениями. ПЛК с успехом заменили блоки релейной автоматики и устройства жесткой логики на интегральных микросхемах малой и средней степени интеграции [1]. Отсюда и название – программируемый логический контроллер (Programmable Logic Controller).

Сегодня ПЛК – это микропроцессорная система специального назначения с проблемно-ориентированным программным обеспечением для реализации алгоритмов логического управления и замкнутых систем автоматического управления в сфере промышленной автоматики. ПЛК отличаются универсальностью структуры и инвариантностью по отношению к объекту управления в пределах определенного класса задач [2].

Программное обеспечение ПЛК не является открытым, однако Международной Электротехнической Комиссией в 1992 г. разработан стандарт МЭК 1131-3 на языки программирования ПЛК.

Большинство современных ПЛК, обладая примерно равными функциональными возможностями, отличаются номенклатурой и количеством входов/выходов [3].

Наиболее распространенной группой в семействе логических контроллеров являются ПЛК малого формата (MicroPLC), к которым относятся MicroLogix 1000 фирмы Allen-Bradley (США), DL 105 фирмы PLC Direct by Koyo (США), LOGO фирмы Siemens (Германия), SYSMAC CPM1 фирмы Omron (Япония) и другие.

ПЛК этой группы характеризуются моноблочной конструкцией, неизменяемой конфигурацией и небольшим (до 100) количеством входов/выходов.

В данном учебном пособии рассмотрены вопросы использования микро-ПЛК SIMATIC S7-200 фирмы Siemens (Германия) с процессором CPU-214 для решения задач автоматизации [4].

В главе 1 приведены состав, основные технические характеристики и коммуникационные возможности семейства ПЛК S7-200. Описаны цикл и режимы работы CPU.

Глава 2 посвящена созданию приложений в среде STEP 7–Micro/WIN 32. Даются подробные указания по работе с редакторами при создании прикладных программ.

Глава 3 является практическим руководством по программированию в среде STEP 7–Micro/WIN 32. Приведены типы данных и способы адресации памяти в CPU, языки программирования и основные элементы для

разработки программ, рекомендации по тестированию и контролю программ, а также устранению ошибок в CPU.

В главе 4 приведены основные правила и рекомендации по решению задач автоматизации, рассматриваются примеры решения конкретных задач с использованием микро-ПЛК S7-200.

Каждая глава сопровождается контрольными вопросами и заданиями, необходимыми для закрепления материала и проверки знаний.

В приложении приведен набор основных операций, используемых при разработке систем автоматизации.

Все команды для программного интерфейса STEP 7–Micro/WIN 32 приведены на английском языке с русским переводом.

Глава 1. ОРГАНИЗАЦИЯ ПЛК S7-200

1.1. Характеристики микроконтроллеров S7-200

Семейство S7–200 включает в себя ряд программируемых логических микроконтроллеров (микро–ПЛК), с помощью которых можно решать широкий спектр задач автоматизации. На рис.1.1 приведена система автоматизации с использованием микро-ПЛК S7-200. Эта система включает центральное устройство S7–200 (CPU), персональный компьютер, программное обеспечение STEP 7–Micro/WIN 32 и соединительный кабель.

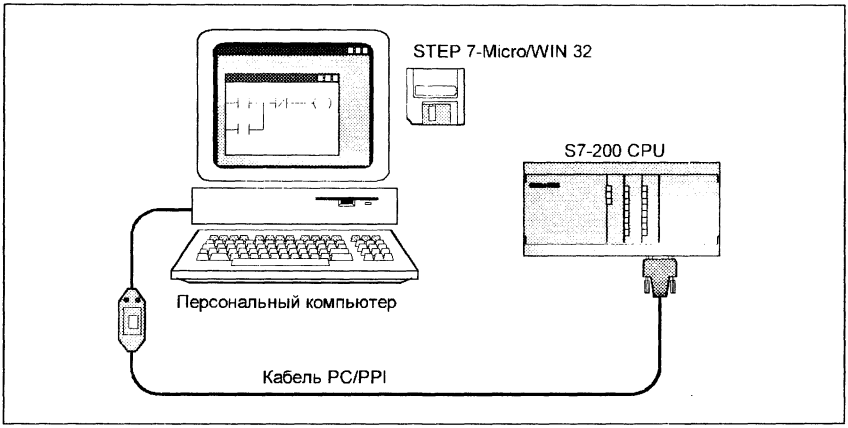


Рис. 1.1

1.1.1. Состав микро-ПЛК S7-200

Центральное устройство S7–200 (CPU).

Центральный модуль S7–200 представляет собой компактное устройство и состоит из центрального процессора (CPU), источника питания и цифровых входов и выходов.

- CPU обрабатывает программу и запоминает данные для задачи автоматизации или процесса.
- Источник питания снабжает током центральное устройство и все подключенные модули расширения.
- Входы и выходы служат для управления автоматизированной системой: входы контролируют сигналы переключателей и датчиков, а выходы управляют исполнительными устройствами.
- Через коммуникационный порт можно подключить к CPU устройство программирования или другие устройства
- Индикаторы состояния предоставляют визуальную информацию о режиме работы CPU (RUN или STOP), текущем состоянии сигналов встроенных входов и выходов и возможных системных ошибках.

Семейство S7–200 включает в себя несколько CPU, основные параметры которых приведены в табл. 1.1.

Таблица 1.1

Характеристика	**CPU–212**	**CPU–214**	**CPU–215**	**CPU–216**
Физический размер	160x80x62 мм	197x80x62 мм	218x80x62 мм	218x80x62 мм
Память				
Программа (EEPROM)	512 слов	2 K слов	4 K слов	4 K слов
Данные пользователя	512 слов	2 K слов	2,5 K слов	2,5 K слов
Внутренние маркеры	128	256	256	256
Модуль памяти	Нет	Да: EEPROM	Да: EEPROM	Да: EEPROM
Мощный конденсатор Модуль батареи (факульт-но)	Тип. 50 час. Нет	Тип. 190 час. Тип. 200 дней	Тип. 190 час. Тип. 200 дней	Тип. 190 час. Тип. 200 дней
Модули ввода и вывода (I/O)				
Встроенные I/O	8 DI/6 DO	14 DI/10 DO	14 DI/10 DO	24DI/16DO
Количество модулей расширения (максим.)	2 модуля	7 модулей	7 модулей	7 модулей
Отображение процесса ввода/ввода	64 DI/64 DO	64 DI/64 DO	64 DI/64 DO	64 DI/64 DO
Аналоговые I/O (расшир.)	16 AI/16 AO	16 AI/16 AO	16 AI/16 AO	16 AI/16 AO
Входной фильтр	Нет	Да	Да	Да
Операции				
Время исполнения булевых операций	1,2 мкс/операцию	0,8 мкс/операцию	0,8 мкс/операцию	0,8 мкс/операцию
Таймеры/Счетчики	64/64	128/128	256/256	256/256
Циклы с FOR/NEXT	Нет	Да	Да	Да
Арифметика с фиксированной точкой	Да	Да	Да	Да
Арифметика с плавающей точкой	Нет	Да	Да	Да
PID	Нет	Нет	Да	Да
Дополнительные функциональные возможности				
Быстрые счетчики	1 SW	1 SW, 2 HW	1 SW, 2 HW	1 SW, 2 HW
Аналоговые потенциометры	1	2	2	2
Импульсные выходы	Нет	2	2	2
Коммуникационные прерывания	1 Передача/ 1 Прием	1 Передача/ 1 Прием	1 Передача/ 2 Прием	2 Передача/ 4 Прием
Прерывания, управляемые временем	1	2	2	2
Входы аппаратных прерываний	1	4	4	4
Часы реального времени	Нет	Да	Да	Да
Связь				
Количество портов	1 (RS–485)	1 (RS–485)	2 (RS–485)	2 (RS–485)
Поддерживаемые протоколы порт 0: порт 1:	 PPI, своб.прог. –/–	 PPI, своб.прог –/–	 PPI, своб.прогр. DP	 PPI, своб.прогр. PPI, своб.прогр.
Точка-точка	Только Slave	Да	Да	Да

Модули расширения.

Центральное устройство S7–200 имеет определенное количество встроенных входов и выходов. Добавление модуля расширения предоставляет дополнительные (цифровые и аналоговые) входы и выходы. На рис. 1.2 показан модуль расширения и шинный соединитель, с помощью которого модуль расширения подключается к центральному устройству.

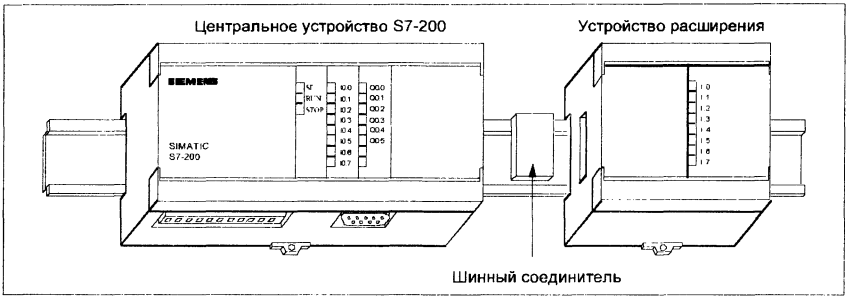


Рис. 1.2

1.1.2. Коммуникационные возможности CPU S7-200

Из табл. 1.2 видно, что CPU S7–200 поддерживают различные коммуникационные возможности.

Системный протокол для S7–200 называется интерфейсом «точка-к-точке» (PPI). Он базируется на архитектуре коммуникаций в семиуровневой модели взаимодействия открытых систем (OSI). Протокол PPI является протоколом Master/Slave («главный/ подчиненный») реализованным на основе маркерной шины (token bus) с уровнями сигналов RS–485. Определение маркерной шины соответствует стандарту PROFIBUS (Process Field Bus) согласно EN 50170. Скорость передачи данных может устанавливаться равной от 9600 бод до 19200 бод.

Протокол PPI поддерживает соединения как между одним Master-устройством и несколькими Slave-устройствами, так и между несколькими Master-устройствами и несколькими Slave-устройствами. Протокол PPI является знакоориентированным протоколом, который использует кадры, состоящие из следующих одиннадцати битов: стартовый бит, восемь битов данных, бит проверки четности и стоп-бит. Блоки передачи данных в коммуникации (за исключением односимвольного квитирования) включают в себя особые символы начала и остановки, абонентские адреса источника и получателя, длину блока передачи данных и символ контрольной суммы для обеспечения целостности данных.

CPU S7–200 являются Slave-устройствами, реагирующими на устройство программирования, интерфейс оператора или другой CPU. CPU не могут порождать сообщения, если они эксплуатируются в качестве Slave-

устройств PPI. CPU в режиме RUN могут получить маркер путем разблокировки режима PPI–Master. Не все CPU можно эксплуатировать в режиме PPI–Master. После активизации режима PPI–Master можно с помощью операций чтения из сети (NETR) и записи в сеть (NETW) передавать сообщения другим CPU.

Таблица 1.2

CPU	Порт	PPI–Slave	PPI–Master	Стандарт DP	Свобод. програм. связь	Скорость передачи данных
CPU 212	Порт 0	Да	Нет	Нет	Да	9600 бод
CPU 214	Порт 0	Да	Да	Нет	Да	9600 бод
CPU 215	Порт 0	Да	Да	Нет	Да	9600 бод, 19200 бод
	Порт DP	Нет	Нет	Да	Нет	9600 бод, 19200 бод, 93750 бод, 187500 бод, 500 кбод, 1 мбод, 1,5 мбод, 3 мбод, 6 мбод, 12 мбод
CPU 216	Порт 0	Да	Да	Нет	Да	9600 бод, 19200 бод
	Порт 1	Да	Да	Нет	Да	9600 бод, 19200 бод

На рис. 1.3 показан пример сети MPI с ведущими (Master) и подчиненными (Slave) устройствами.

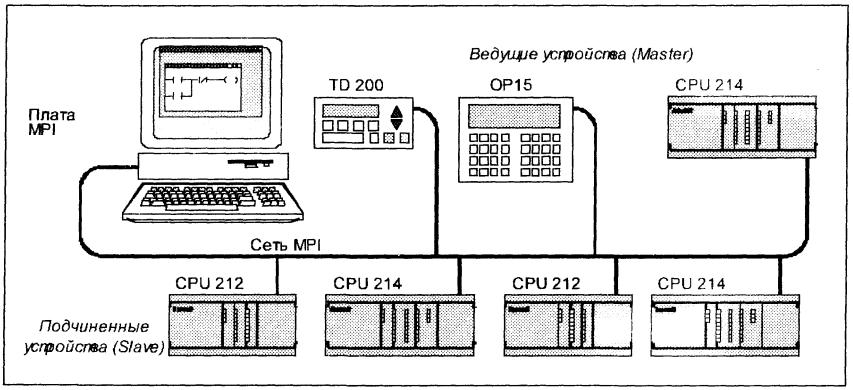


Рис. 1.3

1.2. Управление входами и выходами ПЛК

1.2.1. Адресация входов и выходов

Встроенные входы и выходы в центральном устройстве имеют фиксированные адреса. Адреса входов и выходов в модуле расширения определяются видом входов и выходов, а в случае нескольких модулей одинакового типа – также расположением модуля. В частности, модуль вывода не влияет на адреса входов в модуле ввода и наоборот. Адреса входов и выходов аналоговых и цифровых модулей также не зависят друг от друга.

Для цифровых модулей расширения предусмотрены разделы по восемь битов (байт) в области отображения процесса. Если в модуле имеется в наличии физический вход или выход не для каждого бита зарезервированного байта, то свободные биты теряются и не могут использоваться никаким следующим модулем расширения данного CPU. Свободные биты зарезервированных байтов модулей вывода могут использоваться как внутренние маркеры. В модулях ввода свободные биты сбрасываются в нуль в каждом цикле актуализации входов и поэтому не могут использоваться как внутренние маркеры.

Входы и выходы аналоговых модулей расширения всегда размещаются с двойным шагом. Если в модуле не для каждого из этих входов и выходов имеется в наличии физический вход или выход, то эти входы и выходы теряются и не могут сопоставляться никакому следующему модулю расширения данного CPU. Так как аналоговые входы и выходы не имеют отображения в памяти, то свободные аналоговые входы и выходы не могут использоваться. При выполнении операции доступ к аналоговым входам и выходам всегда происходит непосредственно.

Примеры встроенных и дополнительных входов и выходов

В табл.1.3 показаны примеры адресов CPU-214, из которых видно, как различные конфигурации аппаратных средств влияют на адреса входов и выходов. Обратите внимание на то, что некоторые из конфигураций содержат пробелы в последовательности адресов, которые не могут использоваться программой, в то время как другие свободные адреса входов и выходов можно использовать как внутренние маркеры.

Таблица 1.3

	Модуль 0	Модуль 1	Модуль 2	Модуль 3	Модуль 4
CPU-214	**4 вх./4 вых.**	**8 вх.**	**3AI/1AQ**	**8 вых.**	**3AI/1AQ**
I0.0 Q0.0	I2.0 Q2.0	I3.0	AIW0 AQW0	Q3.0	AIW8 AQW4
I0.1 Q0.1	I2.1 Q2.1	I3.1	AIW2	Q3.1	AIW10
I0.2 Q0.2	I2.2 Q2.2	I3.2	AIW4	Q3.2	AIW12
I0.3 Q0.3	I2.3 Q2.3	I3.3		Q3.3	
I0.4 Q0.4		I3.4		Q3.4	
I0.5 Q0.5		I3.5		Q3.5	
I0.6 Q0.6		I3.6		Q3.6	
I0.7 Q0.7		I3.7		Q3.7	
I1.1 Q1.0					
I1.2 Q1.1					
I1.3					
I1.4					
I1.5					
Отображение процесса на вх./вых., которые можно использовать как внутренние маркеры:					
Q1.2	Q2.4	I4.0		Q4.0	
Q1.3	Q2.5	.		.	
Q1.4	Q2.6	.		.	
Q1.5	Q2.7	.		.	
Q1.6		I7.7		Q7.7	
Q1.7					

Отображение процесса на вх./вых., которые нельзя использовать как внутренние маркеры:					
I1.6 I1.7	I2.4 I2.5 I2.6 I2.7		AIW6 AQW2		AIW14 AQW6

1.2.2. Конфигурирование входных фильтров для подавления помех

Можно выбрать для CPU S7–200 фильтр ввода, который определяет для физических входов время задержки (регулируемое в диапазоне от 0,2 мс до 8,7 мс). Это время прибавляется к обычному времени ответа для групп из четырех входов (рис.1.4). Время задержки служит для того, чтобы отфильтровать в соединительной проводке входа помехи, которые могут вызвать непреднамеренные изменения состояний сигналов на входах.

Входной фильтр является частью данных конфигурации CPU, загружаемых в память CPU и хранимых там.

Выберите пункт меню **CPU → Configure...** [CPU → Конфигурирование] и щелкните мышью на вкладке «**Input Filters**» [«Входные фильтры»]. Установите времена задержки для входных фильтров.

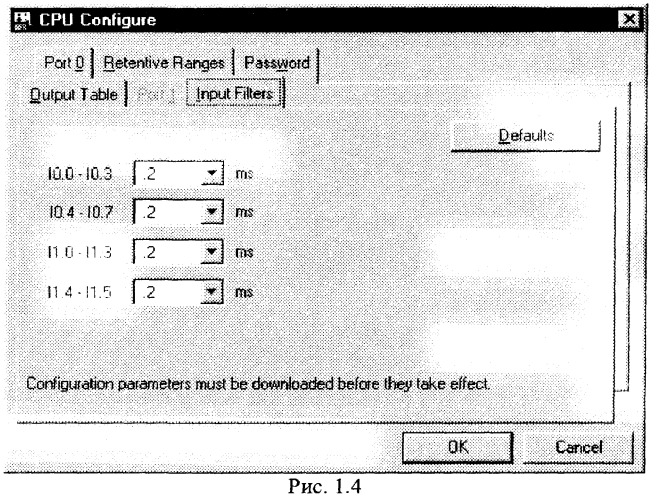


Рис. 1.4

1.2.3. Конфигурирование состояний сигналов выходов

С помощью CPU S7–200 можно устанавливать состояния сигналов цифровых выходов на определенные значения при переходе в режим работы STOP, либо оставлять выходы точно в том состоянии, в котором они находились при переходе в STOP.

Установки выходов являются частью данных конфигурации CPU для системы, которые загружаются в память CPU и хранятся там.

Конфигурирование выходных значений возможно только для цифровых выходов. Аналоговые выходы при переходе в режим работы STOP замораживаются. Это происходит, так как программа ответственна за актуализацию аналоговых выходов. Актуализация аналоговых входов и выходов не является системной функцией CPU. Для аналоговых входов и выходов отображение в памяти CPU не хранится.

Выберите пункт меню **CPU → Configure...** [CPU → Конфигурирование] и щелкните мышью на вкладке «**Output Table**» [«Установки выходов»]. В этом диалоговом окне (рис.1.5) есть следующие две возможности:

1. Если хотите замораживать выходы в их последнем состоянии, то актуализируйте управляющий блок «**Freeze Outputs**» [«Заморозить выходы»] и подтвердите посредством «OK».

2. Если хотите копировать в выходы определенные значения, то укажите установки для выходов. Щелкните мышью на соответствующей кнопке для каждого выхода, который хотите устанавливать в «1» при переходе в режим STOP. Затем подтвердите установки посредством «OK».

По умолчанию для всех выходов установлено состояние «0».

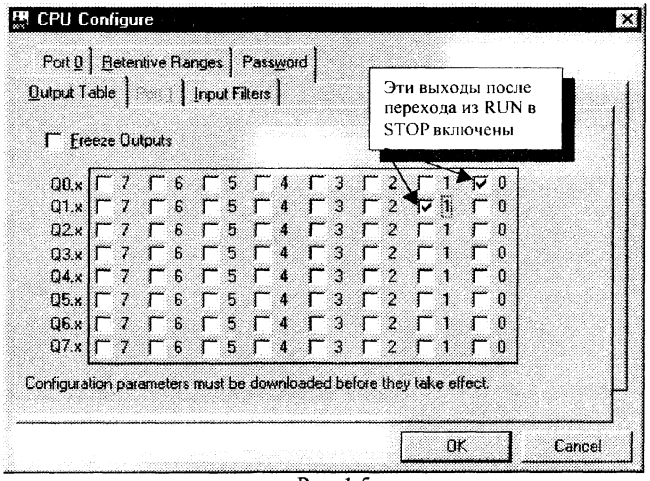


Рис. 1.5

1.2.4. Быстрые входы и выходы

В CPU S7–200 есть быстрые входы и выходы, с помощью которых можно управлять быстрыми событиями.

Быстрые счетчики.

Быстрые счетчики подсчитывают быстрые события, которыми невозможно управлять при скорости цикла CPU S7–200. CPU S7–200 поддерживает один быстрый программный и два быстрых аппаратных счетчика (в зависимости от CPU):

- HSC0 является программным реверсивным счетчиком, поддерживающим тактовый вход. Программа управляет направлением счета (прямое или обратное) через бит, управляющий направлением. Максимальная скорость счета этого счетчика составляет 2 кГц.
- HSC1 и HSC2 являются универсальными аппаратными счетчиками, которые можно конфигурировать на один из двенадцати различных видов операций. Максимальная скорость счета счетчиков HSC1 и HSC2 зависит от CPU.

Каждый счетчик имеет специальные входы, поддерживающие такие функции, как тактовый генератор, управление направлением, сброс и запуск. В счетчиках можно выбирать однократную или четырехкратную скорость счета. HSC1 и HSC2 полностью не зависят друг от друга и не влияют на другие быстрые операции. Оба счетчика работают с максимальной скоростью, не оказывая отрицательного воздействия друг на друга.

Импульсные выходы.

CPU S7–200 поддерживает быстрые импульсные выходы. В этих CPU выходы Q0.0 и Q0.1 либо порождают импульсные последовательности, либо управляют широтно-импульсной модуляцией (ШИМ).

Функция импульсной последовательности представляет выходную последовательность прямоугольных импульсов (относительная длительность включения 50%) с определенным количеством импульсов и заданным периодом. Количество импульсов может находиться в диапазоне от 1 до 4.294.967.295. Период может задаваться в микросекундах (от 250 до 65.535) или в миллисекундах (от 2 до 65.535). Нечетное количество микро- или миллисекунд (напр., 75 мс) вызывает искажение относительной длительности включения импульсов.

Функция ШИМ представляет фиксированный период импульсов с переменной длительностью включения. Период и длительность импульсов могут задаваться в микро- или миллисекундах. Период находится в диапазоне от 250 до 65.535 микросекунд или в диапазоне от 2 до 65.535 миллисекунд. Длительность импульсов находится в диапазоне от 0 до 65.535 микросекунд или в диапазоне от 0 до 65.535 миллисекунд. Если длительность и период импульсов равны, то относительная длительность включения составляет 100%, и выход включен постоянно. Если длительность импульсов равна нулю, то относительная длительность включения составляет 0%, и выход выключается.

1.2.5. Аналоговые потенциометры

CPU S7–200 имеет один или два аналоговых потенциометра (под откидной крышкой CPU). С помощью этих потенциометров можно увеличивать и уменьшать значения, записанные в байтах специальных маркеров (SMB28 и SMB29). Эти защищенные от записи значения могут служить в программе для ряда функций, например, при актуализации текущих значе-

ний таймеров и счетчиков, при вводе или изменении предварительно установленных значений или при установке граничных значений.

SMB28 хранит цифровое значение, представляющее настройку аналогового потенциометра 0. SMB29 хранит цифровое значение, представляющее настройку аналогового потенциометра 1. Аналоговые потенциометры имеют номинальный диапазон от 0 до 255 и гарантированный диапазон от 10 до 200.

На рис.1.6 показан пример программы, использующей аналоговый потенциометр.

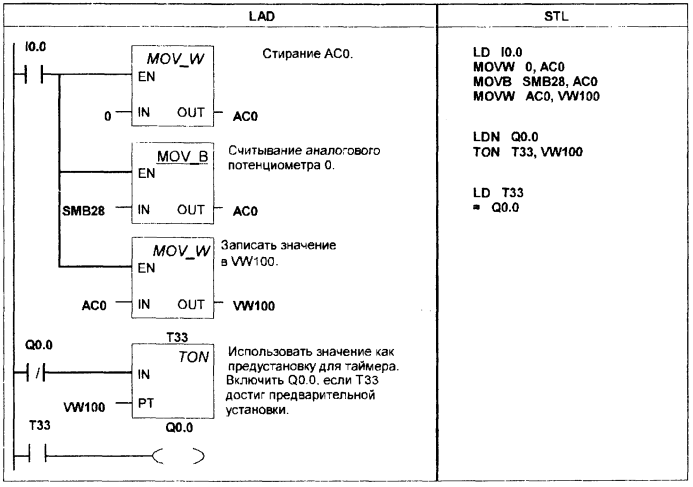


Рис. 1.6

1.3. Цикл CPU

CPU S7–200 обрабатывает программу циклически. Цикл состоит из нескольких шагов, которые выполняются регулярно и в строгой последовательности (рис.1.7):

- считывание входов,
- обработка программы,
- обработка коммуникационных запросов,
- проведение самодиагностики в CPU,
- запись на выходы.

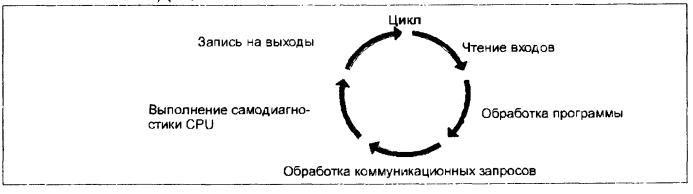


Рис. 1.7

1.3.1. Считывание цифровых входов

В начале цикла считываются текущие значения цифровых входов и затем записываются в область отображения процесса на входах.

В области отображения процесса на входах для CPU предусмотрены разделы по восемь битов (один байт). Если CPU или модуль расширения предоставляет в распоряжение физический вход не для каждого бита зарезервированного байта, то нельзя назначать такие биты последующим модулям в цепи ввода/вывода и использовать их в программе. Свободные входы в области отображения процесса обнуляются CPU в начале каждого цикла. Однако если CPU может дополняться несколькими модулями расширения и эта возможность не исчерпана полностью, то можно использовать свободные входные биты, предусмотренные для модулей расширения, в качестве дополнительных маркеров.

CPU автоматически не активизирует аналоговые входы в качестве части цикла, а также не записывает в память отображение для аналоговых входов. Доступ к аналоговым входам производят непосредственно через программу.

1.3.2. Обработка программы

В этой фазе цикла CPU обрабатывает программу от первой до последней операции. Во время обработки главной программы или программы обработки прерываний можно прямо обращаться к входам и выходам и таким образом управлять ими.

1.3.3. Обработка коммуникационных запросов

В этой фазе цикла CPU обрабатывает все сообщения, принятые через коммуникационный порт.

1.3.4. Проведение самодиагностики в CPU

В этой фазе цикла CPU проверяет программы в ПЗУ, память программ и состояние модулей расширения.

1.3.5. Запись на цифровые выходы

В конце цикла значения из области отображения процесса на выходах записываются на цифровые выходы.

В области отображения процесса на выходах для CPU предусмотрены разделы по восемь битов (один байт). Если CPU или модуль расширения предоставляет в распоряжение физический выход не для каждого бита зарезервированного байта, то нельзя назначать такие биты последующим модулям в цепи ввода/вывода. Однако можно использовать свободные биты в области отображения процесса на выходах в качестве дополнительных внутренних маркеров.

CPU автоматически не активизирует аналоговые выходы в качестве части цикла, а также не записывает в память отображение для аналоговых выходов. Доступ к аналоговым выходам производят непосредственно через программу.

1.3.6. Прерывание цикла

Если в программе используют прерывания, то программы обработки прерываний, поставленные в соответствие событиям прерываний, запоминаются как часть главной программы. Однако программы обработки прерываний не обрабатываются как составная часть цикла, а обрабатываются только тогда, когда появляется событие прерывания (это возможно в любой точке цикла). CPU обрабатывает разблокированные прерывания асинхронно по отношению к циклу и выполняет программу обработки прерываний, когда появляется соответствующее событие прерывания. Обработка прерываний происходит в порядке их появления и в соответствии с их приоритетом.

1.3.7. Отображение процесса на входах и выходах

Во время обработки программы доступ к входам и выходам происходит не прямо, а через соответствующие области отображения процесса. Имеются три важных основания для существования областей отображения процесса:

- Система в начале цикла опрашивает входы. Благодаря этому, значения этих входов синхронизируются и «замораживаются» на период обработки программы. После обработки программы через область отображения процесса актуализируются выходы. Это оказывает стабилизирующее воздействие на систему.
- Программа может обращаться к области отображения процесса гораздо быстрее, чем непосредственно к входам и выходам. Это ускоряет обработку программы.
- Входы и выходы являются битовыми объектами, доступ к которым должен производиться в битовом формате. К областям же отображения процесса можно обращаться в формате бита, байта, слова и двойного слова. Поэтому области отображения процесса обеспечивают дополнительную гибкость.

Дополнительным преимуществом является то, что области отображения процесса достаточно велики для того, чтобы обрабатывать максимальное количество входов и выходов. Так как реальная система состоит из входов и выходов, то в области отображения процесса всегда есть неиспользуемые адреса, которые можете использовать как дополнительные внутренние маркеры.

1.3.8. Прямое управление входами и выходами

С помощью операций прямого управления входами и выходами можно прямо обращаться к входу или выходу, хотя нормально в качестве источника и цели доступа к входам и выходам используются области отображения процесса. Если Вы обращаетесь прямо к входу, то соответствующий адрес в области отображения процесса на входах не изменяется. Если Вы обращаетесь прямо к выходу, то одновременно активизируется соответствующий адрес в области отображения процесса на выходах.

1.4. Режимы работы CPU

CPU S7–200 имеет два режима работы:

1. STOP: CPU не обрабатывает программу. В режиме STOP можно загружать в CPU программу и конфигурировать CPU.
2. RUN: CPU обрабатывает программу. В режиме RUN нельзя загружать в CPU программу и конфигурировать CPU.

Индикация состояния на лицевой панели CPU указывает текущий режим работы. Если Вы хотите загрузить программу в программную память, то необходимо перевести CPU в состояние STOP.

1.4.1. Установка режима работы переключателем режимов

С помощью переключателя режимов работы (находится под защитной крышкой на CPU) можно вручную установить режим работы CPU:

- если переключатель режимов работы установлен в положение TERM, то программное обеспечение (STEP 7–Micro/WIN 32) может управлять режимами работы CPU;
- если переключатель режимов работы установить в положение STOP, обработка программы прекращается;
- если переключатель режимов работы установить в положение RUN, включается обработка программы.

Если переключатель режимов работы находится в положении STOP или TERM и прерывается подача напряжения питания, то при восстановлении напряжения питания CPU автоматически переходит в режим STOP. Если в тот момент, когда прерывается подача напряжения питания, переключатель режимов работы находится в положении RUN, то при восстановлении напряжения питания CPU автоматически переходит снова в режим работы RUN.

1.4.2. Установка режима работы с помощью STEP 7–Micro/WIN 32

Для того чтобы программное обеспечение могло управлять режимом работы (рис.1.8), необходимо перевести переключатель режимов работы на CPU в положение TERM или RUN.

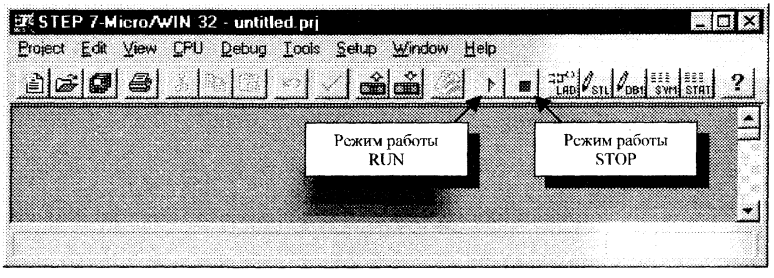


Рис. 1.8

1.4.3. Установка режима работы с помощью программы

Можно в программе записать операцию, переводящую CPU в режим работы STOP. Таким способом можно прерывать обработку программы в зависимости от логики программы.

1.5. Установка пароля для CPU

Все варианты CPU S7–200 предоставляют парольную защиту для ограничения доступа к определенным функциям CPU. Благодаря установке пароля доступ к определенным функциям и памяти CPU имеют только уполномоченные лица. Если парольная защита установлена, то CPU запрещает неуполномоченным лицам все операции, ограниченные в конфигурации пароля.

1.5.1. Уровни защиты CPU

CPU S7–200 предоставляют три разных уровня защиты с различными ограничениями доступа к функциям CPU (табл.1.4). Каждый уровень защиты также допускает неограниченный доступ к определенным функциям без ввода пароля. При всех трех уровнях защиты Вы имеете доступ ко всем функциям CPU, если вводите правильный пароль. По умолчанию для CPU S7–200 установлен уровень защиты 1 (ограничений нет).

Если вводят сетевой пароль, то этот пароль не оказывает влияния на парольную защиту CPU. Если некоторый пользователь наделен правом доступа к защищенным функциям CPU, то из этого не следует, что другие пользователи тоже наделены правом доступа к этим функциям. Всегда только один пользователь имеет неограниченный доступ к CPU.

Таблица 1.4

Функция	Уровень защиты 1	Уровень защиты 2	Уровень защиты 3
Чтение и запись данных пользователя	Не ограничены	Не ограничены	Не ограничены
Запуск, останов и новый запуск CPU			
Чтение и установка часов реального времени			
Чтение принудительно установленных данных CPU			Нужен пароль
Загрузка программы пользователя, данных и конфигурации из CPU			
Загрузка в CPU		Нужен пароль	
Стирание программы пользователя, данных и конфигурации *			
Принудительная установка данных и выполнение определенного количества циклов			
Копирование в модуль памяти			
* Защита от стирания может быть преодолена паролем «clearplc».			

1.5.2. Установка пароля

Для установки пароля выберите команду меню **CPU → Configure** [CPU → Конфигурирование], а затем вкладку «**Password**» [«Пароль»] (рис.1.9). Задайте желаемый уровень защиты и подтвердите пароль.

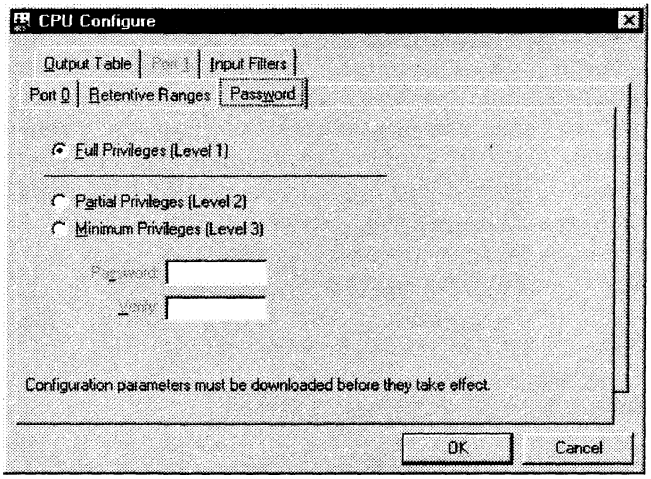


Рис. 1.9

1.5.3. Мероприятия в случае забытого пароля

Если пользователь забыл пароль, то необходимо произвести общее стирание памяти CPU и снова загрузить в CPU свою программу. При общем стирании памяти CPU сначала переводится в состояние STOP, а затем переустанавливается на значения по умолчанию, предварительно установленные предприятием-изготовителем, за исключением адреса абонента и часов реального времени.

Для общего стирания CPU выберите команду меню **CPU → Clear** [CPU → Общее стирание]. В ответ на это отображается диалоговое окно «**Clear**» [«Общее стирание»]. Выберите опцию «**All**» [«Все»] и подтвердите ввод, щелкнув на «OK». При этом на экране отображается диалоговое окно, в котором запрашивается парольная авторизация. Если ввести пароль «**clearplc**» (общее стирание ПЛК), то можно произвести общее стирание всей памяти CPU.

При общем стирании программа в модуле памяти не стирается. Так как в модуле памяти наряду с программой записан и соответствующий пароль, то необходимо также перепрограммировать модуль памяти, чтобы стереть забытый пароль.

При общем стирании CPU выходы выключаются (аналоговые выходы «замораживаются» с определенным значением).

Если при общем стирании CPU S7–200 подключен к оборудованию, то изменения в состоянии выходов могут быть переданы на оборудование. Если изменить «безопасные состояния» выходов, предварительно установленные предприятием, то изменения состояний выходов могут вызвать неожиданные реакции со стороны оборудования, что может привести к гибели или серьезным травмам персонала и/или к повреждению оборудования.

Поэтому примите все необходимые меры безопасности и обеспечьте, чтоб процесс находился в безопасном состоянии, прежде чем производить общее стирание CPU.

1.6. Контрольные вопросы и задания

1.1 Характеристики микроконтроллера S7-200.

1) Из чего состоит система автоматизации с использованием микроконтроллера S7-200?

2) Какие устройства входят в микроконтроллер S7-200 и для него они необходимы?

3) Что представляет из себя протокол PPI?

1.2 Управление входами и выходами микроконтроллера S7-200.

1) Для чего необходимо конфигурирование входных фильтров?

2) Как осуществить конфигурирование входов и выходов?

3) Какие еще существуют входы и выходы?

4) Для чего необходимы специальные маркеры SM 28 и SM 29? Что в них хранится?

1.3 Цикл CPU.

1) Какие операции входят в цикл? (Что именно происходит в каждой операции?).

2) Является ли программа обработки прерывания составной частью цикла? Как, по отношению к циклу, идет разблокировка прерывания?

1.4 Режимы работы CPU.

1) Какие существую режимы работы CPU?

2) Что означают режимы TERM, STOP, RUN при установке вручную режимов CPU?

3) Что происходит при отключении напряжения в сети в каждом из режимов?

1.5 Установка пароля для CPU.

1) Чем отличается друг от друга уровни защиты CPU (укажите разграничения по функциям)?

Глава 2. СОЗДАНИЕ ПРИЛОЖЕНИЙ В СРЕДЕ STEP 7–Micro/WIN 32

2.1. Создание и сохранение проекта

Прежде чем разрабатывать программу необходимо создать или открыть проект. При создании нового проекта STEP 7–Micro/WIN 32 открывает следующие редакторы:

- редактор релейно-контактных схем или списков команд (в зависимости от того, какой редактор установлен);
- редактор для обработки блоков данных;
- редактор для обработки таблиц состояний/принудительного задания;
- редактор для обработки таблиц символов.

2.1.1. Создание нового проекта

Новый проект создают через меню **Project** (рис.2.1). Для этого выберите команду меню **Project → New...** [Проект → Новый...]. В ответ на это открывается диалоговое окно «**CPU**". Если выбрать CPU в раскрывающемся списковом блоке, то программное обеспечение отображает только те варианты, которые доступны для выбранного CPU. Если выбрать «**None**» [«Нет»], то программа не содержит ограничений, специфических для CPU. При загрузке программы в CPU, последний проверяет, используются ли функции, которые ему не доступны. В частности, если программа содержит операцию, не поддерживаемую соответствующим CPU, то программа отвергается.

STEP 7–Micro/WIN 32 не проверяет область параметров. Например, можно ввести VB9999 в качестве параметра LAD–операции, хотя этот параметр и является недействительным.

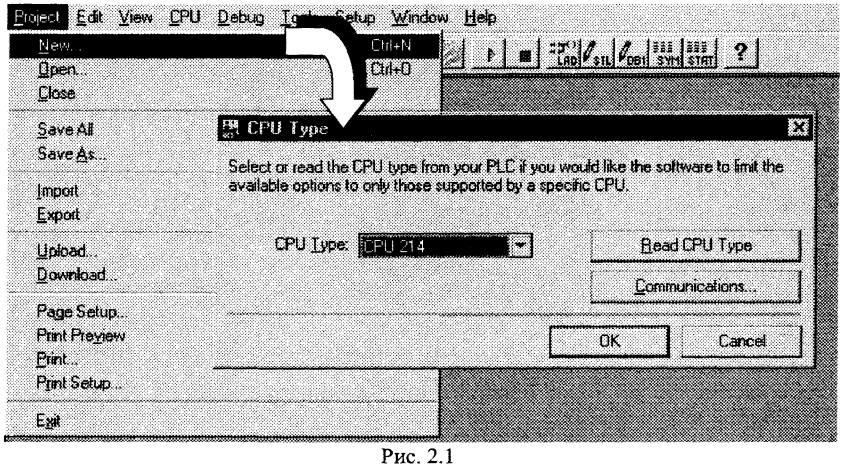


Рис. 2.1

Пояснения к рисунку. Меню Project [Проект]: New – Новый; Open – Открыть; Close – Закрыть; Save All – Сохранить; Save As – Сохранить как...; Import – Импорт; Export – Экспорт; Upload – Загрузить в программатор; Download – Загрузить из программатора; Print – Печатать; Print Setup – Настроить принтер; Exit – Закончить.

2.1.2. Сохранение проекта

Для сохранения всех компонент проекта выберите команду меню **Project → Save All** [Проект → Сохранить] или щелкните мышью на кнопке  . Можно сохранить проект также под другим именем или в другом каталоге. Для этого выберите команду меню **Project → Save As** [Проект → Сохранить как].

2.2. Создание программы

В STEP 7–Micro/WIN 32 можно разрабатывать прикладную программу (OB1) с помощью редактора релейно-контактных схем (LAD) или редактора списков команд (STL).

2.2.1. Ввод программы в форме релейно-контактных схем

В редакторе LAD программу пишут с помощью графических символов (рис.2.2). Панель инструментов содержит некоторые из наиболее часто используемых элементов LAD, которые можно вводить в программу. Первый раскрывающийся списковый блок (самый левый) содержит семейства операций. Доступ к этим семействам осуществляется щелчком мышью или нажатием клавиши F2. Если семейство выбрано, то второй раскрывающийся списковый блок содержит операции соответствующего семейства. Если нужно отобразить список всех доступных операций в алфавитном порядке, то нажмите клавишу F9 или выбрать «**All Categories**» [«Все Операции»].

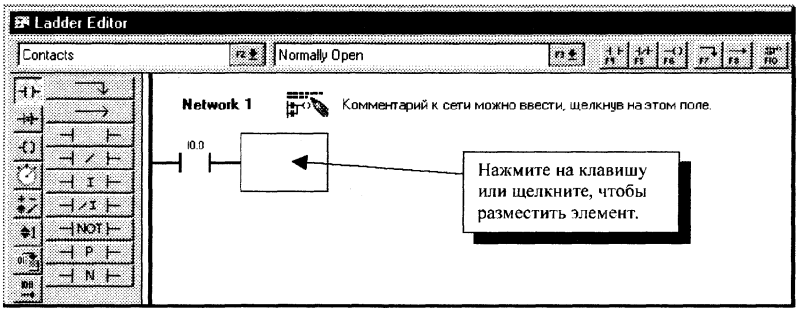


Рис. 2.2

Для каждого сегмента (Network) существуют два комментария:

- однострочные комментарии к сегменту всегда видны в редакторе LAD. Доступ к этим комментариям можно получить, щелкнув мышью по комментарию в произвольном месте;
- комментарии к сегменту, занимающие несколько строк, редактируют после двойного щелчка мышью по номеру сегмента. Многострочные комментарии к сегменту отображаются только в одном диалоговом окне, но в распечатке присутствуют полностью.

Для ввода программы действуйте следующим образом:

1. Для ввода заголовка программы выберите команду меню **Edit → Program Title** [Редактирование → Заголовок программы].

2. Если хотите вводить элементы LAD, щелкните мышью на соответствующей кнопке или выберите элемент из списка операций. Затем нажмите клавишу ввода или дважды щелкните мышью в этом поле.

3. Введите операнды или параметры в текстовые поля и нажмите клавишу ввода.

2.2.2. Ввод программы в форме списка команд

В случае редактора STL речь идет о текстовом редакторе, предоставляющем определенную степень гибкости при вводе команд программы в силу возможности свободного выбора формата. На рис.2.3 показан пример программы в форме списка команд.

STL Editor

```
//Программа Транспортер

NETWORK 1       // Запуск двигателя:
LD      I0.0    // Если включен I0.0
AN      I0.1    // и не включен I0.1,
S       Q0.0, 1 // то включить двигатель транспортера.

NETWORK 2       // Аварийное выключение транспортера:
LD      I0.1    // Если нажата аварийная кнопка 1
O       I0.3    // или нажата аварийная кнопка 2,
R       Q0.0, 1 // то выключить двигатель транспортера

NETWORK 3       // Конец программы
MEND
```

Рис. 2.3

При вводе программы в форме STL необходимо соблюдать следующие указания:

- для возможности отображения программы на STL в форме LAD ее нужно разделить на отдельные сегменты, вводя ключевое слово NETWORK. (Номера сегментов создаются автоматически при компиляции или загрузке программы);

- начинайте каждый комментарий двойной косой чертой (//). Каждая дополнительная строка комментария должна начинаться двойной косой чертой;
- заканчивайте каждую строку возвратом каретки;
- отделяйте каждую операцию от адреса или параметра пробелом или клавишей TAB;
- не вводите пробел между обозначением области памяти и адресом (например, вводите **I0.0**, а не **I 0.0**);
- отделяйте каждый операнд внутри команды запятой, символом пробела или клавишей TAB;
- вводите символические имена в апострофах. Например, если таблица символов содержит символическое имя Start1 для адреса I0.0, то вводите команду следующим образом: **LD «Start1»**.

2.2.3. Компиляция программы

После того, как введены один или несколько сегментов, можно проверить синтаксис программы. Для этого выберите команду меню **CPU → Compile** [CPU → Компиляция] или щелкните мышью на кнопке для компиляции:  .

2.2.4. Загрузка программы в CPU

Если программа введена полностью, то можно загрузить проект в CPU. Для этого выберите команду меню **Projekt → Download...** [Проект → Загрузка из PG...] или щелкните мышью в главном окне на кнопке:  .

Вслед за этим открывается диалоговое окно, в котором можно указать компоненты проекта, которые нужно загрузить в CPU (рис.2.4).

Рис. 2.4

- Кодовый блок (OB1) содержит программу, которая должна обрабатываться CPU.
- Блок данных (DB1) содержит значения, которые должны использоваться программой для инициализации.
- Конфигурация CPU (CFG) содержит информацию для настройки системы. Она включает в себя параметры связи, сохраняемые области, настройки фильтров ввода, пароли и настройки выходов.

Подтвердите ввод посредством «OK» или нажмите клавишу ввода.

2.2.5. Отображение программы в форме LAD и STL

Программу можно просматривать в виде релейно-контактных схем или списка команд. Для этого выберите одну из команд меню **View → STL** [Вид → STL] или **View → Ladder** [Вид → LAD] (рис.2.5).

Если при просмотре переходить из STL в LAD, а затем опять в STL, то можно обнаружить следующие изменения в представлении на STL:

- малые буквы в командах и адресах становятся большими буквами;
- символы пробела между операциями и адресами заменяются символами табуляции.

Можно выполнить такое же преобразование команд STL, выбрав в активном редакторе STL команду меню **CPU → Compile** [CPU → Компиляция].

Определенные последовательности команд STL не могут отображаться в LAD. В этом случае сообщение «**Invalid**» [«Недопустимо»] отмечает части программы, которые не могут представляться в LAD.

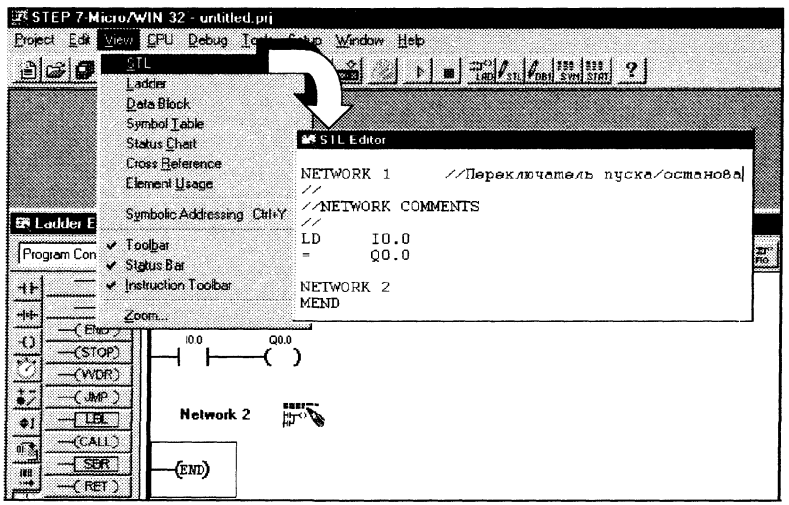


Рис. 2.5

2.3. Создание блока данных

С помощью редактора блоков данных можно предварительно определять или инициализировать переменные, которые должны использоваться программой. Создание блока данных не является безусловной необходимостью.

Редактор блоков данных по умолчанию отображается в виде пиктограммы окна у нижнего края главного окна. Чтобы вызвать редактор блоков данных, нужно дважды щелкнуть на этой пиктограмме или на кнопке «Восстановить/Минимизировать» на этой пиктограмме.

Ввод значений для блоков данных.

Редактор блоков данных является текстовым редактором, предоставляющим определенную степень гибкости при вводе значений блоков данных в силу возможности выбора форматов.

При создании блоков данных примите во внимание следующие указания:

- задавайте в первом столбце строки размер данных и начальный адрес каждого значения для памяти переменных;
- начальный адрес и значения данных должны отделяться друг от друга пробелом или клавишей TAB.

На рис.2.6 показан пример блока данных с комментариями, описывающими каждый элемент.

```
Data Block Editor
VB0  255          //хранится как байт, начиная с VB0
VW2  256          //слово, начинающееся с VW2
VD4  700.50       //вещественное число (32 бита), начинается с  VD4
VB8  -35          //байтовая величина, хранится, начиная с VB8
VW10 16#0A        //слово в 16-ричном формате, хранится, начиная с VW10
VD14 123456       //двойное слово, начиная с VD14
VW20 2 4 8 16     //таблица из слов, начинается с VW20
     -2 64 12 56 //(Обратите внимание, что значения данных второй и
     85 10 20 40 //третьей строки не могут начинаться в столбце 1.)
VB45 'Up'         //строка символов ASCII (2 байта), начинается с  VB45
V50'Это новое сообщение из 34 символов'
                  //строка символов ASCII с началом в VB50 (до VB83)
VW84  65535       //слово с началом на следующем свободном адресе VW84
```

Рис. 2.6

STEP 7–Micro/WIN 32 с помощью первого столбца в каждой строке редактора блоков данных устанавливает начальный адрес для запоминания значения в блоке данных. Если ввести в столбце 1 число, то это число оценивается для последующих данных как начальный адрес в памяти переменных. Если Вы рассчитывали на то, что число в колонке 1 должно задавать значение данных, а не адрес, то может случиться так, что вследствие этого будут непреднамеренно переписаны новыми данными те данные, которые введены в блок данных раньше. Так как STEP 7–Micro/WIN 32 не проверяет, пересекаются ли начальные адреса, то может случиться так, что

данные запоминаются по адресам, отличающимся от предусмотренных, или переписываются уже имеющиеся данные.

Если загрузить в CPU блок данных, в котором адресуются ложные данные, то это может привести к непредусмотренным последствиям в процессе.

Всегда задавайте размер и адрес, например, VB100, чтобы гарантировать запись данных в память переменных по правильным адресам. Также всегда тщательно проверяйте, чтобы по ошибке не записали значение данных в столбец 1.

Табл.2.1 показывает примеры способов записи для ввода значений блоков данных.

Таблица 2.1

Тип данных	**Пример**
Шестнадцатиричное число	16#AB
Целое число (десятичное)	10 *или* 20
Целое число со знаком (десятичное)	–10 *или* +50
Действительное число (число с плавающей точкой): используйте точку («.»), а не запятую («,»)	10.57
Текст (ASCII): строка символов в апострофах (Указание: «$» является специальным знаком для обозначения апострофа или знака доллара внутри строки символов.)	'Siemens' 'So ist$'s' 'Nur $$25'

Табл.2.2 задает допустимые обозначения для ввода размера данных и начального адреса.

Таблица 2.2

Размер данных	**Пример**	**Описание**
Байт	VB10	Записывает в память последующие значения в формате байта, начиная с указанного адреса.
Слово	VW22	Записывает в память последующие значения в формате слова, начиная с указанного адреса.
Двойное слово	VD100	Записывает в память последующие значения в формате двойного слова, начиная с указанного адреса.
Автоматический размер	V10	Записывает в память данные с применением минимального размера (байт, слово или двойное слово), который требуется для записи этих значений. Указанные в этой строке значения записываются в память переменных, начиная с указанного адреса.
Как предыдущий размер	(Столбец адреса остается пустым)	Записывает в память данные в формате байта, слова или двойного слова, в зависимости от того, какой размер был задан в предыдущей строке.

2.4. Работа с таблицей состояний/принудительного задания

С помощью таблицы состояний/принудительного задания можно читать, записывать или принудительно устанавливать переменные программы.

Редактор таблиц по умолчанию отображается в виде пиктограммы окна у нижнего края главного окна. Если хотите вызвать таблицу состоя-

ний/принудительного задания, то дважды щелкните мышью на этой пиктограмме или на кнопке "Восстановить/ Минимизировать" на этой пиктограмме.

2.4.1. Чтение и запись переменных в таблицу состояний/ принудительного задания.

На рис.2.7 показан пример таблицы состояний/принудительного задания. Для чтения или записи переменных в редакторе таблиц действуйте следующим образом:

1. Введите в первом поле в адресном столбце адрес или символическое имя элемента программы, значение которого хотите прочитать или записать. Затем нажмите клавишу ввода. Повторите этот шаг для всех элементов, которые хотите иметь в таблице.

2. Если этим элементом является бит (например, I, Q или M), то во втором столбце отображается двоичный формат. Если этим элементом является байт, слово или двойное слово, то можно отметить поле в столбце формата и двойным щелчком мыши или нажатием клавиши пробела пролистать допустимые форматы.

3. Если хотите отобразить текущее значение элемента в таблице, то щелкните мышью на кнопке для однократного считывания или на кнопке для постоянного чтения .

Если хотите закончить актуализацию, то щелкните мышью на кнопке STOP .

4. Для изменения значения введите новое значение в столбец «**New Value**» [«Новое значение»]. Затем щелкните мышью на кнопке для записи , чтобы записать это значение в CPU.

Рис. 2.7

Пояснения к рисунку. Binary – двоичный, Signed – число со знаком, Unsigned – число без знака, Hexadecimal – 16-ричный, String – плавающая точка.

2.4.2. Задание значений в таблице состояний/принудительного задания

Если хотите задать переменной в таблице состояний/ принудительного задания определенное значение, действуйте следующим образом:

1. Введите в первом поле в адресном столбце адрес или символическое имя элемента программы, значение которого хотите установить.

2. Если этот элемент является битом (например, I, Q или M), то во втором столбце отображается двоичный формат. Для битов формат не может изменяться. Если этим элементом является байт, слово или двойное слово, то можно отметить поле в столбце формата и двойным щелчком мыши или нажатием клавиши пробела пролистать допустимые форматы.

3. Если хотите задать переменной ее текущее значение, то вначале считайте текущие значения из контроллера с помощью команды меню **Debug → Single Read** [Тестирование → Однократное считывание] или щелкните мышью на кнопке для однократного считывания  .

Щелкните мышью на поле, в котором находится текущее значение, которое хотите установить, или перейдите к этому полю. Щелкните мышью на кнопке принудительного задания  , когда курсор находится на текущем значении, которое хотите присвоить переменной.

4. Для изменения и принудительного задания переменной нового значения запишите желаемое значение в столбец «**New Value**» [«Новое значение»] и щелкните мышью на кнопке принудительного задания .

5. Если хотите отобразить все принудительно установленные на данный момент переменные, щелкните мышью на кнопке для чтения принудительно установленных значений  .

6. Если хотите снять заданные значения со всех принудительно переменных в CPU, щелкните мышью на кнопке для отмены принудительного задания  .

2.4.3. Редактирование адресов

Для редактирования адресного поля выберите желаемое поле с помощью клавиш перемещения курсора или мыши.

При начале ввода данных поле очищается и записываются новые символы.

Если дважды щелкнуть мышью или нажать клавишу F2, то поле подсвечивается. После этого можно с помощью клавиш курсора передвигать курсор на место, которое хотите редактировать.

2.5. Использование символической адресации

С помощью таблицы символов можно присваивать символические имена входам, выходам и адресам внутренней памяти (рис.2.8). Можно использовать символы, поставленные в соответствие этим адресам, в редакторах LAD и STL и в таблице состояний/принудительного задания в STEP 7–Micro/WIN 32.

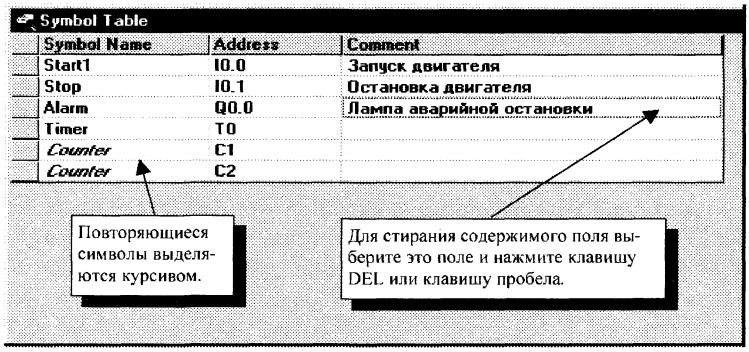


Рис. 2.8

2.5.1. Правила ввода символических адресов

Первый столбец таблицы символов служит для маркировки строки. Другие столбцы предназначены для символического имени, адреса и комментария. В каждой строке Вы присваиваете адресу цифрового входа или выхода, адресу памяти, специальному маркеру или другому элементу символическое имя. Комментарий к символическому имени является необязательным. При редактировании таблицы символов соблюдайте следующие правила:

- Символические имена и абсолютные адреса вводят в произвольной последовательности.
- Можно вводить символические имена максимум из 23 символов (возможно, что в зависимости от установленного в Windows размера шрифта не все символы будут видимыми в редакторе LAD).
- Можно определить максимум 500 символов.
- Таблица символов различает большие и малые буквы: например, «Pumpe1» отличается от «pumpe1».
- Символы пробела перед символическим именем и после него будут стерты. Символы пробела внутри символического имени будут заменены знаком подчеркивания. Пример: «Start_Motor_2».
- Повторяющиеся символические имена и/или адреса выделяются голубым курсивом, не компилируются и не могут использоваться в программе. Пересекающиеся адреса не помечаются как повторяющиеся; например, VB0 и VW0 пересекаются в памяти, но не выделяются как повторяющиеся адреса.

2.5.2. Вызов редактора таблиц символов

Редактор таблиц символов по умолчанию отображается в виде пиктограммы окна у нижнего края главного окна. Если хотите вызвать таблицу символов, то дважды щелкните на этой пиктограмме или на кнопке «Восстановить/Минимизировать» на этой пиктограмме.

2.5.3. Функции редактирования внутри таблицы символов

Таблица символов имеет в своем распоряжении следующие функции редактирования:

- **Edit → Cut/Copy/Paste** [Редактирование → Вырезание/Копирование/Вставка]: внутри поля или между разными полями.
- **Edit → Cut/Copy/Paste** [Редактирование → Вырезание/Копирование/Вставка]: нескольких взаимосвязанных строк.
- **Edit → Insert Row** [Редактирование → Вставка строки]: выше строки, в которой находится курсор. Для этого можно использовать также клавишу INSERT.
- **Edit → Delete Row** [Редактирование → Стирание строки]: одной или нескольких соседних выделенных строк. Для этого можно использовать также клавишу DEL.

При редактировании любого поля данных выбирайте желаемое поле с помощью клавиш курсора или мыши. В начале ввода данных поле очищается и записываются новые символы. Если дважды щелкнуть мышью или нажать клавишу F2, то поле будет подсвечено. После этого можно с помощью клавиш курсора переместить курсор на место, которое хотите обрабатывать.

2.5.4. Сортировка записей таблицы

После ввода символических имен и соответствующих им абсолютных адресов можно сортировать таблицу символов по символическим именам в алфавитном порядке или по возрастанию адресов. Для этого действуйте следующим образом:

- Выберите команду меню **View → Sort by Name** [Вид → Сортировка по имени] для сортировки в алфавитном порядке.
- Выберите команду меню **View → Sort by Address** [Вид → Сортировка по адресу] для сортировки по возрастанию абсолютных адресов в следующей последовательности для областей памяти: I, Q, V, AI, AQ, M, C, T, S, SM и HC.

2.6. Сохранение данных в CPU S7–200

CPU S7–200 предоставляет различные методы для надежного хранения программы, данных программы и данных о конфигурации CPU:

- CPU имеет EEPROM, в котором можно постоянно хранить всю программу, некоторые области данных и данные о конфигурации CPU (рис.2.9).

- CPU имеет мощный конденсатор, обеспечивающий надежность данных в ОЗУ также и после отключения источника питания CPU. В зависимости от варианта исполнения CPU мощный конденсатор может поддерживать ОЗУ в течение нескольких дней.
- Некоторые варианты CPU поддерживают поставляемый по желанию батарейный модуль, с помощью которого можно продлить время, в течение которого поддерживается ОЗУ после отключения источника питания CPU. Этот батарейный модуль принимает на себя поддержку данных после разряда мощного конденсатора.

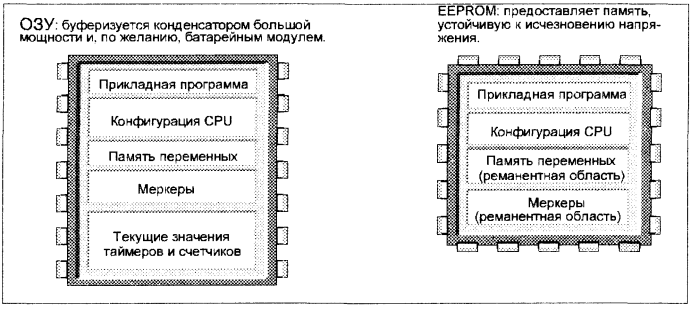


Рис. 2.9

Загрузка программы в CPU и из CPU

Программа состоит из трех компонентов: программы пользователя, блока данных (не обязательно) и конфигурации CPU (не обязательно). При загрузке программы в CPU эти компоненты помещаются в ОЗУ CPU (рис.2.10). Кроме того, CPU автоматически копирует программу пользователя, блок данных (DB1) и конфигурацию CPU в EEPROM для постоянного хранения.

Если загружать программу из CPU в PC (рис.2.11), то прикладная программа и конфигурация CPU загружаются из ОЗУ в компьютер. Если загружать из CPU блок данных, то энергонезависимая область блока данных (хранимая в EEPROM) сливается с возможно имеющимся остатком блока данных, который находится в ОЗУ. После этого компьютеру передается полный блок данных.

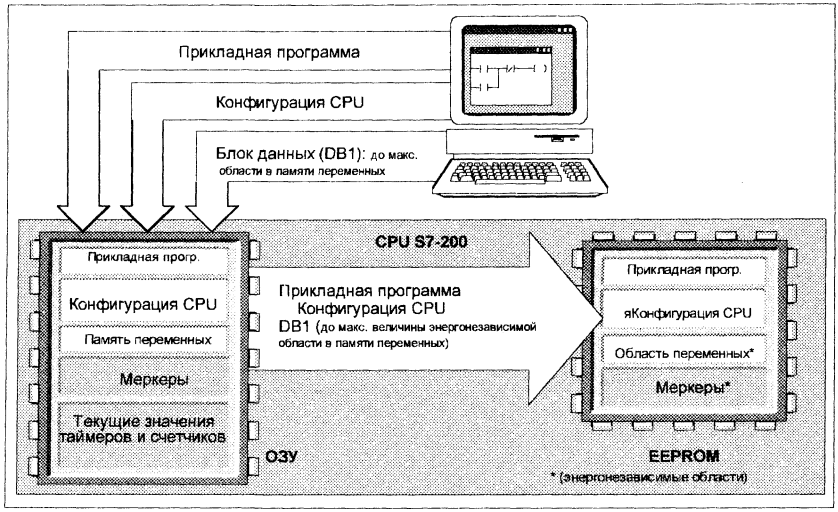


Рис. 2.10

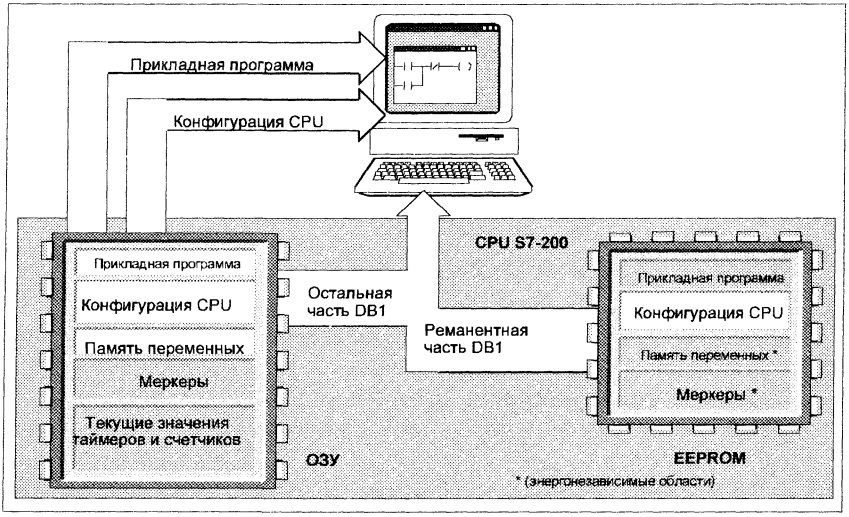


Рис. 2.11

2.7. Контрольные вопросы и задания

2.1 Создание и сохранения проекта.

1) Укажите путь для создания нового проекта.

2.2 Создание программы.

1) С помощью каких редакторов в STEP7-Micro/Win32 можно разработать прикладную программу?

2) Что такое LAD? Какие наборы операций существуют? Как задать заголовок? Какие комментарии возможны в LAD?

3) Создайте новый проект по схеме на рис.1 и сохраните его.

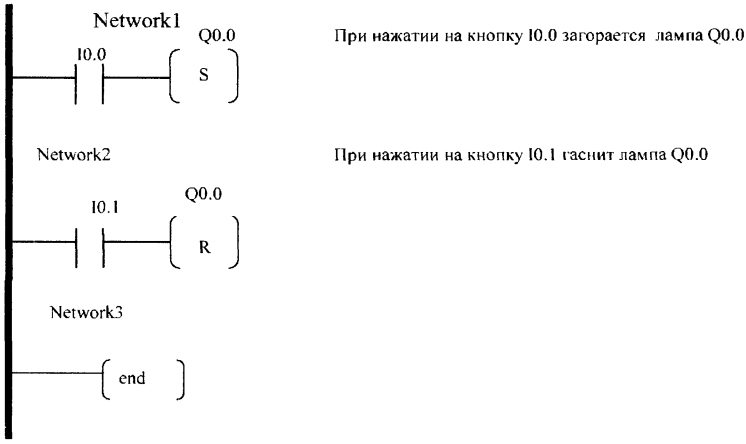


Рис. 1

4) Что такое STL? Какие требования следует соблюдать при вводе программы в редакторе STL?

5) Просмотрите проект из п.2.2.3 в форме STL.

6) Для чего необходима компиляция программы?

2.3 Создание блока данных.

1) Что позволяет сделать блок данных?

2) Как понимать обозначение VB10, VW22, VD100, V10?

2.4 Работа с таблицей состояния/принудительного задания.

1) Для чего необходима таблица состояния/принудительного задания (Status Chart)?

2) Что вводится в каждый столбец таблицы состояний/принудительного задания в режимах:

а) чтения/запись;

б) задания значений?

3) Просмотрите проект из п. 2.2.3 в режиме таблицы состояния/принудительного задания.

2.5 Использование символической адресации.

1) Для чего необходима символическая адресация (Symbol Table)?

2) Какое максимальное количество символов в имени переменной?

2.6 Какие методы CPU S7-200 предоставляет для хранения данных?

Глава 3. ОСНОВЫ ПРОГРАММИРОВАНИЯ В СРЕДЕ STEP 7–Micro/WIN 32

На рис.3.1 показан упрощенный информационный поток: Вход ⇒ Область памяти ⇒ Программа ⇒ Область памяти ⇒ Выход. Каждой области памяти поставлен в соответствие мнемонический идентификатор (в частности, «I» для входа и «Q» для выхода), через который можно производить доступ к данным в соответствующей области памяти.

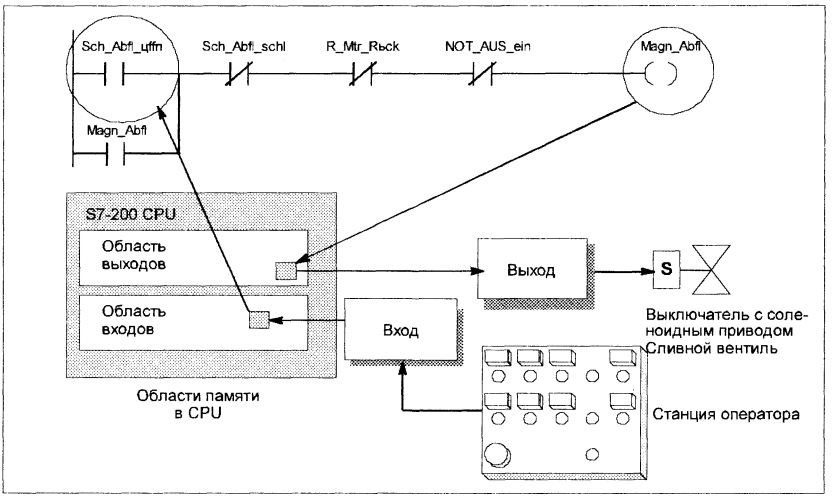


Рис. 3.1

3.1. Прямая адресация областей памяти в CPU

3.1.1. Обращение к данным через адреса

Если хотите обратиться к биту в области памяти, то необходимо указать адрес бита. Этот адрес состоит из идентификатора области памяти, адреса байта, а также номера бита. Такая адресация называется также адресацией «байт.бит». В примере (рис.3.2) за идентификатором области памяти и адресом байта I = вход, 3 = байт следует точка «.», чтобы отделить адрес бита 4.

Когда для адресации используют формат байта, можно обращаться к данным в различных областях памяти CPU (V, I, Q, M и SM) как к байтам, словам или двойным словам. Если хотите обратиться к байту, слову или двойному слову, то нужно задать этот адрес наподобие адреса бита, т.е. указать идентификатор области, размер данных (формат доступа) и начальный адрес значения в формате байта, слова или двойного слову (рис.3.3).

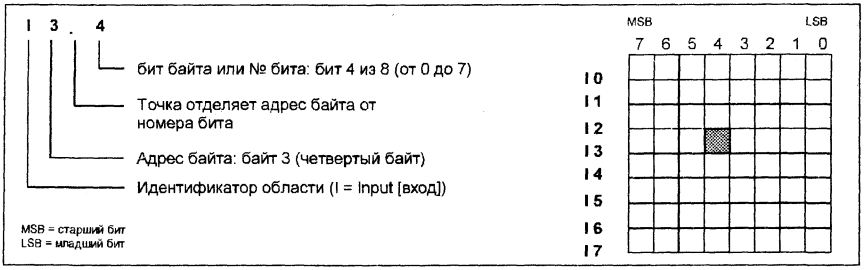


Рис. 3.2

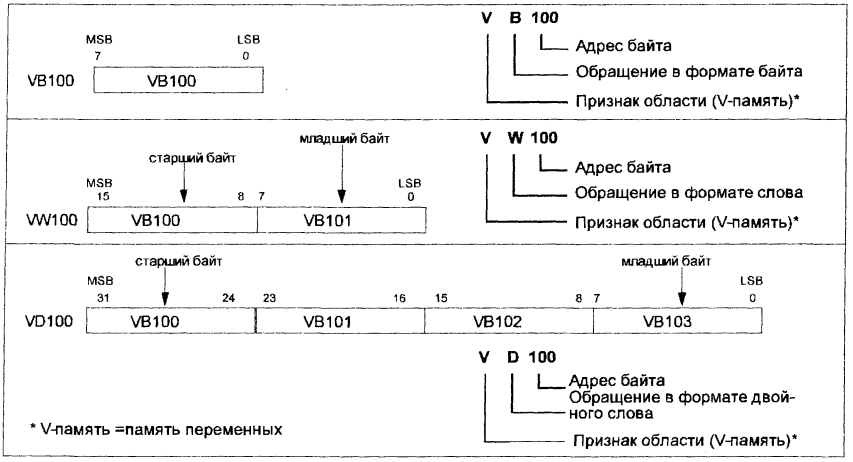


Рис. 3.3

Обращение к данным в других областях памяти CPU, например, T, C, HC и аккумуляторы) производится указанием в качестве адреса идентификатора области и номера элемента.

3.1.2. Представление чисел

В табл.3.1 показана область целочисленных значений, которые могут представляться данными различного размера.

Вещественные числа (числа с плавающей точкой) представляются как числа одинарной точности (32 бита), формат которых описан в стандарте ANSI/IEEE 754–1985. Обращение к значениям вещественных чисел производится в формате двойного слова.

Таблица 3.1

Размер данных	Диапазон целых чисел без знака		Диапазон целых чисел со знаком	
	Десятичный	Шестнадцатиричный	Десятичный	Шестнадцатиричный
B (байт): 8 битов	от 0 до 255	от 0 до FF	от -128 до 127	от 80 до 7F
W (слово): 16 битов	от 0 до 65535	от 0 до FFFF	от -32768 до 32767	от 8000 до 7FFF
D (двойное слово): 32 бита	от 0 до 4294967295	от 0 до FFFF FFFF	от -2147483648 до 2147483647	от 8000 0000 до 7FFF FFFF

3.1.3. Адресация области отображения процесса на входах (I)

В начале каждого цикла CPU опрашивает физические входы и записывает эти значения в область отображения процесса на входах. Можно обращаться к этой области отображения процесса в формате бита, байта, слова или двойного слова.

Формат: бит I*[адрес байта].[адрес бита]* I0.1;
байт, слово, двойное слово I*[размер][начальный адрес байта]* IB4.

3.1.4. Адресация области отображения процесса на выходах (Q)

В конце цикла CPU копирует значения из области отображения процесса на выходах на физические выходы. Можно обращаться к этой области отображения процесса в формате бита, байта, слова или двойного слова.

Формат: бит Q*[адрес байта].[адрес бита]*Q1.1;
байт, слово, двойное слово Q*[размер][начальный адрес байта]*QB5.

3.1.5. Адресация памяти переменных (V)

В памяти переменных можно хранить промежуточные результаты, рассчитываемые операциями программы. Можно хранить в памяти переменных также другие данные, которые относятся к Вашему процессу или к Вашему решению задачи автоматизации. К памяти переменных можно обращаться в формате бита, байта, слова или двойного слова.

Формат: бит V*[адрес байта].[адрес бита]*V10.2;
байт, слово, двойное слово V*[размер][начальный адрес байта]* VW100.

3.1.6. Адресация маркеров (M)

Внутренние маркеры (область памяти маркеров, M) можно использовать как управляющие реле для того, чтобы сохранять промежуточные результаты операций или другую управляющую информацию. Можно обращаться к маркерам в формате бита, байта, слова или двойного слова.

Формат: бит M*[адрес байта].[адрес бита]* M26.7;
байт, слово, двойное слово M*[размер][начальный адрес байта]* MD20.

3.1.7. Адресация реле шагового управления (S)

С помощью реле шагового управления (S) расчленяют алгоритм функционирования установки на отдельные шаги или эквивалентные про-

граммные компоненты. С помощью реле шагового управления можно логически структуризовать управляющую программу. Можно обращаться к S–битам в формате бита, байта, слова или двойного слова.

Формат: бит S*[адрес байта].[адрес бита]* S3.1;

байт, слово, двойное слово S*[размер][начальный адрес байта]*SB4.

3.1.8. Адресация специальных маркеров (SM)

С помощью специальных маркеров можно производить обмен информацией между CPU и программой. Кроме того, специальные маркеры служат для того, чтобы выбирать особые функции CPU S7–200 и управлять ими. К ним относятся:

- бит, который включается только в первом цикле;
- биты, которые включаются и выключаются на определенных тактах;
- биты, которые отображают состояние арифметических и других операций.

Область памяти специальных маркеров основывается на битах, однако можно обращаться к данным в этой области в формате бита, байта, слова или двойного слова.

Формат: бит SM*[адрес байта].[адрес бита]* SM0.1;

байт, слово, двойное слово SM*[размер][начальный адрес байта]* SMB86

В табл. 3.2 приведено описание специальных маркеров байта SMB0.

Таблица 3.2

Специальные маркеры	Описание
SM0.0	Этот бит включен всегда.
SM0.1	Этот бит включен в первом цикле. Он используется, например, для вызова подпрограммы инициализации.
SM0.2	Этот бит включается на время одного цикла, если потеряны ремонентные данные. Он может использоваться либо как маркер ошибки, либо как механизм вызова особых пусковых последовательностей.
SM0.3	Этот бит включается на время одного цикла, если режим работы RUN устанавливается при включении питания. Этот бит может быть использован, чтобы предоставить время на разогрев установки.
SM0.4	Этот бит обеспечивает тактовый импульс, который 30 секунд включен и 30 секунд выключен, то есть время цикла, равное 1 минуте. Благодаря этому, Вы имеете в своем распоряжении легко программируемую задержку или интервал между импульсами, равные 1 минуте.
SM0.5	Этот бит обеспечивает тактовый импульс, который 0,5 секунды включен и 0,5 секунды выключен, то есть время цикла, равное, равное 1 секунде. Благодаря этому, Вы имеете в своем распоряжении легко программируемую задержку или интервал между импульсами, равные 1 секунде.
SM0.6	Этот бит в одном цикле включен, а в следующем цикле выключен. Вы можете использовать этот бит как вход счетчика циклов.

Специальные маркеры	Описание
SM0.7	Этот бит показывает положение переключателя режимов работы (TERM – выключен, RUN – включен). Если Вы используете этот бит для деблокировки свободно программируемой связи, когда переключатель стоит в положении RUN, то можно разрешить нормальную связь с устройством программирования, переставляя переключатель в TERM.

3.1.9. Адресация таймеров (T)

В CPU S7–200 таймеры являются элементами, подсчитывающими приращения времени. Таймеры S7–200 имеют разрешающую способность 1 мс, 10 мс и 100 мс. Каждый таймер имеет следующие две переменные:

- текущее значение – это целое число (16 битов) со знаком хранит значение времени таймера;
- бит таймера – этот бит включается (устанавливается в «1»), когда текущее значение таймера больше или равно предварительно установленному значению (предварительно устанавливаемое значение вводится вместе с операцией).

Обращение к обоим элементам данных производится через адрес таймера (T + номер таймера). Куда происходит обращение – к биту таймера или к текущему значению таймера – определяется по соответствующей операции. Операции с операндами в формате бита обеспечивают доступ к биту таймера, тогда как операции с операндами в формате слова обеспечивают доступ к текущему значению. На рис.3.4 видно, что операция «замыкающий контакт» обеспечивает доступ к биту таймера, тогда как операция передачи слова (MOVW) обеспечивает доступ к текущему значению таймера.

Формат: T*[номер таймера]* T24.

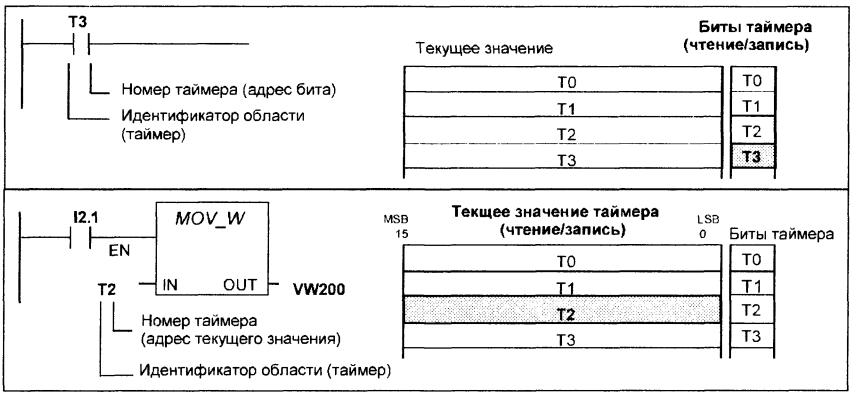


Рис. 3.4

3.1.10. Адресация счетчиков (С)

В CPU S7-200 счетчики являются элементами, подсчитывающими нарастающие фронты сигнала на счетных входах. CPU имеет счетчики двух видов: счетчик первого вида считает только вперед, тогда как счетчик другого вида считает как вперед, так и назад. Каждый счетчик имеет в своем распоряжении следующие две переменные:

- текущее значение – это целое число (16 битов) со знаком хранит накопленное значение счетчика;
- бит счетчика – этот бит включается (устанавливается в «1»), когда текущее значение счетчика больше или равно предварительно установленному значению (предварительно устанавливаемое значение вводится вместе с операцией).

Доступ к обеим переменным осуществляется через адрес счетчика (С + номер счетчика). Куда происходит доступ – к биту счетчика или к текущему значению счетчика – определяется по соответствующей операции. Операции с операндами в формате бита обеспечивают доступ к биту счетчика, тогда как операции с операндами в формате слова обеспечивают доступ к текущему значению. На рис.3.5 видно, что операция «замыкающий контакт» обеспечивает доступ к биту счетчика, тогда как операция передачи слова (MOVW) обеспечивает доступ к текущему значению счетчика.

Формат: С*[номер счетчика]* С20.

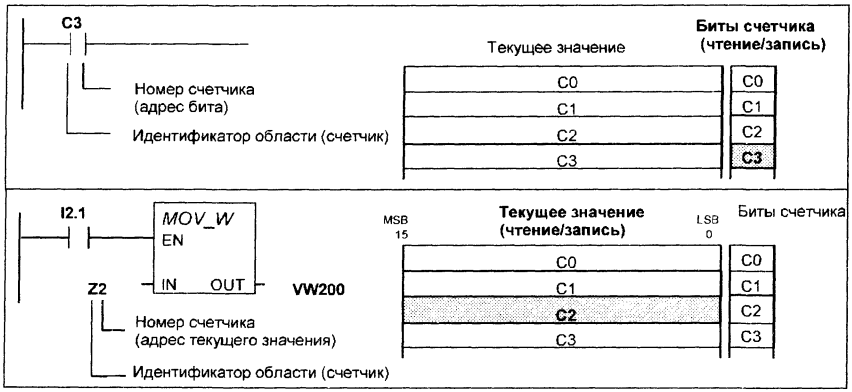


Рис. 3.5

3.1.11. Адресация аналоговых входов (AI)

S7-200 преобразует реальные аналоговые значения (например, напряжение, температуру) в цифровые значения с разрядностью слова (16 битов). Доступ к этим значениям осуществляется через идентификатор области (AI), размер данных (W) и начальный адрес байта. Так как в случае аналоговых входов речь идет о словах, всегда начинающихся с четного

байта (следовательно, с байта 0, 2, 4 и т.д.), то обращение к ним производится с использованием адресов четных байтов (например, AIW0, AIW2, AIW4) (рис.3.6). Аналоговые входы могут только считываться.

Формат: AIW*[начальный адрес байта]* AIW4.

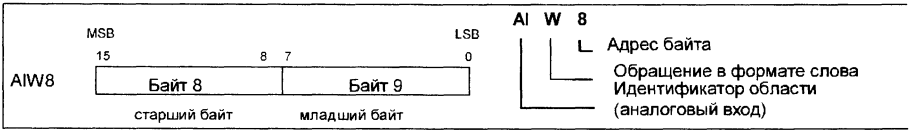


Рис. 3.6

3.1.12. Адресация аналоговых выходов (AQ)

S7–200 преобразует цифровые значения с разрядностью слова (16 битов) в ток или напряжение пропорционально цифровому значению. Обращение к этим значениям производится через идентификатор области (AQ), размер данных (W) и начальный адрес байта. Так как в случае аналоговых выходов речь идет о словах, всегда начинающихся с четного байта (следовательно, с байта 0, 2, 4 и т.д.), то обращение к ним производится с использованием адресов четных байтов (например, AQW0, AQW2, AQW4) (рис.3.7). Программа не может считывать значения аналоговых выходов.

Формат: AQW*[начальный адрес байта]* AQW4.

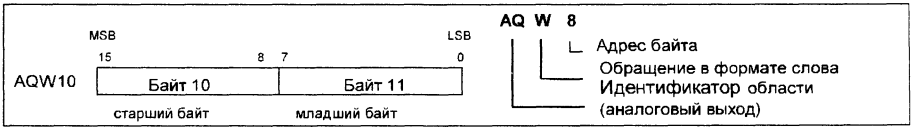


Рис. 3.7

3.1.13. Адресация аккумуляторов

Аккумуляторы являются элементами для чтения/записи, которые используются как память. С помощью аккумуляторов можно, например, передавать параметры в подпрограмму, а также принимать их обратно, или сохранять промежуточные результаты вычислений. CPU имеет четыре 32-разрядных аккумулятора (AC0, AC1, AC2 и AC3). Обращение к данным в аккумуляторах может производиться в формате бита, байта, слова и двойного слова. Если обращение к аккумулятору производится в формате байта или слова, то при этом используются младшие 8 или 16 битов значения, хранимого в аккумуляторе. Если обращение к аккумулятору производится в формате двойного слова, то используются все 32 бита. Размер данных, к которым производится доступ, определяется операцией, посредством которой Вы обращаетесь к аккумулятору (рис.3.8).

Формат: AC*[номер аккумулятора]* AC0.

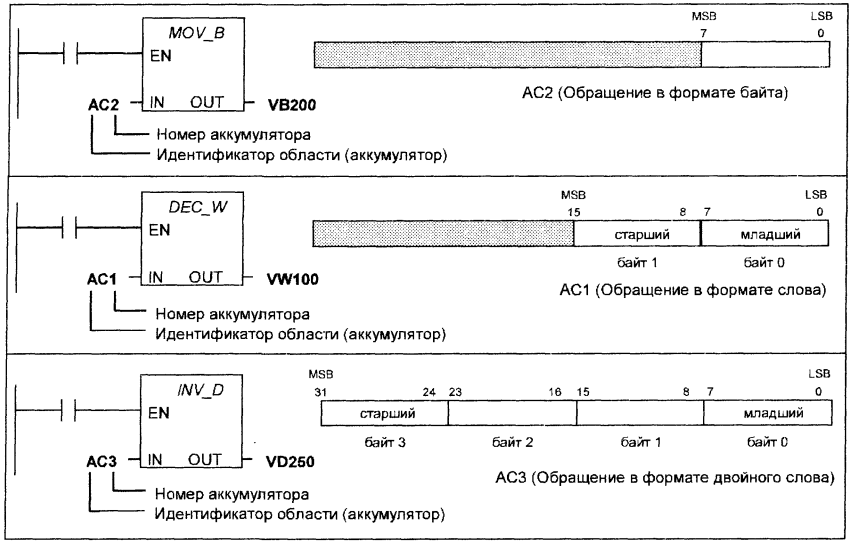


Рис. 3.8

3.1.14. Адресация быстрых счетчиков

Быстрые счетчики подсчитывают события быстрее, чем CPU может опрашивать эти события. Быстрые счетчики имеют в своем распоряжении 32-битное счетное значение (текущее значение). Если хотите обратиться к счетному значению быстрого счетчика, то задайте адрес быстрого счетчика посредством идентификатора области (HC) и номера счетчика (например, HC0). Текущее значение счетчика защищено от записи и может адресоваться только в формате двойного слова (32 бита) (рис.3.9).

Формат: HC*[номер счетчика]* HC1.

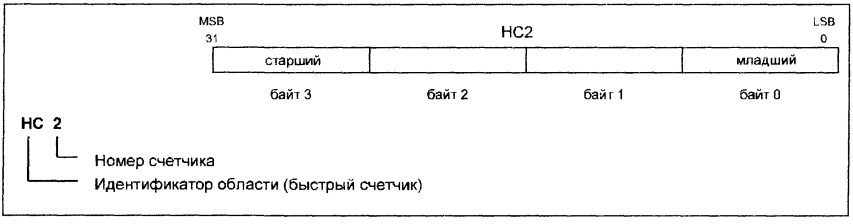


Рис. 3.9

3.1.15. Применение констант

Во многих операциях для S7-200 можно использовать константы. Константы могут быть байтами, словами и двойными словами. CPU хранит все константы как двоичные значения, которые могут представляться в десятичном, шестнадцатиричном и ASCII-формате.

Десятичный формат: [десятичное значение]
Шестнадцатиричный формат: 16#[шестнадцатиричное значение]
ASCII-формат: ' [ASCII-текст] '.

Нельзя задавать в CPU S7-200 специфические типы данных (например, когда хотите указать, что константа должна храниться в виде целого числа (16 битов), целого числа со знаком или целого числа (32 бита)). CPU S7-200 также не проверяет типы данных. Например, операция сложения может использовать хранимое в VW100 значение как целое число со знаком, тогда как операция EXKLUSIV ODER использует то же самое значение из VW100 как двоичное значение без знака.

Следующие примеры показывают константы в десятичном, шестнадцатиричном и ASCII-формате:

- Десятичная константа: 30574
- Шестнадцатиричная константа: 16#2C5F
- ASCII-константа: 'Текст в апострофах'.

В табл. 3.3 приведены допустимые диапазоны, а в табл.3.4 – области операндов CPU-214.

Таблица 3.3

Описание	CPU-214
Размер программы пользователя	2 K слов
Размер данных пользователя	2 K слов
Отображение процесса на входах	I0.0 - I7.7
Отображение процесса на выходах	Q0.0 - Q7.7
Аналоговые входы (защищенные от записи)	AIW0 - AIW30
Аналоговые выходы (защищенные от записи)	AQW0 - AQW30
Память переменных (V) Область, устойчивая к нулевому напряжению (макс.)	V0.0 - V4095.7 V0.0 - V1023.7
Маркеры (M) Область, устойчивая к нулевому напряжению (макс.)	M0.0 - M31.7 MB0 - MB13
Специальные маркеры (SM) Защищенные от записи	SM0.0 - SM85.7 SM0.0 - SM29.7
Таймеры Формирование задержки включения с запоминанием 1 мс Формирование задержки включения с запоминанием 10 мс Формирование задержки включения с запоминанием 100 мс Формирование задержки включения 1 мс Формирование задержки включения 10 мс Формирование задержки включения 100 мс	128 (T0 - T127) T0, T64 T1 - T4, T65 - T68 T5 - T31, T69 - T95 T32, T96 T33 - T36, T97 - T100 T37 - T63, T101 - T127
Счетчики	C0 - C127
Быстрые счетчики	HC0 - HC2

Описание	CPU-214
Реле шагового управления	S0.0 - S15.7
Аккумуляторы	AC0 - AC3
Переходы/Метки перехода	0 - 255
Вызовы/Подпрограммы	0 - 63
Программы обработки прерываний	0 - 127
События прерываний	0 - 20

Таблица 3.4

Формат доступа	CPU-214
Бит (Байт.Бит)	V 0.0 - 4095.7 I 0.0 - 7.7 Q 0.0 - 7.7 M 0.0 - 31.7 SM 0.0 - 85.7 T 0 - 127 C 0 - 127 S 0.0 - 15.7
Байт	VB 0 - 4095 IB 0 - 7 QB 0 - 7 MB 0 - 31 SMB 0 - 85 AC 0 - 3 SB 0 - 15 Константа
Слово	VW 0 - 4094 T 0 - 127 C 0 - 127 IW 0 - 6 QW 0 - 6 MW 0 - 30 SMW 0 - 84 AC 0 - 3 AIW 0 - 30 AQW 0 - 30 SW 0 - 14 Константа
Двойное слово	VD 0 - 4092 ID 0 - 4 QD 0 - 4 MD 0 - 28 SMD 0 - 82 AC 0 - 3 HC 0 - 2 SD 0 - 12 Константа

3.2. Косвенная адресация областей памяти в CPU

Косвенная адресация использует указатели для обращения к данным в памяти. В CPU S7–200 посредством указателей можно косвенно адресовать следующие области памяти: I, Q, V, M, S, T, C (только текущее значение). Нельзя косвенно адресовать значения отдельных битов.

3.2.1. Создание указателя

Если хотите обратиться к адресу косвенно, то вначале должны создать указатель, указывающий на этот адрес. Указатели являются двойными словами. Для создания указателя используют операцию передачи двойного слова (MOVD). Эта операция передает адрес в ячейку памяти с другим адресом или в аккумулятор, которая или который, соответственно, служит потом указателем (рис.3.9). С помощью знака «&» указывают, что именно адрес, а не соответствующее ему значение должно передаваться в пункт назначения.

Формат: *&[адрес памяти]* &MB16.

При создании указателя можно задавать в операции MOVD в качестве целевого адреса только адреса памяти переменных и аккумуляторы AC1, AC2 и AC3. При косвенной адресации нельзя использовать в качестве указателя AC0.

3.2.2. Доступ к данным с помощью указателя

Звездочка (*) перед операндом операции указывает, что этот операнд является указателем. В примере на рис.3.10 запись *AC1 указывает, что AC1 является указателем, который содержит адрес значения слова, заданного операцией «передача слова» (MOVW). В данном примере значения V200 и V201 передаются в аккумулятор AC0.

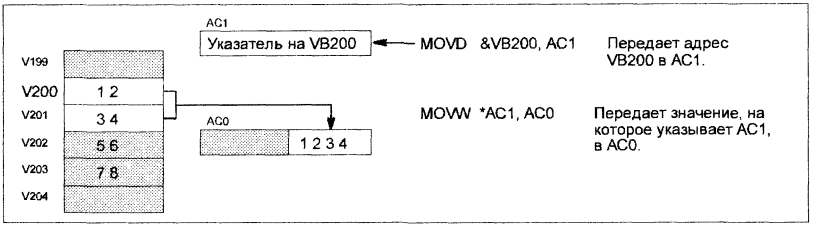


Рис. 3.10

3.2.3. Изменение указателей

Можно изменять значение указателя. Так как указатели являются 32-битными значениями, то изменять значения указателей нужно с помощью операций для двойных слов. Изменять значения указателей можно с помощью простых арифметических операций, например, путем сложения или инкрементирования, учитывая размер данных:

- если обращаетесь к байту, увеличьте значение указателя на 1;
- если обращаетесь к словам или к текущим значениям таймеров или счетчиков, то прибавляйте значение 2 или инкрементируйте значение указателя на 2;
- если обращаетесь к двойному слову, то прибавляйте значение 4 или инкрементируйте значение указателя на 4.

3.3. Языки программирования

CPU S7–200 (и STEP 7–Micro/WIN 32) поддерживает следующие языки программирования:

- список команд (STL), мнемоника которых представляет функцию CPU;
- язык релейно-контакных схем (LAD) является графическим языком программирования, который похож на электрические схемы.

Кроме того, STEP 7–MicroWIN 32 предоставляет два способа отображения адресов и операций в программе: международный и SIMATIC. Между этими двумя способами представления существует прямое соответствие, и функциональные возможности обоих способов представления идентичны.

3.3.1. Основные элементы релейно-контактных схем

При проектировании программы в форме релейно-контактных схем работают с графическими компонентами, с помощью которых строятся логические сети. Для разработки программы используют следующие элементы (рис.3.11):

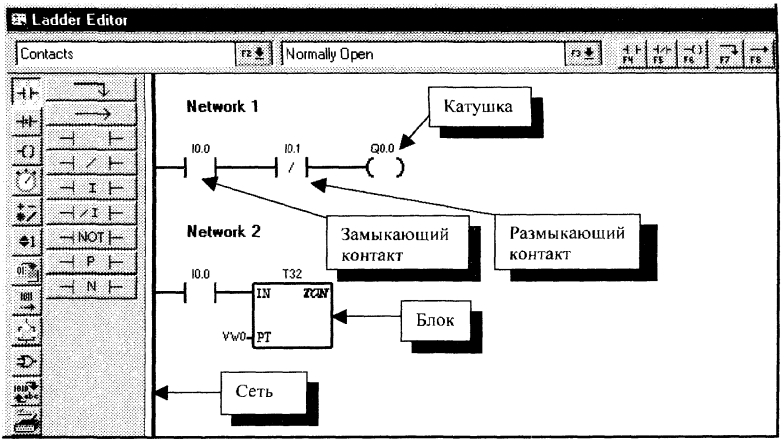


Рис. 3.11

- Контакты: Каждый контакт представляет собой переключатель, через который при замкнутом состоянии может протекать ток.
- Катушки: Каждая катушка представляет собой реле, которое включается при протекании тока.
- Блоки: Каждый блок представляет собой функцию, которая выполняется, когда к блоку течет ток.
- Сети: Сеть образует полную цепь тока. Ток течет от левой шины тока через замкнутые контакты к катушкам или блокам, которые за счет этого активизируются.

3.3.2. Операторы списка команд

Список команд представляет собой язык программирования, в котором каждая команда программы содержит операцию, мнемоника которой представляет функцию CPU.

На рис.3.12 показаны основные элементы программы в форме списка команд.

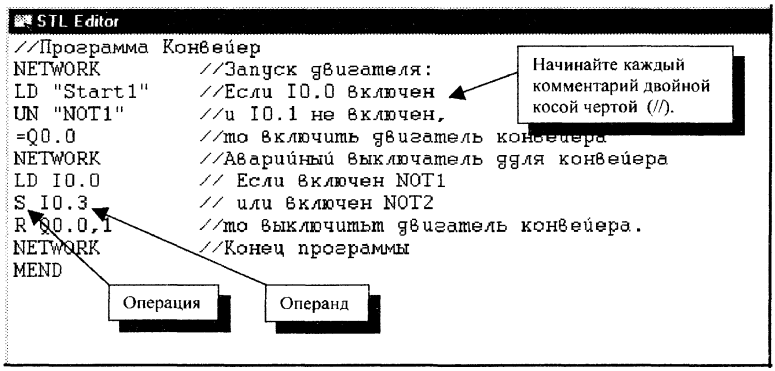


Рис. 3.12

Для решения логических задач операции STL работают с логическим стеком в CPU. Этот логический стек имеет глубину девять битов и разрядность один бит (рис.3.13). Большинство операций STL работает либо с первым битом, либо с первым и вторым битами стека. Новые значения могут вдвигаться (или добавляться) в стек. Если два самых верхних бита стека логически сопрягаются, то стек уменьшается на один бит.

В то время как большинство операций STL только считывают значение из стека, некоторые операции STL изменяют сохраненные в стеке значения. На рис.3.14 показаны три примера работы со стеком некоторых операций. В данном примере надписи «aw0» – «aw8» обозначают выходы логического стека, «nw» обозначает новое значение, подготавливаемое операцией, и S0 обозначает рассчитанное значение, которое сохраняется в логическом стеке. В примерах используются следующие булевы операторы логического сопряжения: UND (*) и ODER (+).

Биты логического стека		
	S0	Stack 0 - Первый уровень стека или вершина стека
	S1	Stack 1 - Второй уровень стека
	S2	Stack 2 - Третий уровень стека
	S3	Stack 3 - Четвертый уровень стека
	S4	Stack 4 - Пятый уровень стека
	S5	Stack 5 - Шестой уровень стека
	S6	Stack 6 - Седьмой уровень стека
	S7	Stack 7 - Восьмой уровень стека
	S8	Stack 8 - Девятый уровень стека

Рис. 3.13

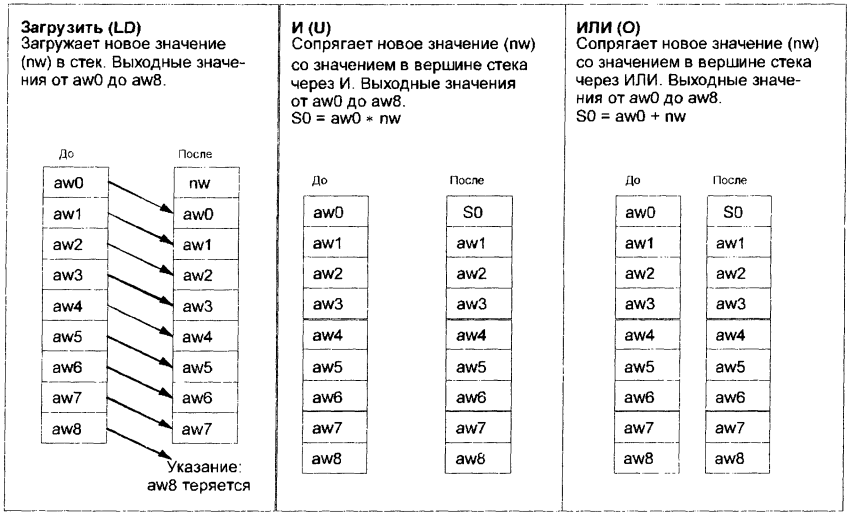


Рис. 3.14

3.4. Основные элементы для разработки программы

CPU S7–200 непрерывно обрабатывает программу, чтобы управлять задачей или процессом. Из главной программы можно вызывать различные подпрограммы и программы обработки прерываний.

Структуризация программы.

Программы для CPU S7–200 составляют из трех основных элементов: главная программа, подпрограммы (необязательные) и программы обработки прерываний (необязательные) (рис.3.15):

- Главная программа: В этой основной части программы располагаются операции, управляющие приложением. Операции главной про-

граммы в каждом цикле обрабатываются последовательно. Для окончания главной программы в форме LAD используется катушка абсолютного завершения программы, а в форме STL – операция окончания главной программы (MEND) (см. (1) на рис.3.15).

- Подпрограммы: Эти необязательные компоненты программы обрабатываются только тогда, когда они вызываются из главной программы. Располагайте подпрограммы после главной программы (после катушки абсолютного завершения в LAD или после операции MEND в STL). Завершайте каждую подпрограмму командой RET, «вернуться» (см. (2) на рис. 3.15).
- Программы обработки прерываний: Эти необязательные компоненты программы обрабатываются только тогда, когда появляется событие прерывания. Располагайте программы обработки прерываний после главной программы (после катушки абсолютного завершения в LD или после операции MEND в STL). Завершайте каждую программу обработки прерываний операцией RETI, «вернуться из программы обработки прерываний» (см. (3) на рис.3.15).

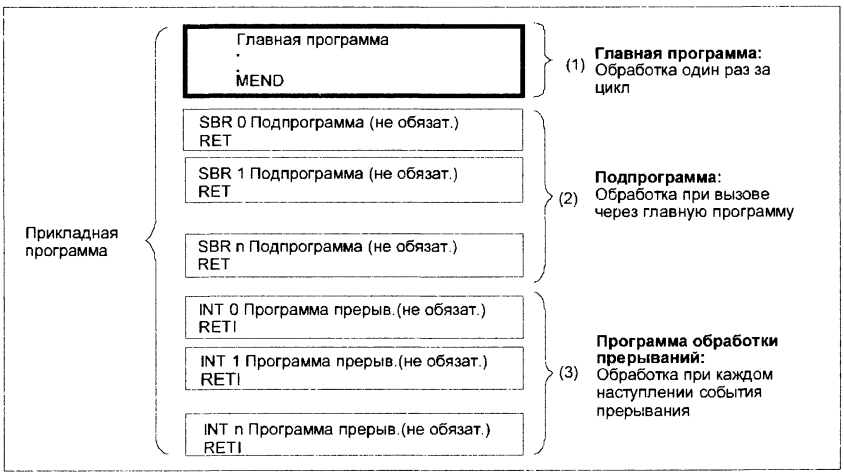


Рис. 3.15

Подпрограммы и программы обработки прерываний следуют за катушкой абсолютного завершения в LAD или за операцией MEND в STL в главной программе. Других указаний, которые необходимо соблюдать при размещении подпрограмм и программ обработки прерываний в программе, не существует. Можно располагать подпрограммы и программы обработки прерываний после главной программы в смешанной последовательности. Однако если хотите строить программу в легко читаемой и понятной фор-

ме, то лучше присоединить все подпрограммы непосредственно к главной программе, а затем расположить все программы обработки прерываний вслед за подпрограммами.

На рис.3.16 показан пример программы управляемого временем прерывания, с помощью которого можно считывать значение аналогового входа через каждые 100 мс.

LAD	STL
Главная программа	
Сегмент 1 SM0.1 — 0 (CALL) Сегмент 2 (END)	Сегмент 1 LD SM0.1 //Меркер первого цикла CALL 0 //Вызов подпрограммы 0 Сегмент 2 MEND
Подпрограмма	

Сегмент 3
0 SBR
Сегмент 4
SM0.0
MOV_B EN
100 — IN OUT — SMB34
(ENI)
ATCH EN
0 — INT
10 — EVENT
Сегмент 5
(RET)
	Сегмент 3
	SBR 0 //Начало подпрограммы 0
	Сегмент 4
	LD SM0.0 //Меркер утсановлен постоянно
	MOVB 100, SMB34 //Установка интервала прерыва-
	//ния, управляемого временем, на 100 мс
	ENI //Разрешить все события прерывания
	ATCH 0, 10 //Управляемое временем прерывание 0
	//поставить в соответствие программе прерываний 0
	Сегмент 5
	RET //Завершить подпрограмму

Программа обработки прерываний	
Сегмент 6 0 INT Сегмент 7 MOV_W EN AEW4 — IN OUT — VW100 Сегмент 8 (RETI)	Сегмент 6 INT 0 //Начало программы обработки прерываний 0 Сегмент 7 MOVW AEW4,VW100 //Опрос AEW4 Сегмент 8 RETI //Завершение программы обработки //прерываний

Рис. 3.16

3.5. Тестирование и контроль программы

3.5.1. Контроль программы путем выполнения определенного количества циклов

Можно указать, что CPU должен обрабатывать программу в течение определенного количества циклов (от 1 до 65 535 циклов). Если Вы выберете, сколько циклов должен выполнить CPU, то можете наблюдать обработку переменных процесса. Количество циклов, которое должен выполнить CPU, задается с помощью команды меню **Debug → Execute Scans** [Тестирование → Выполнение циклов]. На рис.3.17 показано диалоговое окно, в котором задается количество циклов.

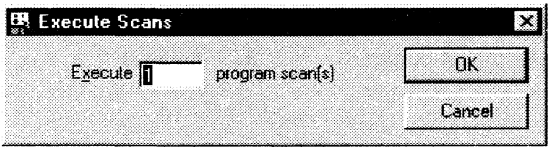


Рис. 3.17

3.5.2. Управление и контроль программы с помощью таблицы состояний/ принудительного задания

С помощью таблицы состояний/принудительного задания можно считывать, записывать, устанавливать и наблюдать переменные в то время, когда обрабатывается программа (рис. 3.18).

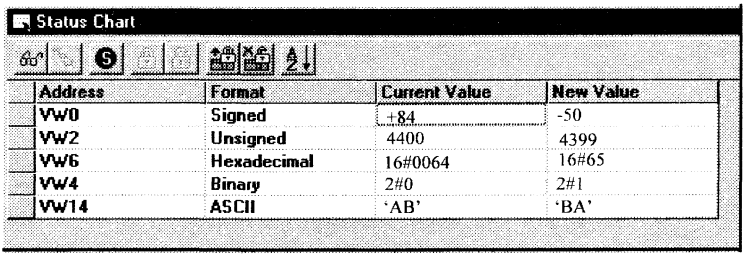


Рис. 3.18

3.5.3. Отображение статуса в программе, представленной в форме LAD

С помощью редактора программ STEP 7–Micro/WIN 32 можно контролировать статус программы в режиме «online», причем программа должна отображаться в форме релейно-контакных схем (рис.3.19). Для этого выберите команду меню **Debug → Ladder Status On** [Тестирование → Статус LAD Включен]. Так можно наблюдать состояние операций в программе при ее исполнении в CPU.

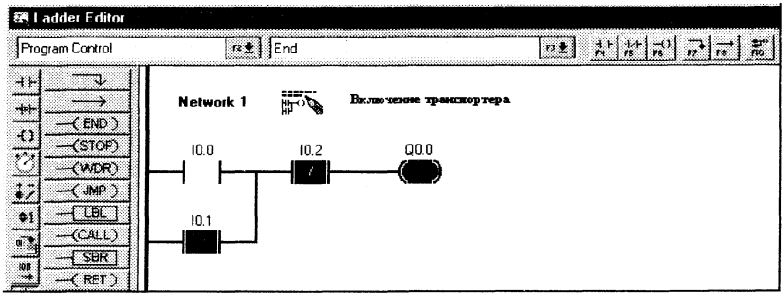


Рис. 3.19

3.5.4. Установка значений с помощью таблицы состояний/ принудительного задания

С помощью CPU S7–200 можно некоторые или все входы и выходы (биты I и Q) принудительно устанавливать на определенные значения (см. рис.3.18). Дополнительно можно принудительно задавать до 16 внутренних маркеров (V или M) либо аналоговых входов или выходов (AI или AQ). Значения в памяти переменных и значения маркеров можно задавать в форме байтов, слов и двойных слов. Аналоговые значения могут быть заданы только в форме слов, т.е. по четным байтам, например, AIW6 или AQW14). Все принудительно установленные значения записываются в EEPROM CPU.

Во время выполнения цикла принудительно установленные значения данных могут изменяться (программой, при актуализации входов и выходов или вследствие обработки коммуникаций). Поэтому CPU снова и снова переписывает принудительно установленные значения в различные моменты времени цикла. На рис.3.20 показано, в каких местах цикла CPU обновляет принудительно установленные переменные.

Функция принудительной установки (Force) подавляет операции прямого считывания или записи входов и выходов. Эта функция подавляет также выход, который был сконфигурирован на установку определенного значения после перехода в состояние STOP. Если CPU переходит в STOP, то выход сохраняет принудительно установленное, а не сконфигурированное значение.

Рис. 3.20

3.6. Устранение ошибок

CPU S7–200 подразделяет встречающиеся ошибки на серьезные (неисправимые) и незначительные (исправимые) ошибки. С помощью STEP 7–Micro/WIN 32 можно отобразить те коды ошибок, которые были порождены встретившимися ошибками. На рис.3.21 показано диалоговое окно «**CPU → Information**» [«CPU → Информация»], в котором отображаются код и описание ошибки. Список кодов ошибок приведен в табл.3.5 – 3.7.

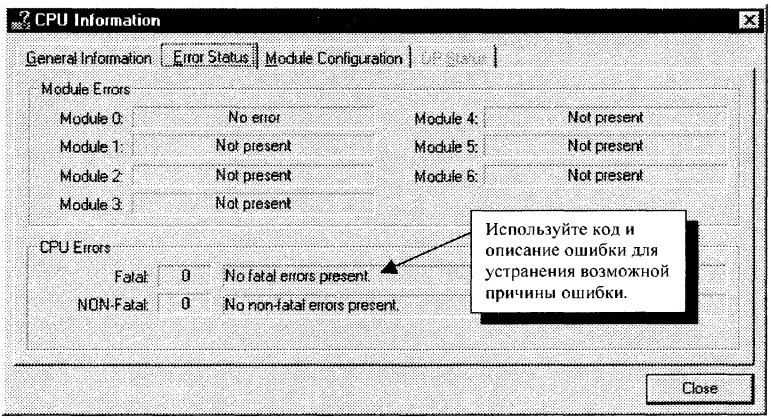


Рис. 3.21

3.6.1. Реакция на серьезные ошибки

Если появляется серьезная ошибка, то CPU прекращает обработку программы. В зависимости от типа ошибки CPU больше не может выполнять некоторые или даже все функции. Для того, чтобы устранить серьезные ошибки, необходимо перевести CPU в безопасное состояние, чтобы

можно было запросить информацию о сбойной ситуации. Если CPU обнаруживает серьезную ошибку, то он переходит в режим STOP, включает светодиодную индикацию серьезной ошибки (SF) и светодиодную индикацию режима STOP и выключает выходы. CPU остается в этом режиме до тех пор, пока не будет устранена сбойная ситуация.

Если условие возникновения серьезной ошибки устранено, то нужно снова запустить CPU. Для этого можно либо выключить и снова включить напряжение питания, либо перевести переключатель режимов работы из положения RUN или TERM в положение STOP. Новый пуск CPU стирает состояние ошибки и при запуске выполняется диагностика для того, чтобы проверить, устранена ли фактически серьезная ошибка. Если при этом обнаруживается еще одна серьезная ошибка, то снова загорается светодиодный индикатор CPU и тем самым указывается на то, что ошибка все еще имеет место. В противном случае CPU начинает свою нормальную работу.

Существует различные возможные сбойные ситуации, которые делают CPU неспособным к коммуникации. В таких случаях нельзя отобразить код ошибки CPU. Этот вид ошибки указывает чаще всего на неисправность аппаратуры, которая может быть устранена только путем ремонта CPU. Такие сбойные ситуации не могут быть устранены только путем изменения программы или общего стирания памяти CPU.

Таблица 3.5

Код	Неисправимые (фатальные) ошибки
0000	Фатальная ошибка не встретилась.
0001	Ошибка контрольной суммы в программе пользователя.
0002	Ошибка контрольной суммы в скомпилированной KOP–программе.
0003	Ошибка времени при контроле времени цикла.
0004	Отказ внутреннего EEPROM.
0005	Ошибка контрольной суммы внутреннего EEPROM в программе пользователя.
0006	Ошибка контрольной суммы внутреннего EEPROM в параметрах конфигурации.
0007	Ошибка контрольной суммы EEPROM в принудительно установленных данных.
0008	Ошибка контрольной суммы EEPROM в значениях по умолчанию в таблице выходов.
0009	Ошибка контрольной суммы внутреннего EEPROM в данных пользователя, DB1.
000A	Отказ модуля памяти.
000B	Ошибка контрольной суммы модуля памяти в программе пользователя. Сообщение об этой ошибка появляется, когда модуль памяти пуст (не содержит программу).
000C	Ошибка контрольной суммы модуля памяти в параметрах конфигурации.
000D	Ошибка контрольной суммы модуля памяти в принудительно установленных данных.
000E	Ошибка контрольной суммы модуля памяти в значениях по умолчанию в таблице выходов.
000F	Ошибка контрольной суммы модуля памяти в данных пользователя, DB1.
0010	Внутренняя ошибка программного обеспечения.
0011	Ошибка косвенной адресации сравнивающего контакта.
0012	Недопустимое значение в сравнивающем контакте.
0013	Неподходящий модуль памяти для данного CPU.

3.6.2. Устранение незначительных ошибок

Незначительные ошибки могут частично ограничивать работу CPU. Однако CPU и далее в состоянии обрабатывать программу и актуализировать входы и выходы. Можно с помощью STEP 7–Micro/WIN 32 отобразить коды ошибок, которые были порождены незначительными ошибками (см. рис.3.21). Есть три основные группы незначительных ошибок:

1) Ошибки этапа выполнения: все незначительные ошибки, которые обнаруживаются в режиме RUN, отмечаются в специальных маркерах. Программа может контролировать и анализировать эти специальные маркеры.

При пуске CPU считывает конфигурацию входов и выходов и сохраняет эту информацию в памяти системных данных и в специальных маркерах. При нормальной работе состояние входов и выходов регулярно обновляется и запоминается в специальных маркерах. Если CPU обнаруживает различия в конфигурации входов и выходов, то он устанавливает в байте ошибок модуля бит индикации измененной конфигурации. Модуль расширения не актуализируется до тех пор, пока этот бит не будет сброшен. Чтобы CPU мог сбросить этот бит, необходимо снова согласовать входы и выходы модуля с конфигурацией входов и выходов, записанной в памяти системных данных.

2) Ошибки при компиляции программы: CPU компилирует программу при ее загрузке. Если CPU обнаруживает, что программа нарушает некоторое правило компиляции, то он прерывает процесс загрузки и генерирует код ошибки. (Если программа уже была загружена в CPU, то эта программа имеется еще в EEPROM и она не теряется). После исправления программы можно загружать ее снова.

3) Ошибки программирования на этапе выполнения: сбойная ситуация может возникнуть во время исполнения программы. Например, косвенный указатель адреса, который при компиляции программы был действительным, во время обработки программы может быть изменен таким образом, что будет указывать на адрес вне допустимой области. Это рассматривается как ошибка программирования на этапе выполнения. Используя диалоговое окно «**CPU → Information**» [«CPU → Информация»] (см. рис.3.21), можно установить, какой вид ошибки встретился.

CPU не переходит в STOP, когда обнаруживается незначительная ошибка. Он отмечает эти события в специальных маркерах (SM) и продолжает обработку программы. Однако можно написать программу таким образом, что при появлении незначительной ошибки происходит принудительный переход в режим работы STOP. На рис.3.22 показан сегмент программы, которая контролирует специальный маркер. Операция переводит CPU в режим работы STOP, когда обнаруживается ошибка ввода/ вывода.

Таблица 3.6

Код	Ошибки во время выполнения (не фатальные ошибки)
0000	Ошибок нет.
0001	Блок HSC разблокирован перед обработкой блока HDEF.
0002	Назначение прерывания входу, который уже был поставлен в соответствие HSC.
0003	Назначение HSC входам, которые уже были поставлены в соответствие входному прерыванию.
0004	Попытка выполнения одной из операций ENI, DISI или HDEF в программе обработки прерываний.
0005	Попытка выполнения второго HSC с тем же самым номером до завершения первого HSC (HSC в программе обработки прерываний конфликтует с HSC в главной программе).
0006	Ошибка косвенной адресации.
0007	Ошибка в данных для операции TODW (запись в часы реального времени).
0008	Превышение максимальной глубины вложения подпрограмм.
0009	Обработка другой операции XMT в то время, когда отправитель активен.
000A	Попытка нового определения HSC с помощью второй операции HDEF для того же самого HSC.
0091	Ошибка области (вместе с адресной информацией): Проверьте области операндов.
0092	Ошибка в счетном поле операции (вместе со счетной информацией): Проверьте максимальное счетное значение.
0094	Ошибка области при записи в энергонезависимую память (вместе с адресной информацией).

Таблица 3.7

Код	Ошибки компиляции (не фатальные ошибки)
0080	Транслируемая программа слишком велика: Сократите программу.
0081	Выход за нижнюю границу стека: Разделите сегмент на несколько сегментов.
0082	Недопустимая операция: Проверьте мнемонику операции.
0083	Отсутствует MEND или недопустимая операция в главной программе: Введите операцию MEND или удалите недопустимую операцию.
0084	Резервный.
0085	Отсутствует FOR: Добавьте операцию FOR или удалите операцию NEXT.
0086	Отсутствует NEXT: Добавьте операцию NEXT или удалите операцию FOR.
0087	Отсутствует метка перехода (LBL, INT, SBR): Добавьте соответствующую метку перехода.
0088	Отсутствует RET или недопустимая операция в подпрограмме: Введите операцию RET в конце подпрограммы или удалите недопустимую операцию.
0089	Отсутствует RETI или недопустимая операция в программе обработки прерываний: Введите операцию RETI в конце программы обработки прерываний или удалите недопустимую операцию.
008A	Резервный.
008B	Дублированная область SCR.
008C	Дублированная метка перехода (LBL, INT, SBR): Переименуйте одну из меток перехода.
008D	Недопустимая метка перехода (LBL, INT, SBR): Обеспечьте, чтобы не превышалось допустимое количество меток перехода.

Код	Ошибки компиляции (не фатальные ошибки)
0090	Недопустимые параметры: Проверьте, являются ли допустимыми параметры операции.
0091	Ошибка области (вместе с адресной информацией): Проверьте области операндов.
0092	Ошибка в счетном поле операции (вместе со счетной информацией): Проверьте максимальное счетное значение.
0093	Превышение глубины вложения FOR/NEXT.
0095	Отсутствует операция LSCR (загрузка реле шагового управления).
0096	Отсутствует операция SCRE (завершение реле шагового управления) или недопустимая операция перед SCRE.

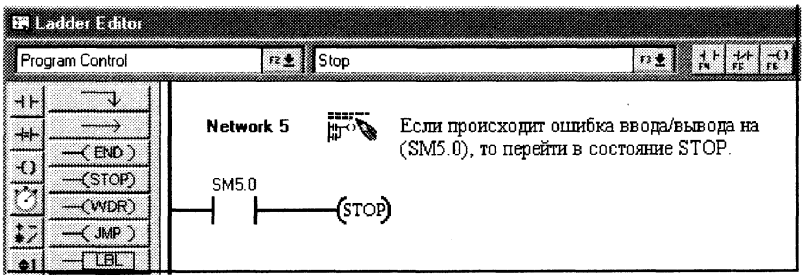


Рис. 3.22

3.7. Контрольные вопросы и задания

3.1.Прямая адресация области памяти в CPU.

1) Как происходит адресация области отображения процесса на входах, на выходах к биту, байту, слову и двойному слову?

Как происходит адресация памяти переменных, маркеров, специальных маркеров и шагового реле управления к биту, байту, слову и двойному слову?

2) Чем отличается обращение к области памяти таймера, счетчика, быстрого счетчика и аккумулятора от обращение к области памяти переменной?

3) Что означают сигналы I, V, Q, M SM T C AI AQ AC и HC?

4) Какой формат переменных имеет таймер и каков диапазон их значений?

5) Какой формат переменных имеет счетчик и каков диапазон их значений?

6)Соберите схему с рис.3.1 и рис.3.2. Измените значения переменных таймера и счетчика, посмотрите что это дает. Объясните работу этих схем.

7) Какие диапазоны значений принимают сигналы I, V, Q, M, SM, T, C, AI, AQ, AC и HC для процессора CPU-214?

3.2 Косвенная адресация областей памяти в CPU.

1) Что такое косвенная адресация? К каким областям памяти возможна косвенная адресация?
2) Зачем нужен указатель?
3) Как создать указатель?
4) Что означает &, *?
5) Возможно ли изменить значение указателя?

3.3 Языки программирования.

1) Какие языки программирования поддерживает STEP7-Micro/Win 32?
2) Укажите основные элементы РКС?
3) Что представляет собой список команд?

3.4 Основные элементы для разработки программы.

1) Какие элементы составляют структуру программы? Для чего необходим каждый.
2) Соберите схему с рис.3.3 и рассмотрите работу подпрограммы.

3.5. Тестирование и контроль программы.

1) Как задать контроль программы путем выполнения определенного количества циклов?
2) Для чего необходима таблица состояний/принудительного задания?
3) Что показывает «online»? Как задать данный режим?
4) Откройте проект из пункта 2.2.3 и постройте его в режиме «online».

3.6. Устранение ошибок.

1) Какие ошибки встречаются в CPU?
2) Как происходит отображение ошибок?
3) Как устраняются ошибки?

Глава 4. РЕШЕНИЕ ЗАДАЧ АВТОМАТИЗАЦИИ С ИСПОЛЬЗОВАНИЕМ ПЛК

4.1. Основные правила решения задач автоматизации

Существует много методов проектирования системы автоматизации. На рис.4.1 показаны некоторые из важных шагов при проектировании системы автоматизации.

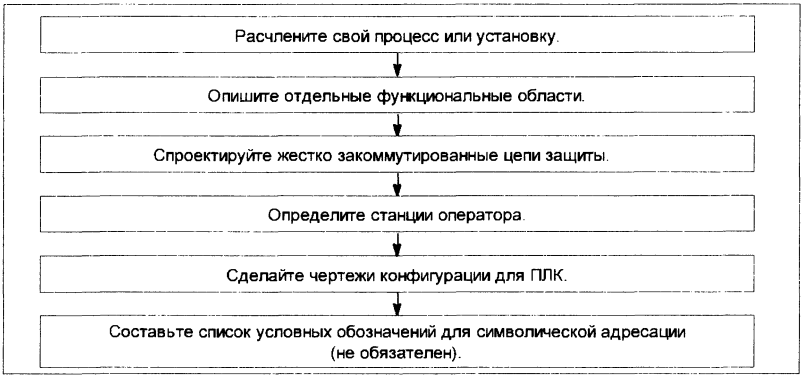


Рис. 4.1

Расчленение процесса или установки.

Расчлените процесс или установку на части, не зависящие друг от друга. Такие части устанавливают границы между несколькими системами автоматизации и влияют на описание функциональных областей, а также на распределение оборудования.

Описание функциональных областей.

Опишите принцип действия каждой части процесса или установки. Укажите при этом следующие пункты:

- входы/выходы (I/O),
- описание принципа действия,
- условия разблокировки (то есть состояния, которые должны достигаться, прежде чем становится возможным управление) для каждого исполнительного устройства (магнитные вентили, электродвигатели, приводы и т.д.),
- описание системы управления и контроля,
- интерфейсы с другими частями процесса или установки.

Проектирование цепей защиты.

Определите устройства, которые по соображениям безопасности нуждаются в жестко закоммутированных схемах. Управляющие устройства могут принимать опасные рабочие состояния, что может привести к не-

ожиданному пуску или изменению процессов функционирования установки. Если существует опасность того, что неожиданная или ошибочная работа установки приведет к тяжелым телесным повреждениям или материальному ущербу, то Вы должны предотвратить опасные рабочие состояния с помощью электромеханических средств вмешательства в программу, работающих независимо от CPU.

Для проектирования цепей защиты действуйте следующим образом:

- Выявите ненадлежащую или неожиданную работу исполнительных устройств, которая заключает в себе потенциальную опасность.
- Определите условия, при которых работа является безопасной, и установите, как эти условия распознаются независимо от CPU.
- Определите, как CPU и модули расширения влияют на процесс, когда включается и снова выключается напряжение и когда обнаруживаются ошибки. Такая информация должна использоваться только при проектировании нормальных и ожидаемых ненормальных режимов работы и не должна вводиться по соображениям безопасности.
- Спроектируйте коррекцию через ручное вмешательство или электромеханические средства вмешательства в программу, с помощью которых опасные процессы блокируются независимо от CPU.
- Предоставьте возможность передавать статусную информацию от независимых электрических цепей в CPU, так чтобы программа и каждый интерфейс оператора располагал требуемой информацией.
- Определите другие требования к безопасности, чтобы процесс мог протекать надежно.

Определение станций оператора.

Составьте планы станций оператора на основании требований из описаний функциональных областей. Включите следующие пункты:

- Расположение всех станций оператора относительно процесса или установки.
- Механическая компоновка устройств станции оператора (дисплей, переключатели, лампы и т.д.)
- Схемы проводных соединений с соответствующими входами и выходами CPU или модулей расширения.

Вычерчивание конфигурационных планов для ПЛК.

Составьте конфигурационные планы системы автоматизации на основании требований из описаний функциональных областей. Включите следующие пункты:

- Расположение всех CPU относительно процесса или установки.
- Механическая компоновка CPU и модулей расширения (включая шкафы и т.д.)
- Схемы проводных соединений для всех CPU и модулей расширения (включая номера устройств, коммуникационные адреса и адреса входов и выходов).

Составление списка символических имен.

Если Вы выбираете символическую адресацию, то должны поставить в соответствие абсолютным адресам символические имена. Задайте не только физические входы и выходы, но также и все элементы, используемые в программе.

4.2. Постановка задачи

Рассмотрим следующий пример:

С помощью микроконтроллера S7-200 (CPU-214) нужно управлять транспортером для перемещения деталей. Перед транспортером находится установка, которая каждые 30 секунд поставляет на ленту детали. Для перемещения каждой детали по ленте требуется около 1 минуты. Так как установка может простаивать, то транспортер, в зависимости от того, должны детали транспортироваться или нет, должен автоматически запускаться или останавливаться. После транспортировки детали помещаются в контейнер, емкость которого 150 деталей. После того как контейнер будет заполнен, необходимо остановить транспортер для того, чтобы отвезти заполненный контейнер и подвезти пустой. Кроме того, около транспортера установлено ограждение. Если в пределах огражденной зоны оказывается человек, то лента транспортера автоматически останавливается.

Протокол работы системы, которую необходимо разработать, может быть представлен в виде:

- четкого словесного описания,
- блок-схемы алгоритма,
- графа состояния,
- временной диаграммы.

4.2.1. Протокол работы системы

Установка включается кнопкой ВКЛ, на электрической схеме ей соответствует ключ S1, а выключается кнопкой ВЫКЛ – ключ S2. Транспортер приводится в движение электродвигателем М, а датчик S4 регистрирует детали на ленте. С помощью датчика S3 детали, поступающие с установки, регистрируются в начале ленты. Реле К1 управляет электродвигателем М в зависимости от состояния S3 и S4. Если нажата кнопка ВКЛ, и детали необходимо перемещать, то лента запускается. Если детали с установки не поступают на транспортер более 100 секунд (отсчет времени ведет таймер Т1), то лента останавливается. Детали в конце ленты регистрируются датчиком S5. После того как контейнер будет наполнен деталями (количество деталей в контейнере подсчитывает счетчик С1) лента транспортера останавливается, о чем сигнализирует лампа L1. Присутствие человека в пределах огражденной зоны регистрирует датчик S6, который управляет реле К2. Реле К2, в свою очередь, отключает электродвигатель М и одновременно включает лампу аварийной сигнализации L2.

На рис. 4.2 приведена блок-схема алгоритма работы системы.

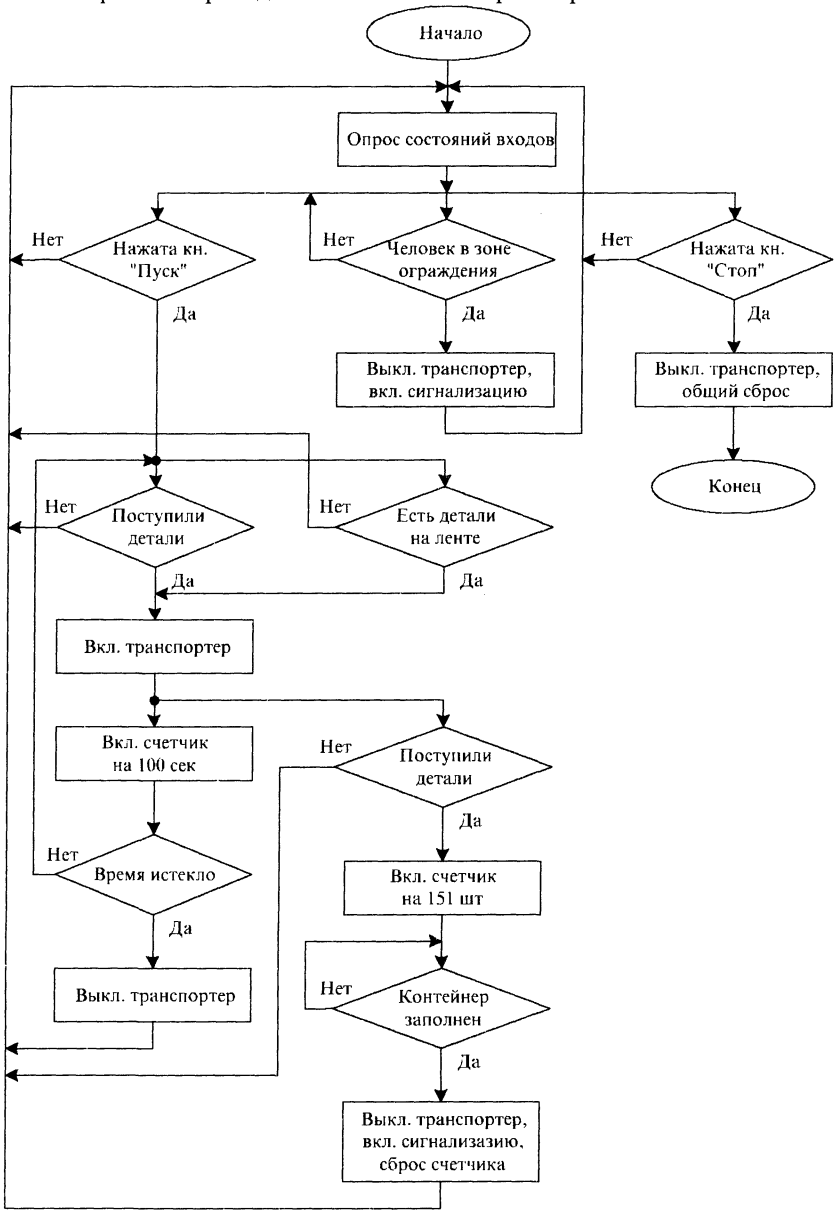


Рис. 4.2

На рис.4.3. приведена принципиальная электрическая схема устройства управления транспортером.

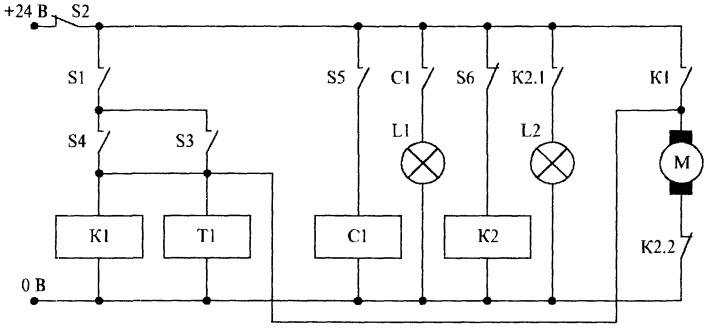


Рис.4.3

4.2.2. Описание входов/выходов (I/O) системы

На рис.4.4 приведена функциональная схема системы управления транспортером с использованием микроконтроллера S7-200, входы которого соединены со следующими элементами системы:

I0.1 Кнопка ВКЛ (замыкающий контакт) – S1.

I0.2 Кнопка ВЫКЛ (размыкающий контакт) – S2.

I0.3 Датчик для регистрации деталей с предшествующей установки (замыкающий контакт) – S3.

I0.4 Датчик для регистрации деталей на ленте (замыкающий контакт) – S4.

I0.5 Датчик, регистрирующий детали в конце ленты (замыкающий контакт) – S5.

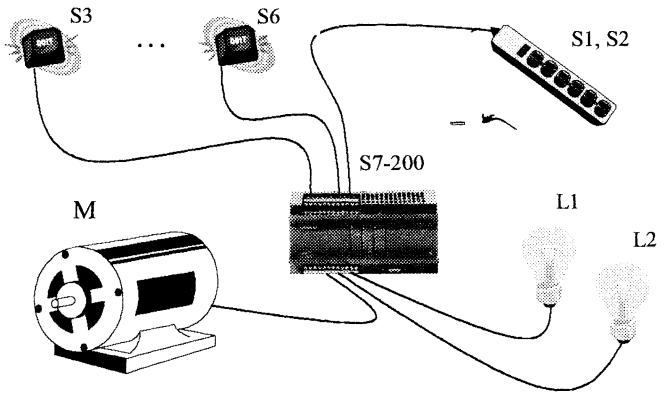


Рис. 4.4

I0.6 Датчик, регистрирующий присутствие человека в зоне ограждения (размыкающий контакт) – S6.

Q0.1 Лента транспортера – M.

Q0.2 Сигнальная лампа – L1.

Q0.3 Сигнальная лампа – L2.

4.3. Решение задачи

Для решения поставленной задачи необходимо выполнить следующие этапы проектирования систем:

- создание проекта,
- присвоение имени проекту,
- создание программы,
- создание таблицы символов,
- ввод программы в форме LAD или STL,
- компиляция программы,
- сохранение примера программы,
- создание таблицы состояний/принудительного задания,
- загрузка проекта в CPU,
- контроль программы,
- наблюдение статуса LAD,
- отображение текущего состояния элементов программы.

Подробные инструкции и описание каждого этапа приведены в разделах 2 и 3.

На рис.4.5 приведена программа управления транспортером на языке релейно-контактных схем (LAD), а на рис.4.6 – на языке STL.

На рис.4.7 приведена таблица символов данной программы, а на рис.4.8 – таблица состояний/принудительного задания.

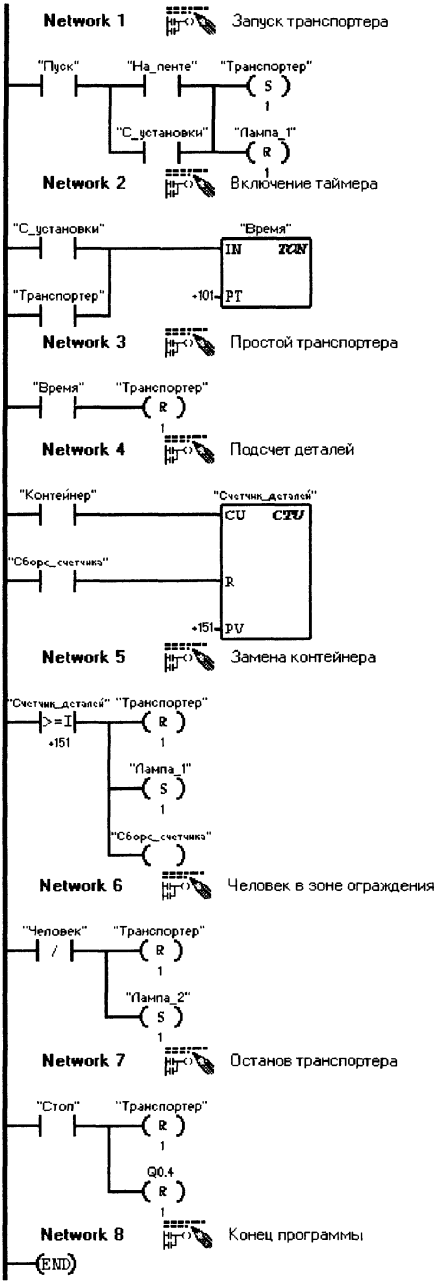

Network 1
Запуск транспортера
"Пуск"
"На_ленте"
"Транспортер"
S
1
"С_установки"
"Лампа_1"
R
1
Network 2
Включение таймера
"С_установки"
"Время"
IN
TON
"Транспортер"
+101
PT
Network 3
Простой транспортера
"Время"
"Транспортер"
R
1
Network 4
Подсчет деталей
"Контейнер"
"Счетчик_деталей"
CU
CTU
"Сброс_счетчика"
R
+151
PV
Network 5
Замена контейнера
"Счетчик_деталей"
"Транспортер"
>=I
+151
R
1
"Лампа_1"
S
1
"Сброс_счетчика"
Network 6
Человек в зоне ограждения
"Человек"
"Транспортер"
/
R
1
"Лампа_2"
S
1
Network 7
Останов транспортера
"Стоп"
"Транспортер"
R
1
Q0.4
R
1
Network 8
Конец программы
END

Рис. 4.5

```
NETWORK 1      //Запуск транспортера
LD      "Пуск"
LD      "На_ленте"
O       "С_установки"
ALD
S       "Транспортер", 1
R       "Лампа_1", 1
NETWORK 2      //Включение таймера
LD      "С_установки"
O       "Транспортер"
TON     "Время", +101
NETWORK 3      //Простой транспортера
LD      "Время"
R       "Транспортер", 1
NETWORK 4      //Подсчет деталей
LD      "Контейнер"
LD      "Сборс_счетчика"
CTU     "Счетчик_деталей", +151
NETWORK 5      //Замена контейнера
LDW>=   "Счетчик_деталей", +151
R       "Транспортер", 1
S       "Лампа_1", 1
=       "Сборс_счетчика"
NETWORK 6      //Человек в зоне ограждения
LDN     "Человек"
R       "Транспортер", 1
S       "Лампа_2", 1
NETWORK 7      //Останов транспортера
LD      "Стоп"
R       "Транспортер", 1
R       Q0.4, 1
NETWORK 8      //Конец программы
MEND
```

Рис. 4.6

Symbol Name	Address	Comment
Пуск	I0.1	Запуск системы
Стоп	I0.2	Останов транспортера
С_установки	I0.3	Детали, поданные с установки
На_ленте	I0.4	Детали, находящиеся на ленте
Транспортер	Q0.1	Включение двигателя транспортера
Лампа_1	Q0.2	Контейнер заполнен
Лампа_2	Q0.3	Человек в зоне ограждения
Время	T37	Таймер
Контейнер	I0.5	Конец ленты транспортера
Счетчик_деталей	C1	Счетчик деталей, достигших контейнера
Человек	I0.6	Чкеловек в зоне ограждения
Сборс_счетчика	M0.0	Сброс счетчика

Рис. 4.7

Address	Format	Current Value	New Value
"Пуск"	Bit	2#0	
"Стоп"	Bit	2#0	
"С_установки"	Bit	2#0	
"На_ленте"	Bit	2#0	
"Транспортер"	Bit	2#0	
"Лампа_1"	Bit	2#0	
"Лампа_2"	Bit	2#1	
"Время"	Signed	+0	
"Контейнер"	Bit	2#0	
"Счетчик_деталей"	Signed	+3	
"Человек"	Bit	2#0	
"Сброс_счетчика"	Bit	2#0	

Рис. 4.8

4.4. Примеры решения задач

Задача 1. Таймер реального времени.

Данная программа использует часы реального времени и позволяет осуществить:

1) ввод текущей даты и времени;
2) вывод текущих значений месяца, числа и часа на дисплей;
3) запуск процесса и отсчета времени;
4) остановку процесса и выдачу сигнала окончания отсчета.

Ввод текущей даты и времени производится при помощи подпрограммы, которая запускается нажатием кнопки «Подпрограмма». В данном примере введены следующие значения даты и времени:

Год – 2001 (01)
Месяц – март (03)
Число – 27
Значение часа – 11
Минуты – 59
Секунды – 30
День недели – вторник (03)

Вывод текущих значений месяца, числа и часа на дисплей осуществляются кнопками «Месяц», «Число», «Час» соответственно в двоично-десятичном коде. Сброс дисплея после отображения производится нажатием кнопки I0.7.

Запуск процесса и одновременно отсчета времени производится нажатием кнопки «Запуск».

Остановка отсчета производится на основе сравнения байтов, хранящих текущее значение даты и времени, и констант даты и времени остановки процесса. Сравнение реализуется на основе команд сравнения на строгое равенство. Все операторы соединены последовательно: месяц – число – час. В данном примере введено следующее время остановки процесса:

Месяц – март (03)

Число – 27

Значение часа – 12

Результатом работы данной программы будет срабатывание через 30 секунд лампы «Индикатор», сигнализирующей об окончании отсчета.

На рис.4.9 приведена программа работы таймера на языке релейно-контактных схем (LAD), а на рис.4.10 – на языке STL.

На рис.4.11 приведена таблица символов данной программы.

Задача 2. Управление ёлочной гирляндой.

Программа позволяет реализовать аналоговое управление скоростью работы гирлянды, а также переключение режимов работы по нажатию на кнопку соответствующего режима.

Задание режимов работы гирлянды производится пересылкой в байт выходов определенного числового значения:

1) 1 – передвижение одной горящей лампочки по цепи;
2) 3 – передвижение двух подряд горящих лампочек по цепи;
3) 204 – передвижение лампочек, горящих две через две, по цепи;
4) 170 – передвижение лампочек, горящих через одну, по цепи.

Управление скоростью передвижения по цепи производится пересылкой значения с аналогового входа в ячейку памяти, используемую как параметр времени таймера.

Переключение режимов работы производится кнопками «Режим 1» ... «Режим 4».

На рис.4.12 приведена таблица символов данной программы.

На рис.4.13 приведена программа управления гирляндой на языке релейно-контактных схем (LAD), а на рис.4.14 – на языке STL.

Задача 3. Преобразователь двоично-десятичного кода в двоичный.

В данной программе реализован алгоритм преобразования двоично-десятичного кода в двоичный.

Входными данными являются две тетрады двоично-десятичного кода (входы I0.0 – I0.7), а выходными – семь разрядов двоичного кода (выходы Q0.0 – Q0.6), на выходе Q0.7 формируется бит контроля четности.

На рис.4.15 приведена таблица символов данной программы.

На рис.4.16 приведена программа преобразования на языке релейно-контактных схем (LAD), а на рис.4.17 – на языке STL.

Задача 4. Преобразователь кода Грея в двоичный код.

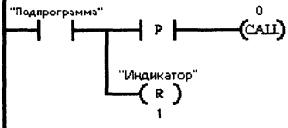
В данной программе реализован алгоритм преобразования кода Грея в двоичный код.

Входными данными являются восемь разрядов кода Грея (входы I0.0–I0.7), а выходными – восемь разрядов двоичного кода (выходы Q0.0 – Q0.6).

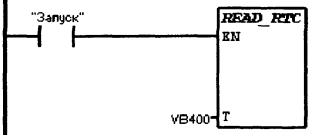
На рис.4.18 приведена программа преобразователя кода Грея на языке релейно-контактных схем (LAD), а на рис.4.19 – на языке STL.

На рис.4.20 приведена таблица символов данной программы.

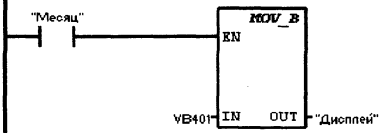
Network 1 Вызов подпрограммы



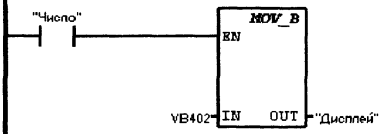
Network 2 Часы реального времени



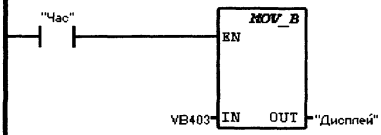
Network 3 Отображение текущего месяца



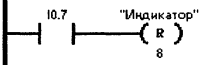
Network 4 Отображение текущего числа



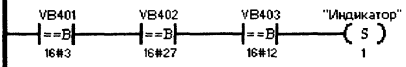
Network 5 Отображение текущего часа



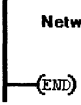
Network 6 Сброс дисплея



Network 7 Индикация окончания процесса



Network 8 Конец программы



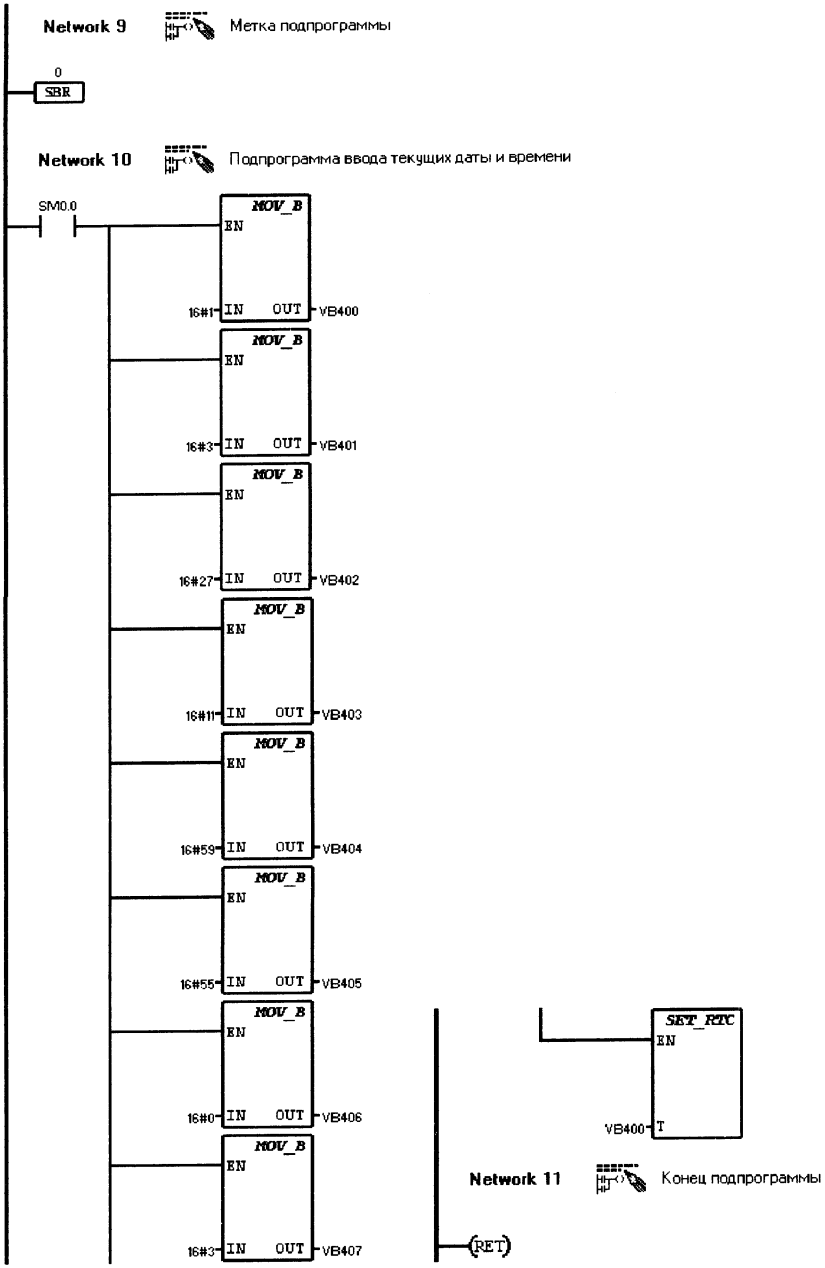

Network 9
Метка подпрограммы
0
SBR
Network 10
Подпрограмма ввода текущих даты и времени
SM0.0
MOV_B
EN
16#1
IN
OUT
VB400
16#3
VB401
16#27
VB402
16#11
VB403
16#59
VB404
16#55
VB405
16#0
VB406
16#3
VB407
SET_RTC
VB400
T
Network 11
Конец подпрограммы
RET

Рис.4.9

```
NETWORK 1 //Вызов подпрограммы
LD      "Подпрограмма"
LPS
EU
CALL    0
LPP
R       "Индикатор", 1
NETWORK 2 //Часы реального времени
LD      "Запуск"
TODR    VB400
NETWORK 3 //Отображение текущего месяца
LD      "Месяц"
MOVB    VB401, "Дисплей"
NETWORK 4 //Отображение текущего числа
LD      "Число"
MOVB    VB402, "Дисплей"
NETWORK 5 //Отображение текущего часа
LD      "Час"
MOVB    VB403, "Дисплей"
NETWORK 6 //Сброс дисплея
LD      I0.7
R       "Индикатор", 8
NETWORK 7 //Индикация окончания процесса
LDB=    VB401, 16#3
AB=     VB402, 16#27
AB=     VB403, 16#12
S       "Индикатор", 1
NETWORK 8 //Конец программы
MEND
NETWORK 9 //Метка подпрограммы
SBR     0
NETWORK 10 //Подпрограмма ввода текущих даты и времени
LD      SM0.0
MOVB    16#1, VB400
MOVB    16#3, VB401
MOVB    16#27, VB402
MOVB    16#11, VB403
MOVB    16#59, VB404
MOVB    16#55, VB405
MOVB    16#0, VB406
MOVB    16#3, VB407
TODW    VB400
NETWORK 11 //Конец подпрограммы
RET
```

Рис.4.10

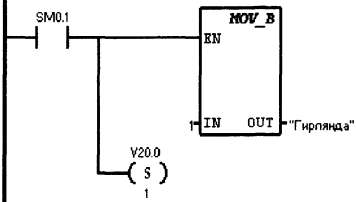
Symbol Name	Address	Comment
Подпрограмма	I0.0	Запуск подпрограммы записи даты и време
Запуск	I0.1	Запуск процесса и часов
Месяц	I0.2	Вывод текущего значения месяца
Число	I0.3	Вывод текущего значения числа
Час	I0.4	Вывод текущего значения часа
Индикатор	Q0.0	Индикация окончания процесса
Дисплей	QB0	Отображение месяца (числа, часа)

Рис.4.11

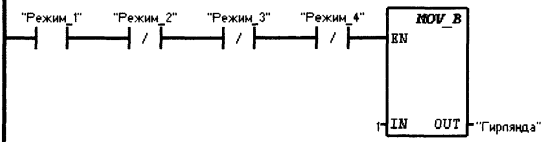
Symbol Name	Address	Comment
Режим_1	I0.0	Кнопка включения режима 1
Режим_2	I0.1	Кнопка включения режима 2
Режим_3	I0.2	Кнопка включения режима 3
Режим_4	I0.3	Кнопка включения режима 4
Гирлянда	QB0	Лампы гирлянды
Частота	AIW0	Регулирования частоты сигнала
Длительность	T32	Задание длительности сигнала

Рис.4.12

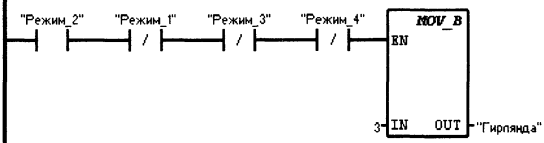
Network 1 Начальная инициализация программы



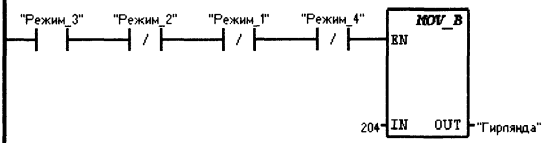
Network 2 Включение режима 1



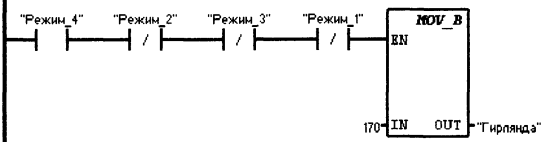
Network 3 Включение режима 2



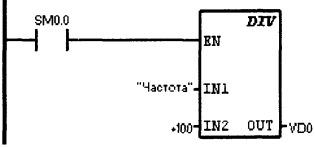
Network 4 Включение режима 3



Network 5 Включение режима 4



Network 6 Считывание значения с AIW0 и деление его на 100



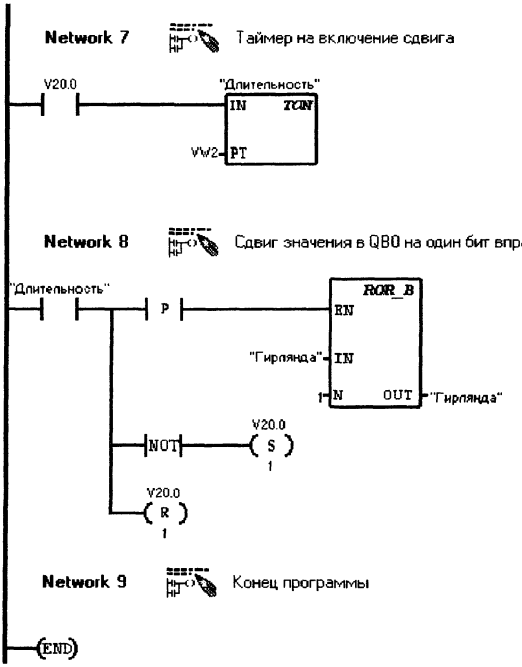

Network 7
Таймер на включение сдвига
V20.0
"Длительность"
IN
TON
VW2
PT
Network 8
Сдвиг значения в QB0 на один бит вправо
"Длительность"
P
ROR_B
EN
"Гирлянда"
IN
1
N
OUT
"Гирлянда"
NOT
V20.0
S
1
V20.0
R
1
Network 9
Конец программы
END

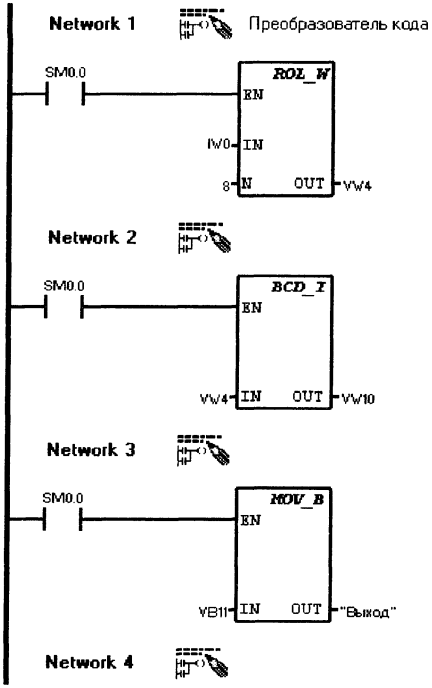
Рис.4.13

```
NETWORK 1 //Начальная инициализация программы
LD      SM0.1
MOVB    1, "Гирлянда"
S       V20.0, 1
NETWORK 2 //Включение режима 1
LD      "Режим_1"
AN      "Режим_2"
AN      "Режим_3"
AN      "Режим_4"
MOVB    1, "Гирлянда"
NETWORK 3 //Включение режима 2
LD      "Режим_2"
AN      "Режим_1"
AN      "Режим_3"
AN      "Режим_4"
MOVB    3, "Гирлянда"
NETWORK 4 //Включение режима 3
LD      "Режим_3"
AN      "Режим_2"
AN      "Режим_1"
AN      "Режим_4"
MOVB    204, "Гирлянда"
NETWORK 5 //Включение режима 4
LD      "Режим_4"
AN      "Режим_2"
AN      "Режим_3"
AN      "Режим_1"
MOVB    170, "Гирлянда"
NETWORK 6 //Считывание значения с AIW0 и деление его на 100
LD      SM0.0
MOVW    "Частота", VW2
DIV     +100, VD0
NETWORK 7 //Таймер на включение сдвига
LD      V20.0
TON     "Длительность", VW2
NETWORK 8 //Сдвиг значения в QB0 на один бит вправо
LD      "Длительность"
LPS
EU
RRB     "Гирлянда", 1
LRD
NOT
S       V20.0, 1
LPP
R       V20.0, 1
NETWORK 9 //Конец программы
MEND
```

Рис. 4.14

Symbol Name	Address	Comment
Вход	IB0	Двоично-десятичный код
Выход	QB0	Двоичный код
Контроль	Q0.7	Бит контроля четности

Рис.4.15



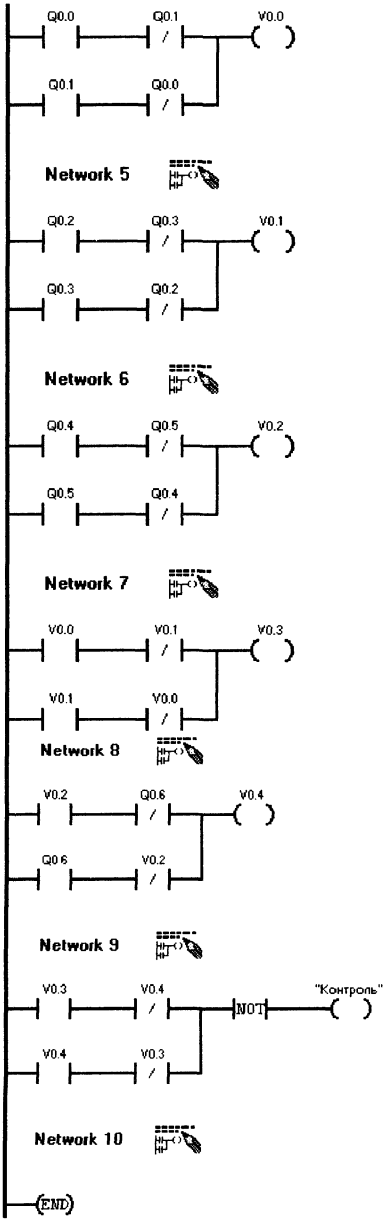

Q0.0
Q0.1
V0.0
Q0.1
Q0.0
Network 5
Q0.2
Q0.3
V0.1
Q0.3
Q0.2
Network 6
Q0.4
Q0.5
V0.2
Q0.5
Q0.4
Network 7
V0.0
V0.1
V0.3
V0.1
V0.0
Network 8
V0.2
Q0.6
V0.4
Q0.6
V0.2
Network 9
V0.3
V0.4
NOT
"Контроль"
V0.4
V0.3
Network 10
END

Рис.4.16

```
NETWORK 1 //Преобразователь кода
LD      SM0.0
MOVW    IW0, VW4
RLW     VW4, 8
NETWORK 2
LD      SM0.0
MOVW    VW4, VW10
BCDI    VW10
NETWORK 3
LD      SM0.0
MOVB    VB11, "Выход"
NETWORK 4
LD      Q0.0
AN      Q0.1
LD      Q0.1
AN      Q0.0
OLD
=       V0.0
NETWORK 5
LD      Q0.2
AN      Q0.3
LD      Q0.3
AN      Q0.2
OLD
=       V0.1
NETWORK 6
LD      Q0.4
AN      Q0.5
LD      Q0.5
AN      Q0.4
OLD
=       V0.2
NETWORK 7
LD      V0.0
AN      V0.1
LD      V0.1
AN      V0.0
OLD
=       V0.3
NETWORK 8
LD      V0.2
AN      Q0.6
LD      Q0.6
AN      V0.2
OLD
=       V0.4
NETWORK 9
LD      V0.3
AN      V0.4
LD      V0.4
AN      V0.3
OLD
NOT
=       "Контроль"
NETWORK 10
MEND
```

Рис.4.17

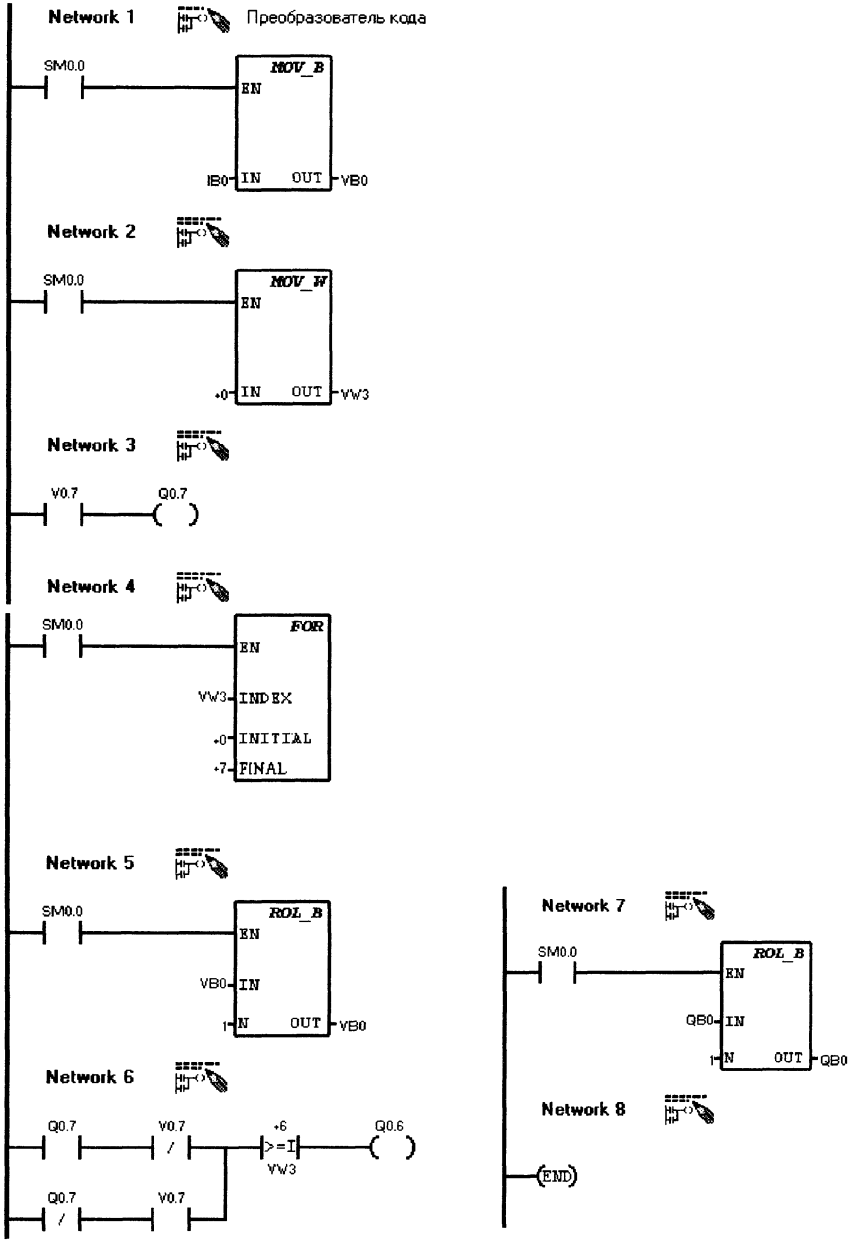

Network 1
Преобразователь кода
SM0.0
MOV_B
EN
IB0
IN
OUT
VB0
Network 2
SM0.0
MOV_W
EN
+0
IN
OUT
VW3
Network 3
V0.7
Q0.7
Network 4
SM0.0
FOR
EN
VW3
INDEX
+0
INITIAL
+7
FINAL
Network 5
SM0.0
ROL_B
EN
VB0
IN
1
N
OUT
VB0
Network 6
Q0.7
V0.7
/
+6
>=I
VW3
Q0.6
Q0.7
/
V0.7
Network 7
SM0.0
ROL_B
EN
QB0
IN
1
N
OUT
QB0
Network 8
END

Рис.4.18

```
NETWORK 1 //Преобразователь кода
LD      SM0.0
MOVB    IB0, VB0
NETWORK 2
LD      SM0.0
MOVW    +0, VW3
NETWORK 3
LD      V0.7
=       Q0.7
NETWORK 4
LD      SM0.0
FOR     VW3, +0, +7
NETWORK 5
LD      SM0.0
RLB     VB0, 1
NETWORK 6
LD      Q0.7
AN      V0.7
LDN     Q0.7
A       V0.7
OLD
AW>=    +6, VW3
=       Q0.6
NETWORK 7
LD      SM0.0
RLB     QB0, 1
NETWORK 8
MEND
```

Рис.4.19

Symbol Name	Address	Comment
Вход	IB0	Код Грея
Выход	QB0	Двоичный код

Рис.4.20

ЗАКЛЮЧЕНИЕ

В учебном пособии рассмотрены вопросы проектирования систем автоматизации с использованием программируемых логических контроллеров на примере микроконтроллера S7-200 с процессором CPU-214.

Приведены технические характеристики данного микроконтроллера и подробные инструкции по программированию.

Рассмотрены примеры решения конкретных задач, при этом задачи подобраны таким образом, чтобы максимально продемонстрировать возможности программного обеспечения STEP 7–Micro/WIN 32.

В библиографическом списке приведен список литературы, знакомство с которой поможет глубже разобраться в рассматриваемых здесь вопросах.

К сожалению, объем учебного пособия не позволил глубже рассмотреть некоторые темы. Так за рамками издания остались: операции с быстрыми счетчиками и быстрыми выходами, коммуникационные операции.

Однако даже в таком виде учебное пособие должно оказать неоценимую помощь в изучении программируемых логических контроллеров и освоении методов проектирования систем управления различными производственными процессами и техническими системами.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

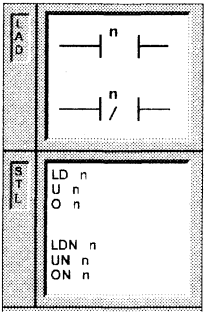
1. *Мишель Ж.* Программируемые контроллеры. Архитектура и применение. Пер. с франц.– М.: Машиностроение, 1992.– 320 с.
2. *Ремизевич Т.В.* Современные программируемые логические контроллеры. // Приводная техника.– 1999.– № 1-2.– С. 8-20.
3. *Ремизевич Т.В.* Современные программируемые логические контроллеры. // Приводная техника.– 1999.– № 3-4.– С. 6-17.
4. *Митин Г.П.* Как выбрать программируемый логический контроллер. // Мир компьютерной автоматизации.– 2000.– № 1.– С. 66-69.
5. SIMATIC Components for Totally Integrated Automation. Catalog ST 70, 1997. Order No.: E86060-K4670-A111-A3-7600.
6. *Митин Г.П., Хазанова О.В.* Микроконтроллеры в системах автоматизации: Учебное пособие.– М.: МГТУ «Станкин», 2001.– 108 с.
7. *Митин Г.П.* Опыт использования микроконтроллеров в учебном процессе. // Доклады международной конференции «Информационные средства и технологии», т. 3, Москва, 2001.– С. 95-98.
8. *Митин Г.П., Погонин А.А., Хазанова О.В.* Решение задач автоматизации с использованием программируемых логических контроллеров: Учебное пособие.– М.: МГТУ «Станкин», 2001.– 119 с.
9. *Митин Г.П.* Микроконтроллеры в учебном процессе. //Автоматизация и управление в машиностроении.– 2002.– № 18.

ПРИЛОЖЕНИЕ

НАБОР ОПЕРАЦИЙ

1.1. Операции над контактами

Стандартные контакты



Замыкающий контакт замкнут (включен), если значение бита с адресом n равно 1.

В STL замыкающий контакт представляется операциями *Загрузка, Логическое И* и *Логическое ИЛИ.* Эти операции загружают значение бита с адресом n в вершину стека или логически связывают значение бита с значением в вершине стека через «И» или «ИЛИ».

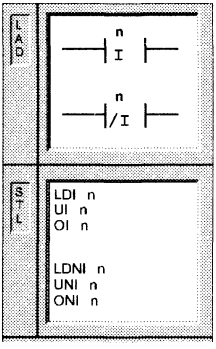
Размыкающий контакт замкнут (включен), если значение бита с адресом n равно 0.

В STL размыкающий контакт представляется операциями *Загрузка логического отрицания бита, Логическое И-НЕ* и *Логическое ИЛИ-НЕ.* Эти операции загружают логическое отрицание бита с адресом n в вершину стека или логически связывают логическое отрицание бита с значением в вершине стека через "И" или "ИЛИ".

Операнды n: I, Q, M, SM, T, C, V.

При активизации в начале цикла CPU эти операции получают заданное значение из области отображения процесса.

Контакты с непосредственным доступом



Замыкающий контакт с непосредственным доступом замкнут (включен), если значение бита заданного входа n равно 1.

В STL замыкающий контакт с непосредственным доступом представляется операциями *Непосредственная загрузка, Непосредственное логическое И* и *Непосредственное логическое ИЛИ.* Эти операции загружают непосредственное значение бита заданного входа n в вершину стека или логически связывают это значение бита с значением в вершине стека через «И» или «ИЛИ».

Размыкающий контакт с непосредственным доступом замкнут (включен), если значение бита заданного входа n равно 0.

В STL размыкающий контакт с непосредственным доступом представляется операциями *Непосредственная загрузка логического отрицания*

бита, Непосредственное логическое И логического отрицания бита и *Непосредственное логическое ИЛИ логического отрицания бита*. Эти операции загружают логическое отрицание непосредственного значения бита с адресом n в вершину стека или логически связывают логическое отрицание этого бита с значением в вершине стека через "И" или "ИЛИ".

Операнды n: I.

Операция прямого доступа считывает опрашиваемое значение из физического входа, когда она выполняется. Регистр отображения процесса не актуализируется.

Сравнение байтов

Операция *Сравнение байтов* сравнивает два значения n1 и n2 друг с другом. Можно производить следующие сравнения: n1 = n2, n1 ≥ n2 и n1 ≤ n2.

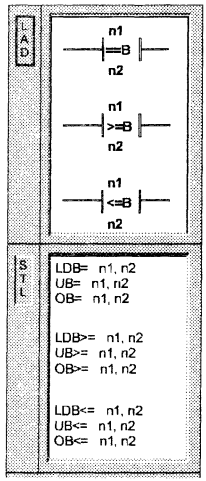
Операнды n1, n2: VB, IB, QB, MB, SMB, AC, константа, *VD, *AC, SB.

В LAD контакт замкнут, если результатом сравнения является истина.

В STL эти операции загружают значение "1" в вершину стека или логически связывают значение "1" с значением в вершине стека через "И" или "ИЛИ", если результатом сравнения является истина.

Сравнения байтов не учитывают знак.

Можно выполнять сравнения ≠, < и >, используя операцию NOT совместно с операциями ≥, = или ≤.. Две следующие операции соответствуют сравнению ≠ между VB100 и значением 50: LDB = VB100, 50 NOT



Сравнение слов

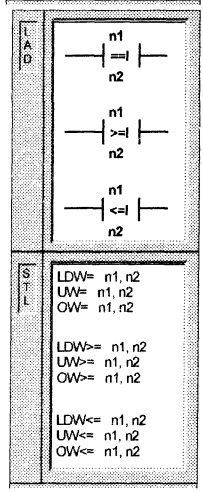
Операция *Сравнение слов* сравнивает два значения n1 и n2 друг с другом. Можно производить следующие сравнения: n1 = n2, n1 ≥ n2 и n1 ≤ n2.

Операнды n1, n2: VW, T, C, IW, QW, MW, SMW, AC, AIW, константа, *VD, *AC, SW.

В LAD контакт замкнут, если результатом сравнения является истина.

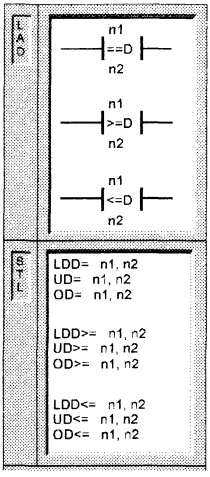
В STL эти операции загружают значение «1» в вершину стека или логически связывают значение «1» с значением в вершине стека через «И» или «ИЛИ», если результатом сравнения является истина.

Сравнения слов учитывают знак (16#7FFF > 16#8000).



Можно выполнять сравнения ≠, < и >, используя операцию NOT совместно с операциями ≥, = или ≤. Две следующие операции соответствуют сравнению ≠ между VW100 и значением 50: LDW = VW100, 50 NOT

Сравнение двойных слов



Операция *Сравнение двойных слов* сравнивает два значения n1 и n2 друг с другом. Можно производить следующие сравнения: n1 = n2, n1 ≥ n2 и n1 ≤ n2.

Операнды n1, n2: VD, ID, QD, MD, SMD, AC, HC, константа, *VD, *AC, SD.

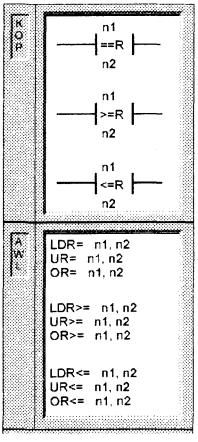
В LAD контакт замкнут, если результатом сравнения является истина.

В STL эти операции загружают значение «1» в вершину стека или логически связывают значение «1» с значением в вершине стека через «И» или «ИЛИ», если результатом сравнения является истина.

Сравнения двойных слов учитывают знак (16#7FFFFFFF > 16#80000000).

Можно выполнять сравнения ≠, < и >, используя операцию NOT совместно с операциями ≥, = или ≤. Две следующие операции соответствуют сравнению ≠ между VD100 и значением 50: LDD = VD100, 50 NOT

Сравнение действительных чисел



Операция *Сравнение действительных чисел* сравнивает два значения n1 и n2 друг с другом. Можно производить следующие сравнения: n1 = n2, n1 ≥ n2 и n1 ≤ n2.

Операнды n1, n2: VD, ID, QD, MD, SMD,AC, константа, *VD, *AC, SD.

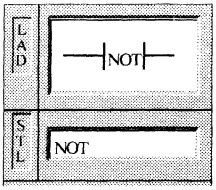
В LAD контакт замкнут, если результатом сравнения является истина.

В STL эти операции загружают значение «1» в вершину стека или логически связывают значение «1» с значением в вершине стека через «И» или «ИЛИ», если результатом сравнения является истина.

Сравнения действительных чисел учитывают знак.

Можно выполнять сравнения ≠, < и >, используя операцию NOT совместно с операциями ≥, = или ≤. Две следующие операции соответствуют сравнению ≠ между VD100 и значением 50: LDR = VD100, 50 NOT

NOT

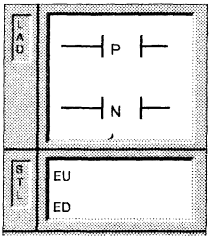


Контакт *NOT* изменяет состояние потока сигнала.

В STL операция **NOT** изменяет вершину стека с «0» на «1» или с «1» на «0».

Операнды: нет.

Нарастающий фронт и спадающий фронт



Контакт *Обнаружение нарастающего фронта* пропускает поток сигнала в течение цикла при каждом нарастающем фронте.

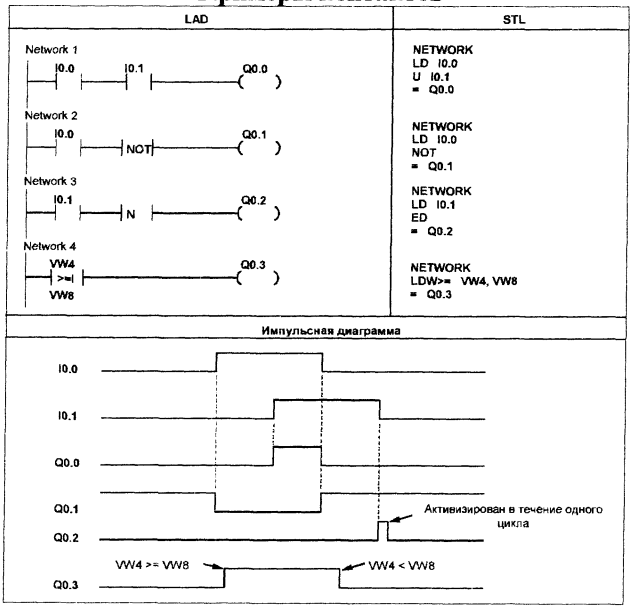
В STL операция *Обнаружение нарастающего фронта* устанавливает вершину стека в «1», если в вершине стека обнаруживается нарастающий фронт (смена с «0» на «1»). Если нарастающий фронт не обнаруживается, то вершина стека устанавливается в «0».

Контакт *Обнаружение спадающего фронта* пропускает поток сигнала в течение цикла при каждом спадающем фронте.

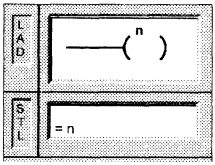
В STL операция *Обнаружение спадающего фронта* устанавливает вершину стека в «1», если в вершине стека обнаруживается спадающий фронт (смена с «1» на «0»). Если спадающий фронт не обнаруживается, то вершина стека устанавливается в «1».

Операнды: нет.

Примеры контактов



1.2. Операции над выходами



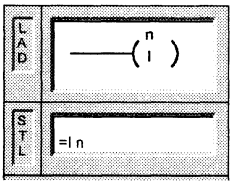
Присваивание

Если выполняется операция *Присваивание*, то заданный параметр (n) включается.

В STL операция *Присваивание* копирует вершину стека в заданный параметр (n).

Операнды n: I, Q, M, SM, T, C, V, S.

Прямое присваивание значения биту



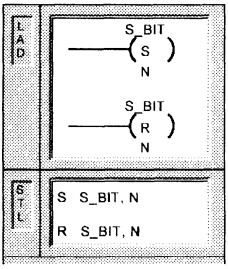
Если выполняется операция *Прямое присваивание значения биту*, то заданный физический выход (n) непосредственно включается.

В STL операция *Прямое присваивание значения биту копирует* вершину стека непосредственно в заданный физический выход (n).

Операнды n: Q.

Знак «I» указывает на прямой доступ. При выполнении операции новое значение записывается как в область отображения процесса, так и непосредственно на физический выход. В этом прямая операция отличается от других, в которых значение для адресуемого входа или выхода записывается только в область отображения процесса.

Установка и сброс

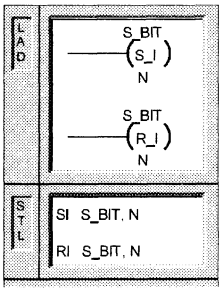


Если выполняются операции *Установка* и *Сброс*, то заданное количество (N) входов или выходов, начиная с S_BIT, устанавливается (включается) или сбрасывается (выключается).

Операнды S_BIT: I, Q, M, SM, T, C, V, S

N: IB, QB, MB, SMB, VB, AC, константа, *VD, *AC, SB.

Область входов или выходов, которые могут устанавливаться или сбрасываться, находится в диапазоне от 1 до 255. Если в операции «Сброс» в качестве параметра S_BIT задан бит таймера или счетчика, то сбрасывается как бит таймера/счетчика, так и текущее значение таймера или счетчика.



Прямая установка и сброс

Если выполняются операции *Прямая установка* и *Прямой сброс*, то заданное количество (N) входов или выходов, начиная с S_BIT, устанавливается (включается) или сбрасывается (выключается).

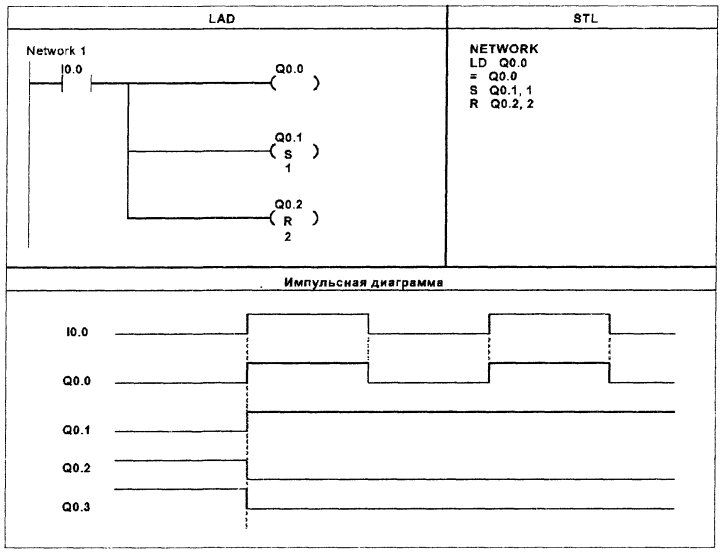
Операнды S_BIT: Q

N: IB, QB, MB, SMB, VB, AC, константа, *VD, *AC, SB.

Область входов или выходов, которые могут устанавливаться или сбрасываться, находится в диапазоне от 1 до 64.

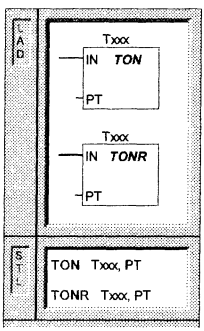
«I» указывает на прямой доступ. При выполнении операции новое значение записывается как в область отображения процесса, так и непосредственно на физический выход. В этом прямая операция отличается от других, в которых значение для адресуемого входа или выхода записывается только в область отображения процесса.

Пример операций над выходами



1.3. Таймерные операции и операции со счетчиками

Запуск таймера как формирователя задержки включения и запуск таймера как формирователя задержки включения с запоминанием



Операции *Запуск таймера как формирователя задержки включения* и *Запуск таймера как формирователя задержки включения с запоминанием* ведут отсчет времени до максимального значения времени, если они активизируются. Если текущее значение (Txxx) больше или равно чем предварительно установленное значение (PT), то включается бит таймера.

Если операция «Запуск таймера как формирователя задержки включения» деактивизируется, то таймер сбрасывается. Если операция «Запуск таймера как формирователя задержки включения с запоминанием» деактивизируется, то таймер останавливается. Оба таймера останавливаются, когда они достигают максимального значения.

Операнды Txxx: TON, TONR

PT: VW, T, C, IW, QW, MW, SMW, AC, AEW, константа, *VD, *AC, SW.

Таймеры TON и TONR доступны с тремя разными разрешающими способностями (см. табл.3.2). Каждый отсчет в текущем значении представляет величину, кратную разрешающей способности таймера. Например, для таймера с разрешающей способностью 10 мс отсчет 50 соответствует текущему значению 500 мс. Максимальное значение отсчета (PT) для всех таймеров – 32.767.

Описание таймерных операций для S7–200

С помощью таймеров можно выполнять функции, управляемые временем. Оба таймера (TON и TONR) отсчитывают значение времени, когда активизирован вход разблокировки. При выключенном входе разблокировки оба таймера не работают, однако в то время как таймер TON автоматически сбрасывается, таймер TONR сохраняет свое последнее значение времени, а не сбрасывается. Поэтому лучше всего использовать таймер TON, когда нужен отдельный интервал времени, и использовать таймер TONR, когда нужно накапливать несколько интервалов времени.

Если Вы сбрасываете таймер, то текущее значение таймера устанавливается в нуль, а бит таймера выключается. Операцию «Сброс» можно использовать для сброса любого таймера, но эта операция является единственной, с помощью которой можно сбросить таймер TONR. Запись нуля в текущее значение таймера не сбрасывает бит таймера. Запись нуля в бит таймера не сбрасывает его текущее значение.

С помощью таймеров с разрешающей способностью 1 мс можно создать событие прерывания.

Так как накопление интервалов происходит независимо от разблокировки и блокировки таймеров, то таймеры разблокируются внутри определенного интервала времени. Этот интервал для таймера может иметь максимальную длительность, равную разрешающей способности. Следует задавать в предварительной установке значение, которое на 1 больше, чем самый короткий желаемый интервал. Например, для того, чтобы иметь в распоряжении интервал величиной не менее 140 мс при использовании таймера с разрешающей способностью 10 мс, нужно задать предварительно устанавливаемое значение времени равным 15.

Активизация таймеров с разрешающей способностью 1 мс

CPU S7–200 имеет таймеры, которые активизируются один раз в миллисекунду системной программой, ответственной за активизацию базы времени. Эти таймеры служат для точного управления операцией.

Текущее значение активного таймера с разрешающей способностью 1 мс автоматически активизируется системной программой. После разблокировки таймера выполнение операций TON/TONR для таймера с разрешающей способностью 1 мс требуется только для того, чтобы управлять состоянием таймера ВКЛ/ВЫКЛ.

Так как текущее значение и бит таймера в таймере с разрешающей способностью 1 мс активизируются системной программой (независимо от цикла контроллера и от программы пользователя), то текущее значение и Т–бит такого таймера могут активизироваться в произвольной точке цикла. Они могут активизироваться в течение одного цикла также многократно, если время цикла превышает одну миллисекунду. Поэтому не гарантируется, что эти значения остаются постоянными во время обработки главной программы.

Активизация таймеров с разрешающей способностью 10 мс

CPU S7–200 имеет таймеры, которые подсчитывают количество интервалов величиной 10 мс, истекших после разблокировки активного таймера с разрешающей способностью 10 мс. Эти таймеры активизируются в начале каждого цикла путем прибавления количества истекших интервалов величиной 10 мс (с начала предыдущего цикла) к текущему значению таймера.

Текущее значение активного таймера с разрешающей способностью 10 мс автоматически активизируется в начале каждого цикла. Если таймер разблокирован, то операции TON/TONR для таймера с разрешающей способностью 10 мс выполняются только для того, чтобы управлять состоянием таймера ВКЛ/ВЫКЛ. В противоположность таймерам с разрешающей способностью 1 мс, текущее значение таймера с разрешающей способностью 10 мс активизируется только один раз за цикл и не изменяется во время обработки программы пользователя.

Активизация таймеров с разрешающей способностью 100 мс

Большинство таймеров в контроллерах S7–200 имеют разрешающую способность 100 мс. Эти таймеры подсчитывают количество интервалов величиной 100 мс, прошедших с момента последней активизации таймера. Таймеры с разрешающей способностью 100 мс активизируются путем прибавления накопленного количества 100-миллисекундных интервалов (считая от начала предыдущего цикла) к текущему значению таймера всякий раз, когда выполняется таймерная операция.

Активизация таймера с разрешающей способностью 100 мс происходит не автоматически, так как текущее значение таймера активизируется только тогда, когда выполняется таймерная операция. Поэтому текущее значение таймера не активизируется, если даже таймер и разблокируется, но таймерная операция выполняется не в каждом цикле. Вследствие этого таймер теряет время. Если одна и та же таймерная операция выполняется для таймера с разрешающей способностью 100 мс в каждом цикле многократно, то накопленное значение многократно прибавляется к текущему значению таймера. Вследствие этого таймер получает дополнительное время. Следовательно, таймеры с разрешающей способностью 100 мс должны использоваться только тогда, когда таймерная операция выполняется ровно один раз за цикл.

Активизация текущего значения таймера

Воздействие, оказываемое различной активизацией текущих значений таймеров, определяется тем, как используют таймеры. Рассмотрим, например, таймерную операцию на рис.П.1.

Если используется таймер с разрешающей способностью 1 мс, то Q0.0 включается на время цикла всякий раз, когда текущее значение таймера активизируется после исполнения команды «Размыкающий контакт T32» и перед исполнением команды «Замыкающий контакт T32».

Если используется таймер с разрешающей способностью 10 мс, то Q0.0 не включается ни разу, потому что бит таймера T33 включается с начала цикла до момента времени, когда выполняется таймерная операция. После выполнения таймерной операции текущее значение таймера и бит таймера устанавливаются в нуль. Если выполняется команда «Замыкающий контакт T33», то T33 не активизирован и Q0.0 выключается.

Если используется таймер с разрешающей способностью 100 мс, то Q0.0 включается на время цикла всегда, когда текущее значение таймера достигает предварительно установленного значения.

При использовании размыкающего контакта Q0.0 вместо бита таймера в качестве входа разблокировки таймера гарантируется, что выход Q0.0 включается на время цикла всякий раз, когда таймер достигает предварительно установленного значения (см. рис.П.1).

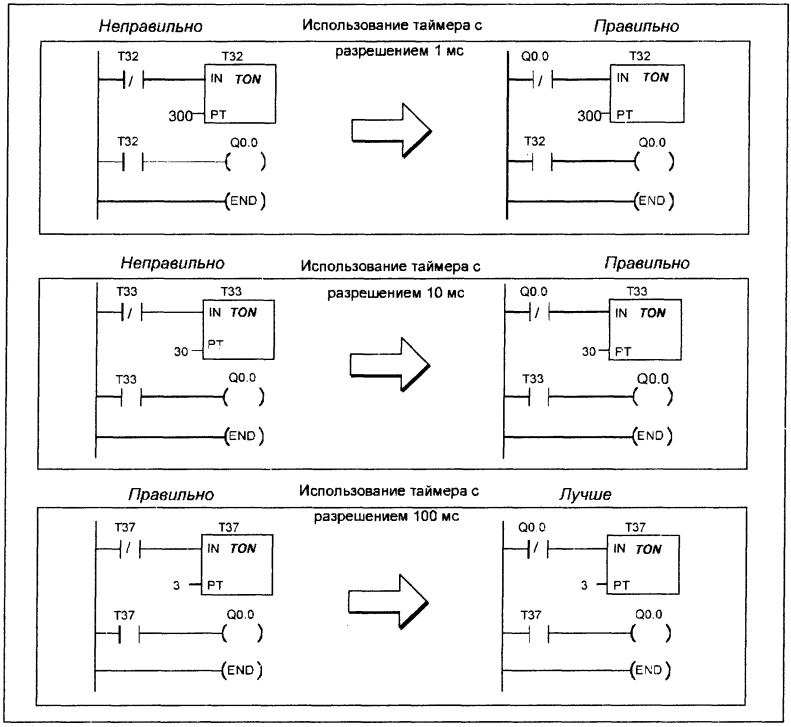
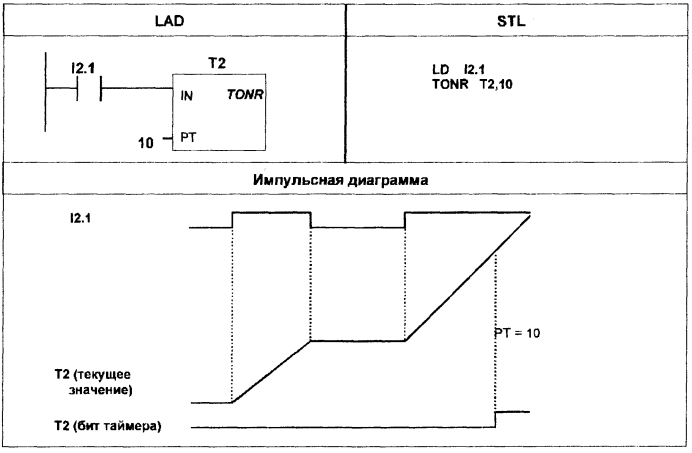


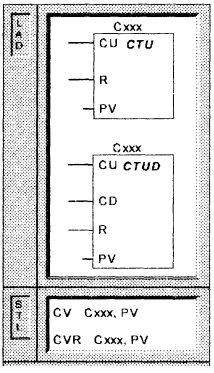
Рис.П.1

Примеры операций с таймерами: «Запуск таймера как формирователя задержки включения»

«Запуск таймера как формирователя задержки включения с запоминанием»



Прямой счет и реверсивный счет



Операция *Прямой счет* выполняет счет в прямом направлении при нарастающем фронте на входе прямого счета (CU) до достижения максимального значения. Если текущее значение (Cxxx) больше или равно предварительно установленному значению (PV), то включается бит счетчика (Cxxx). Счетчик сбрасывается, если активизируется вход сброса.

В STL вход сброса находится в вершине стека, а вход прямого счета представляет собой значение, загружаемое на второй уровень стека.

Операция *Реверсивный счет* выполняет счет в прямом направлении при нарастающем фронте на входе прямого счета (CU). При нарастающем фронте на входе обратного счета (CD) операция выполняет счет в обратном направлении. Если текущее значение (Cxxx) >= предварительно установленному значению (PV), то включается бит счетчика (Cxxx). Счетчик сбрасывается, если активизируется вход сброса.

В STL вход сброса находится в вершине стека. Вход прямого счета представляет собой значение на втором уровне стека, а вход обратного счета - значение на третьем уровне стека.

Операнды Cxxx: от 0 до 255

PV: VW, T, C, IW, QW, MW, SMW, AC, AIW, константа, *VD, *AC, SW.

Описание операций со счетчиками S7–200

Прямой счетчик (CTU) при нарастающем фронте на входе прямого счета выполняет отсчет вперед от текущего значения счетчика. Счетчик сбрасывается, если активизируется вход сброса или выполняется операция «Сброс». Счетчик останавливается, когда достигнуто максимальное значение (32.767).

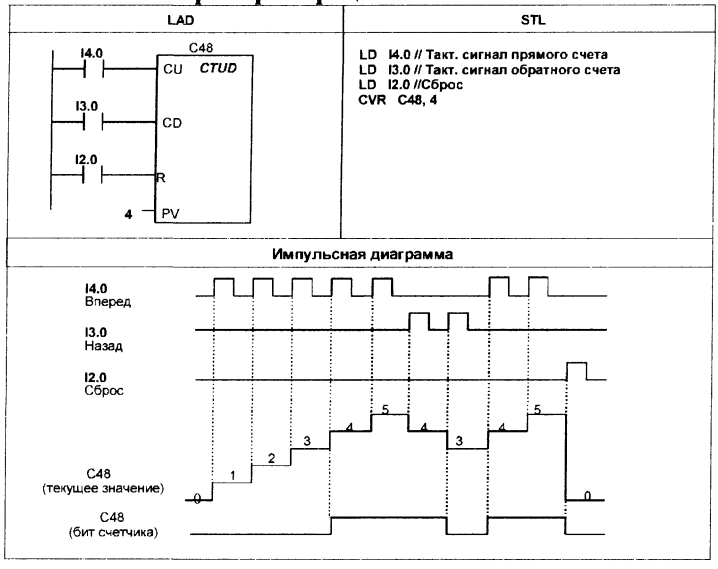
Реверсивный счетчик (CTUD) при нарастающем фронте на входе прямого счета выполняет счет вперед, а при нарастающем фронте на входе обратного счета выполняет счет назад. Счетчик сбрасывается, если активизируется вход сброса или выполняется операция «Сброс». Если достигается максимальное значение (32.767), то следующий нарастающий фронт на входе прямого счета вызывает «опрокидывание» счетчика, и счет начинается снова с минимального значения (–32.767). Если при счете достигается минимальное значение (–32.767), то при следующем нарастающем фронте на входе обратного счета счетчик «опрокидывается» и считает дальше с максимального значения (32.767).

Если Вы сбрасываете счетчик посредством операции «Сброс», то сбрасывается как бит счетчика, так и текущее значение счетчика.

Обращение к текущему значению, а также к биту счетчика осуществляется с помощью номера счетчика.

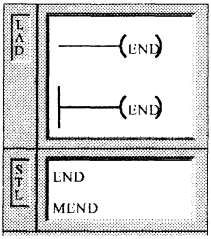
Так как каждый счетчик обладает собственным текущим значением, то не назначайте одинаковый номер нескольким счетчикам.

Пример операций со счетчиком



1.4. Операции управления программой

END



Операция *Условное окончание обработки* заканчивает обработку главной программы в зависимости от результата предшествующей логической операции.

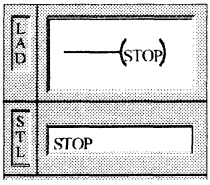
Главная программа должна заканчиваться катушкой *Абсолютное окончание обработки.*

В STL операция *Абсолютное окончание обработки* представляется командой MEND.

Операнды: нет.

Все программы пользователя должны завершать главную программу операцией «Абсолютное окончание обработки». Операция «Условное окончание обработки» используется, когда обработка должна завершаться до операции «Абсолютное окончание обработки».

STOP

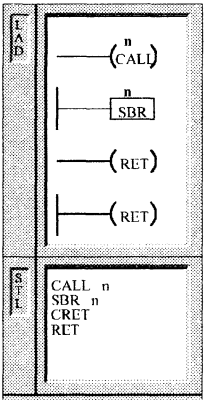


Операция *STOP* заканчивает обработку программы пользователя немедленно, переводя CPU из режима работы RUN в режим работы STOP.

Операнды: нет.

Если операция STOP выполняется в программе обработки прерываний, то последняя немедленно завершается и все стоящие в очереди прерывания игнорируются. Остаток программы обрабатывается, и в конце цикла CPU переходит в режим STOP.

Вызов, начало и окончание подпрограммы



Операция *Вызов подпрограммы* передает управление подпрограмме (n).

Операция *Начало подпрограммы* отмечает начало подпрограммы (n).

Операция *Условное окончание подпрограммы* завершает подпрограмму в зависимости от результата предшествующей логической операции.

Любая подпрограмма должна заканчиваться операцией *Абсолютное окончание подпрограммы.*

Операнды n: от 0 до 63

Если обработка подпрограммы закончена, то главная программа выполняется дальше с той операции, которая следует за операцией CALL.

Можно производить вложение подпрограмм (внутри одной подпрограммы вызывать другую под-

программу) с глубиной вложения, равной максимум восьми уровням. Допустима рекурсия (подпрограмма вызывает саму себя), однако применять рекурсию следует с осторожностью.

При вызове подпрограммы весь стек запоминается, вершина стека устанавливается в «1», все остальные значения в стеке устанавливаются в «0», и обрабатывается вызванная подпрограмма. Если обработка этой подпрограммы закончена, то восстанавливается стек со значениями, сохраненными в момент вызова подпрограммы. После этого вызывающая программа обрабатывается дальше.

Таким образом, когда вызывается подпрограмма, то самый верхний элемент стека всегда равен «1». Поэтому можно в сегменте, следующим за операцией SBR, подключать выходы и блоки непосредственно к левой шине тока. В STL операция загрузки, следующая за операцией SBR, может опускаться.

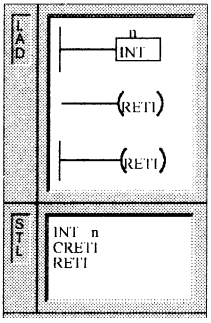
Аккумуляторы используются как главной программой, так и подпрограммами. Вызов подпрограммы не приводит к запоминанию и последующему восстановлению аккумуляторов.

Пример операций над подпрограммами

LAD	STL
Network 1 SM0.1 10 (CALL) В первом цикле: вызов SBR10 для инициализации. Network 39 (END) Все подпрограммы нужно располагать после операции END. Network 50 10 SBR Начало подпрограммы 10. Network 65 M14.3 (RET) Подпрограмма 10 может заканчиваться условно ? RET) Network 68 (RET) Любая подпрограмм должна заканчиваться абсолютно (RET).Здесь завершается подпрограмма 10.	Network LD SM0.1 CALL 10 Network MEND Network SBR 10 Network LD M14.3 CRET Network RET

1.5. Прерывания

Начало и окончание программы обработки прерываний



Операция *Начало программы обработки прерываний* отмечает начало программы обработки прерываний (n).

Операция *Условное окончание программы обработки прерываний* заканчивает программу обработки прерываний в зависимости от предшествующего логического сопряжения.

Каждая программа обработки прерываний должна заканчиваться операцией *Абсолютное окончание программы обработки прерываний.*

Операнды n: от 0 до 127.

Программы обработки прерываний

Можно снабдить каждую программу обработки прерываний меткой прерывания, отмечающей начало программы. Программа обработки прерываний состоит из операций, которые располагаются между меткой прерывания и операцией абсолютного окончания программы. Вы можете закончить программу (и тем самым снова передать управление главной программе), выполняя операцию «Абсолютное окончание программы обработки прерываний» (RETI) или операцию «Условное окончание программы обработки прерываний». Каждая программа обработки прерываний должна завершаться операцией «Абсолютное окончание программы обработки прерываний».

Указания по использованию программ обработки прерываний

С помощью обработки прерываний можно реагировать на особые внутренние или внешние события. Программу обработки прерываний следует строить таким образом, чтобы она выполняла определенную задачу, а затем снова передавала управление главной программе. Программируйте как можно более короткие программы обработки прерываний с точными указаниями, так чтобы эти программы могли обрабатываться быстро и другие процессы прерывались ненадолго. Пренебрежение этими указаниями может привести к непредвиденным состояниям, способным нарушить работу оборудования, управляемого главной программой. Для программ обработки прерываний девиз «чем короче, тем лучше» определенно верен.

Ограничения

При работе с программами обработки прерываний обратите внимание на следующие указания:

- Присоединяйте все программы обработки прерываний к концу главной программы LAD.

- В программах обработки прерываний нельзя использовать операции DISI, ENI, CALL, HDEF, FOR/NEXT, LSCR, SCRE, SCRT и END.
- Заканчивайте каждую программу обработки прерываний абсолютно (операция RETI).

Системная поддержка прерываний

Контакты, катушки и аккумуляторы могут испытывать воздействие прерываний. Поэтому система запоминает стек, аккумуляторы и специальные маркеры (SM), отображающие состояние аккумуляторов и команд, и загружает их позже снова. Благодаря этому предотвращается нарушение главной программы вследствие перехода к программе обработки прерываний и возврата из нее.

Совместное использование данных в главной программе и программе обработки прерываний

Можно совместно использовать данные в главной программе и в одной или нескольких программах обработки прерываний. Так, например, часть главной программы может предоставлять данные, которые обрабатываются программой обработки прерываний, и наоборот. Если главная программа и программа обработки прерываний совместно используют данные, то необходимо осознавать тот факт, что события прерываний происходят асинхронно по отношению к главной программе. Поэтому они могут появляться в любой момент времени обработки главной программы. Вследствие выполнения программ обработки прерываний могут возникать нарушения целостности совместно используемых данных, когда обработка операций в главной программе прерывается событиями прерываний.

Существует ряд способов программирования, предотвращающих ошибки при совместном использовании данных в главной программе и программах обработки прерываний. Эти способы ограничивают доступ к совместно используемым данным или создают последовательности команд, которые обращаются к совместно используемым данным и не могут прерываться.

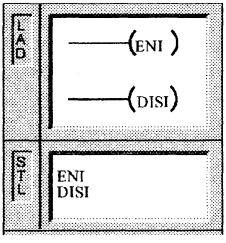
Для программы на STL, которая совместно использует одну единственную переменную: если совместно используется только одна переменная в виде байта, слова или двойного слова и программа написана в форме STL, то промежуточные результаты операций с совместно используемыми данными следует записывать только по адресам памяти или в аккумуляторах, которые не используются совместно.

Для программы на LAD, которая совместно использует одну единственную переменную: если совместно используется только одна переменная в виде байта, слова или двойного слова и программа написана в форме LAD, то обращение к совместно используемым адресам памяти следует производить только с помощью операций передачи (MOV_B, MOV_W,

MOV_DW, MOV_R). Многие операции в LAD соответствуют последовательностям команд в STL, которые могут прерываться. Однако каждая из этих операций передачи соответствует одной единственной команде STL, обработка которой не может испытывать воздействия событий прерываний.

Для программ на STL или LAD, которые совместно использует несколько переменных: если совместно используются нескольких взаимосвязанных байтов, слов или двойных слов, то исполнением программы обработки прерываний можно управлять посредством операций «Блокировка всех событий прерываний» (DISI) и «Разблокировка всех событий прерываний» (ENI). В том месте главной программы, где расположены операции, выполняющие доступ к совместной памяти, необходимо блокировать события прерываний. После того, как выполнены все операции, работающие с совместной памятью, нужно снова разблокировать события прерываний. В течение времени, когда события прерываний блокированы, программы обработки прерываний не могут выполняться и не имеют доступа к совместной памяти. Однако, такой способ программирования может вызывать замедленную реакцию на события прерывания.

Разблокировка и блокировка всех событий прерываний

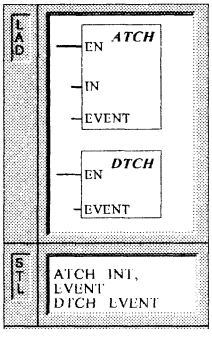


Операция *Разблокировка всех событий прерываний* разблокирует обработку всех назначенных событий прерываний.

Операция *Блокировка всех событий прерываний* блокирует обработку всех событий прерываний.

Операнды: нет.

При переходе в режим RUN прерывания блокируются. Если CPU находится в режиме RUN, то можно с помощью операции ENI разблокировать все события прерываний. Команда «Блокировка всех событий прерываний» допускает постановку прерываний в очередь, но не разрешает вызывать программы обработки прерываний.



Назначение прерывания и отделение прерывания

Операция *Назначение прерывания* назначает событию прерывания (EVENT) номер программу обработки прерываний (INT) и затем разблокирует это событие.

Операция *Отделение прерывания* отделяет событие прерывания (EVENT) от всех программ обработки прерываний и затем блокирует это событие.

Операнды INT: от 0 до 127

EVENT: от 0 до 20

Описание операций назначения и отделения прерываний

Прежде чем вызывать программу обработки прерываний, нужно организовать связь между событием прерывания и частью программы, которую Вы хотите обрабатывать при появлении события прерывания. С помощью операции «Назначение прерывания» (ATCH) назначают событию прерывания (характеризуемому номером события) часть программы (характеризуемую номером программы обработки прерываний). Можно сопоставить одной программе обработки прерываний несколько событий прерываний. Однако одно событие прерывания не может быть одновременно назначено нескольким программам обработки прерываний. Если при разблокированных прерываниях появляется событие, то выполняется только одна программа обработки прерываний, которая была позже всех назначена этому событию.

Если сопоставляют событие прерывания программе обработки прерываний, то это событие автоматически разблокируется. Если выполняют операцию «Блокировка всех событий прерываний», то все возникающие прерывания становятся в очередь до тех пор, пока снова не отменить блокировку прерываний с помощью операции «Разблокировка всех событий прерываний».

Можно блокировать отдельные события прерываний, отменяя сопоставление события программе с помощью операции DTCH (отделение прерывания). Эта операция устанавливает прерывание в неактивное состояние, в котором оно пропускается и потому не обрабатывается.

Прерывания коммуникационных портов

Последовательным коммуникационным портом контроллера можно управлять с помощью программы на LAD или STL. Коммуникация через такой порт называется свободно программируемой коммуникацией. В случае свободно программируемой коммуникации программа определяет скорость передачи данных, количество битов на символ, контроль четности и протокол. Прерывания передачи и приема облегчают коммуникацию с программным управлением.

Прерывания от ввода/вывода

К прерываниям от ввода/вывода относятся прерывания при нарастающем или спадающем фронте, прерывания от быстрых счетчиков и прерывания от последовательности импульсов. CPU-214 может создавать прерывание при нарастающем и/или спадающем фронте на входах и выходах от I0.0 до I0.3. События «Нарастающий фронт» и «Спадающий фронт» могут восприниматься по каждому из этих входов. С помощью этих событий могут также отображаться сбойные ситуации, которые должны сразу приниматься во внимание при появлении события.

С помощью прерываний от быстрых счетчиков можно реагировать на следующие события: совпадение текущего значения с предварительно установленным значением, смена направления счета на обратное по отношению к направлению вращения вала, внешний сброс счетчика. С помощью каждого из этих событий в быстрых счетчиках можно реагировать на быстрые события, которыми невозможно управлять с частотой циклов контролера.

Прерывания от последовательности импульсов сразу уведомляют об окончании вывода заданного количества импульсов. Последовательности импульсов часто используются для управления шаговыми двигателями.

Можно разблокировать описанные выше прерывания, назначая программу обработки прерываний соответствующему событию ввода/вывода.

Прерывания, управляемые временем

CPU-214 может поддерживать два прерывания, управляемых временем. С помощью этих прерываний можно определять действия, которые должны выполняться периодически. Период задается с шагом 1 мс, значения лежат в диапазоне от 5 мс до 255 мс. Период для управляемого временем прерывания 0 запишите в SMB34, период для управляемого временем прерывания 1 запишите в SMB35.

Управляемое временем событие прерывания вызывает соответствующую программу обработки прерываний каждый раз, когда истекает время. В общем случае с помощью управляемых временем событий прерываний Вы управляете регулярным опросом аналоговых входов.

Управляемое временем прерывание разблокируется и время начинает отсчитываться, когда назначают программу обработки прерываний управляемому временем событию прерывания. При этом система фиксирует период, чтобы последующие изменения не влияли на период. Если хотите изменить период, то нужно задать новое значение для периода и затем снова назначить программу обработки прерываний управляемому временем событию прерывания. При новом назначении эта функция стирает накопленное значение времени предыдущего назначения, и время начинает отсчитываться с новым значением периода.

После разблокировки прерывание, управляемое временем, функционирует непрерывно и обрабатывается каждый раз, когда истекает заданный интервал времени. Если выходите из режима RUN или отделяете прерывание от программы обработки прерываний (DTCH), то управляемое временем прерывание блокируется. Если выполняете операцию «Блокировка всех событий прерываний», то управляемые временем прерывания в дальнейшем хотя и появляются, однако ставятся в очередь (до тех пор, пока прерывания снова не разблокируются, либо очередь не переполнится).

Управляемые временем прерывания T32/T96 служат для управляемого временем реагирования на истечение заданного интервала времени.

Эти прерывания поддерживаются только формирователями задержки включения (TON) с разрешающей способностью 1 мс - T32 и T96. В противном случае таймеры T32 и T96 имеют обычные функциональные возможности. Когда прерывание разблокировано, назначенная программа обработки прерываний выполняется, если при актуализации таймеров с разрешающей способностью 1 мс в цикле CPU окажется, что текущее значение активного таймера равно предварительно установленному значению таймера. Вы разблокируете эти прерывания, назначая программу обработки прерываний событию прерывания T32/T96.

Приоритеты прерываний и очереди

Приоритеты прерываний назначаются согласно следующей схеме:

- коммуникационные прерывания - высший приоритет,
- прерывания от ввода/вывода (включая HSC и последовательности импульсов),
- управляемые временем прерывания - низший приоритет

Прерывания обрабатываются контроллером в пределах соответствующих им приоритетов в последовательности из появления. Всегда активна только одна программа обработки прерываний. Если в данный момент обрабатывается программа обработки прерываний, то эта программа доводится до конца. Она не может прерываться программой обработки прерываний, появляющейся позже, даже если приоритет этой программы выше. Прерывания, возникающие во время обработки другого прерывания, принимаются в очередь и обрабатываются позже. В табл.П.1 показаны три очереди для прерываний и максимальное количество прерываний, которые могут приниматься в каждую очередь.

Таблица П.1

Очередь	CPU 212	CPU 214	CPU 215	CPU 216
Коммуникационные прерывания	4	4	4	8
Прерывания от ввода/вывода	4	16	16	16
Управляемые временем прерывания	2	4	8	8

В принципе может появиться больше прерываний, чем сможет принять очередь. Поэтому система имеет в своем распоряжении маркеры переполнения очередей, указывающие вид событий прерываний, которые не смогли быть приняты в очередь. Табл.П.2 поясняет специальные маркеры, которые устанавливаются при переполнении очереди. Их можно использовать только в одной программе обработки прерываний, так как они сбрасываются, когда очередь обработана и снова начинается обработка главной программы.

Таблица П.2

Описание (0 = нет переполнения, 1 = переполнение)	Специальный маркер
Переполнение очереди для коммуникационных прерываний	SM4.0
Переполнение очереди для прерываний от ввода/вывода	SM4.1
Переполнение очереди для прерываний, управляемых временем	SM4.2

В табл.П.3 показаны событие прерывания, приоритет и назначенный номер прерывания.

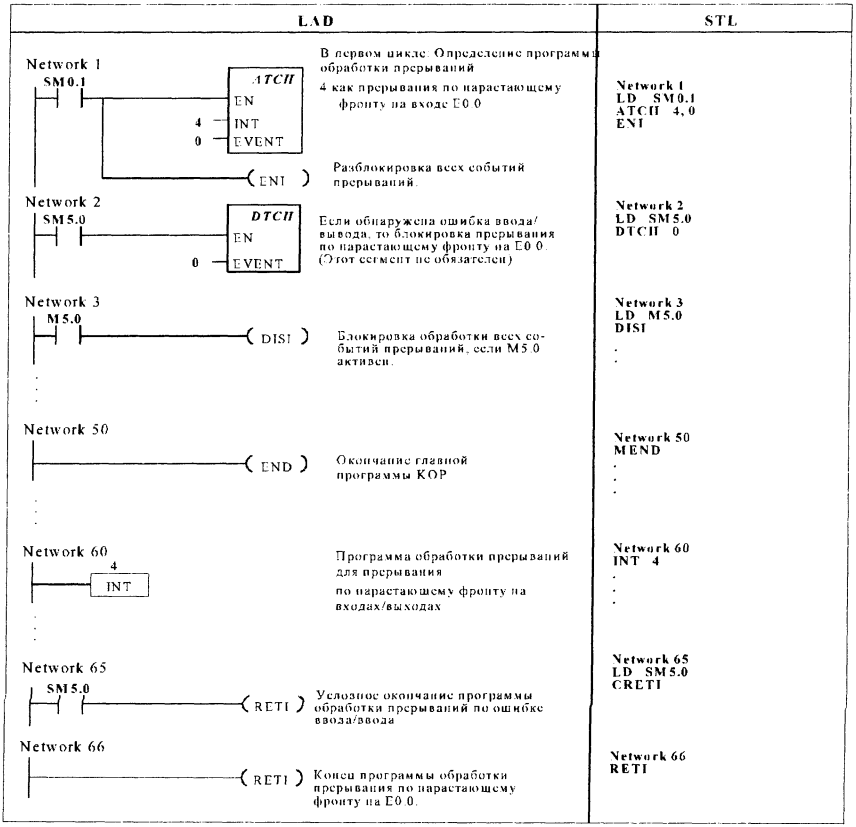
Таблица П.3

Номер события	Описание прерывания	Класс приоритета	Приоритет
8	Порт 0: Принят символ	Коммуникационные прерывания: высший класс приоритета	0
9	Порт 0: Закончена передача		0*
23	Порт 0: Закончен прием сообщения		0*
24	Порт 1: Закончен прием сообщения		1
25	Порт 1: Принят символ		1*
26	Порт 1: Закончена передача		1*
0	Нарастающий фронт, I0.0**	Прерывания от ввода/вывода: средний класс приоритета	0
2	Нарастающий фронт, I0.1		1
4	Нарастающий фронт, I0.2		2
6	Нарастающий фронт, I0.3		3
1	Спадающий фронт, I0.0**		4
3	Спадающий фронт, I0.1		5
5	Спадающий фронт, I0.2		6
7	Спадающий фронт, I0.3		7
12	HSC0: CV = PV (текущее значение = предварительно установленному значению)**		0
13	HSC1: CV = PV (текущее значение = предварительно установленному значению)		8
14	HSC1: Смена направления счета		9
15	HSC1: Внешний сброс		10
16	HSC2: CV = PV (текущее значение = предварительно установленному значению)		11
17	HSC2: Смена направления счета		12
18	HSC2: Внешний сброс		13
19	PLS0: Счет импульсов завершен		14
20	PLS1: Счет импульсов завершен		15
10	Управляемое временем прерывание 0	Управляемые временем прерывания: низший класс приоритета	0
11	Управляемое временем прерывание 1		1
21	Таймер T32: Прерывание CT = PT		2
22	Таймер T96: Прерывание CT = PT		3

* Так как коммуникация является полудуплексной, то прерывания передачи и приема имеют одинаковый приоритет.

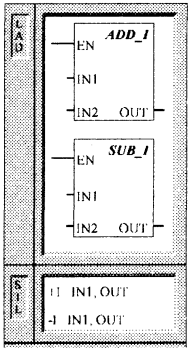
** Если событие 12 (HSC0, PV = CV) сопоставлено прерыванию, то события 0 и 1 не могут быть сопоставлены никакому прерыванию. Если одно из событий 0 и 1 сопоставлено прерыванию, то событие 12 не может быть сопоставлено никакому прерыванию.

Пример операций в программе обработки прерываний.



Пример считывания значения аналогового входа с помощью прерывания, управляемого временем.

LAD	STL
Главная программа	
Network 1 SM0.1 — 0 (CALL) Маркер первого цикла: Вызов подпрограммы 0.	Network 1 LD SM0.1 CALL 0
Network 2 (END)	Network 2 MEND
Подпрограмма	
Network 3 0 SBR Начало подпрограммы 0.	Network 3 SBR 0
Network 4 SM0.0 MOV_B EN 100 — IN OUT — SMB34 Маркер постоянно включен: Установка интервала 100 мс для управляемого временем прерывания 0.	Network 4 LD SM0.0 MOVB 100, SMB34
(ENI) Разблокировка всех событий прерывания	ENI
ATCH EN 0 — INT 10 — EVENT Назначение программы обработки прерываний 0 управляемому временем прерыванию 0.	ATCH 0, 10
Network 5 (RET) Окончание подпрограммы.	Network 5 RET
Программа обработки прерываний	
Network 6 0 INT Начало программы обработки прерываний 0.	Network 6 INT 0
Network 7 MOV_W EN AIW4 — IN OUT — VW100 Опрос AEW4.	Network 7 MOVW AEW4, VW100
Network 8 (RETI) Окончание программы обработки прерываний.	Network 8 RETI



1.6. Арифметические операции, инкрементирование и декрементирование

Сложение и вычитание целых чисел (16 бит) [1]

Операции *Сложение целых чисел (16 бит)* и *Вычитание целых чисел (16 бит)* складывают или вычитают два целых числа (16 бит) и передают результат (16 бит) в OUT.

Операнды IN1, IN2: VW, T, C, IW, QW, MW, SMW, AC, AIW, константа, *VD, *AC, SW

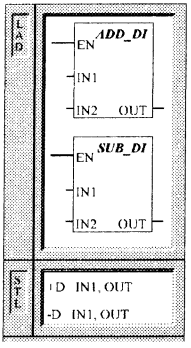
OUT: VW, T, C, IW, QW, MW, SMW, AC, *VD, *AC, SW

LAD: IN1 + IN2 = OUT , IN1 - IN2 = OUT

STL: IN1 + OUT = OUT , OUT - IN1 = OUT

Эти операции влияют на специальные маркеры: SM1.0 (нуль); SM1.1 (переполнение); SM1.2 (отрицательный)

Сложение и вычитание целых чисел (32 бита) [1]



Операции *Сложение целых чисел (32 бита)* и *Вычитание целых чисел (32 бита)* складывают или вычитают два целых числа (32 бита) и передают результат (32 бита) в OUT.

Операнды IN1, IN2: VD, ID, QD, MD, SMD, AC, HC, константа, *VD, *AC, SD

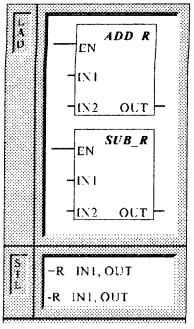
OUT: VD, ID, QD, MD, SMD, AC, *VD, *AC, SD

LAD: IN1 + IN2 = OUT, IN1 - IN2 = OUT

STL: IN1 + OUT = OUT, OUT - IN1 = OUT

Эти операции влияют на специальные маркеры: SM1.0 (нуль); SM1.1 (переполнение); SM1.2 (отрицательный)

Сложение и вычитание действительных чисел



Операции *Сложение действительных чисел* и *Вычитание действительных чисел* складывают или вычитают два действительных числа (32 бита) и передают действительное число в качестве результата в OUT.

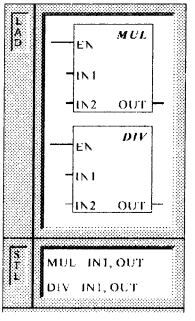
Операнды IN1, IN2: VD, ID, QD, MD, SMD, AC, константа, *VD, *AC, SD

OUT: VD, ID, QD, MD, SMD, AC, *VD, *AC, SD

LAD: IN1 + IN2 = OUT, IN1 - IN2 = OUT

STL: IN1 + OUT = OUT, OUT - IN1 = OUT

Умножение и деление целых чисел (16 бит) [1]



Операция *Умножение целых чисел (16 бит)* умножает два целых числа (16 бит) и передает результат (32 бита) в OUT.

В AWL младшее слово (16 бит) значения OUT (32 бита) используется как один из сомножителей.

Операция *Деление целых чисел (16 бит)* делит два целых числа (16 бит) и передает результат (32 бита) в OUT. Результат (32 бита) в OUT состоит из частного (16 младших битов) и остатка от деления (16 старших битов).

В STL младшее слово (16 бит) значения OUT (32 бита) используется как делимое.

Операнды IN1, IN2: VW, T, C, IW, QW, MW, SMW, AC, AEW, константа, *VD, *AC, SW

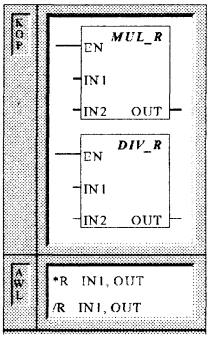
OUT: VD, ID, QD, MD, SMD, AC, *VD, *AC, SW

LAD: IN1 * IN2 = OUT, IN1 / IN2 = OUT

STL: IN1 * OUT = OUT, OUT / IN1 = OUT

Эти операции влияют на специальные маркеры: SM1.0 (нуль); SM1.1 (переполнение); SM1.2 (отрицательный); SM1.3 (деление на нуль).

Умножение и деление действительных чисел [1]



Операция *Умножение действительных чисел* умножает два действительных числа (32 бита) и передает результат (32 бита) в OUT.

Операция *Деление действительных чисел* делит два действительных числа (32 бита) и передает результат (32 бита) в OUT.

Операнды IN1, IN2: VD, ID, QD, MD, SMD, AC, константа, *VD, *AC, SD

OUT: VD, ID, QD, MD, SMD, AC, *VD, *AC, SD

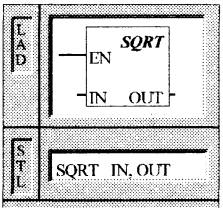
LAD: IN1 * IN2 = OUT, IN1/ IN2 = OUT

STL: IN1 * OUT = OUT, OUT / IN1 = OUT

Эти операции влияют на специальные маркеры: SM1.0 (нуль); SM1.1 (переполнение); SM1.2 (отрицательный); SM1.3 (деление на нуль)

Если устанавливается SM1.1 или SM1.3, то другие биты состояния для арифметических операций и первичные входные операнды не изменяются.

Извлечение квадратного корня из действительного числа



Операция *Извлечение квадратного корня из действительного числа* извлекает квадратный корень из действительного числа (32 бита), заданного в IN. Результат (OUT) тоже является действительным числом (32 бита).

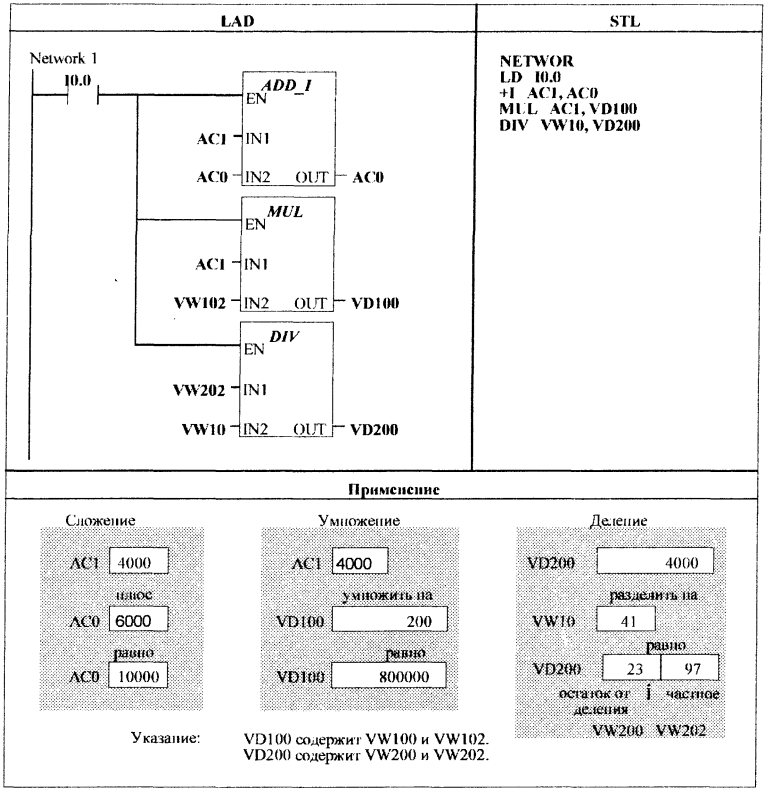
$\sqrt{IN} = OUT$

Операнды IN: VD, ID, QD, MD, SMD, AC, константа, *VD, *AC, SD

OUT: VD, ID, QD, MD, SMD AC, *VD, *AC, SD

Эти операции влияют на специальные маркеры: SM1.0 (нуль); SM1.1 (переполнение); SM1.2 (отрицательный)

Примеры арифметических операций



Инкрементирование и декрементирование слова [1]

Операции *Увеличить слово на 1* и *Уменьшить слово на 1* прибавляет или вычитает «1» из значения входного слова.

Операнды IN: VW, T, C, IW, QW, MW, SMW, AC, AIW, константа, *VD, *AC, SW

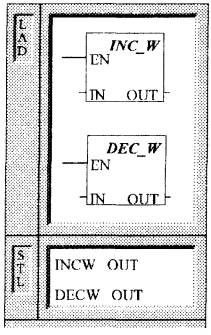
OUT: VW, T, C, IW, QW, MW, SMW, AC, *VD, *AC, SW

LAD: IN + 1 = OUT, IN - 1 = OUT

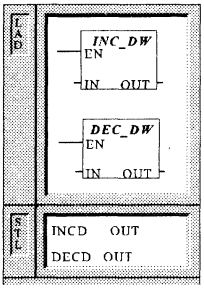
STL: OUT + 1 = OUT, OUT - 1 = OUT

Операции «Увеличить слово на 1» и «Уменьшить слово на 1» учитывают знак (16#7FFF > 16#8000).

Эти операции влияют на специальные маркеры: SM1.0 (нуль); SM1.1 (переполнение); SM1.2 (отрицательный)



Инкрементирование и декрементирование двойного слова



Операции *Увеличить двойное слово на 1* и *Уменьшить двойное слово на 1* прибавляет или вычитает «1» из значения входного двойного слова.

Операнды IN: VD, ID, QD, MD, SMD, AC, HC, константа, *VD, *AC, SD

OUT: VD, ID, QD, MD, SMD, AC, *VD, *AC, SD

LAD: IN + 1 = OUT, IN - 1 = OUT

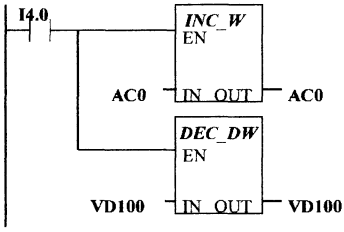
STL: OUT + 1 = OUT, OUT - 1 = OUT

Операции «Увеличить двойное слово на 1» и «Уменьшить двойное слово на 1» учитывают знак (16#7FFFFFFF > 16#80000000).

Эти операции влияют на специальные маркеры: SM1.0 (нуль); SM1.1 (переполнение); SM1.2 (отрицательный)

Пример инкрементирования и декрементирования

LAD



STL

```
LD  I4.0
INCW  AC0
DECD  VD100
```

Применение

Увеличить слово на 1

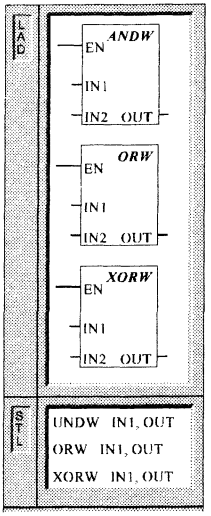
AC0	125	
увеличить на 1		
AC0	126	

Уменьшить двойное слово на 1

VD100	128000	
уменьшить на 1		
VD100	127999	

1.7. Логические операции

Логическое сопряжение слов через И, ИЛИ и ИСКЛЮЧАЮЩЕЕ ИЛИ [1]



Операция *Логическое сопряжение слов через И* логически связывает через И соответствующие биты двух входных слов и загружает результат (OUT) в слово.

Операция *Логическое сопряжение слов через ИЛИ* логически связывает через ИЛИ соответствующие биты двух входных слов и загружает результат (OUT) в слово.

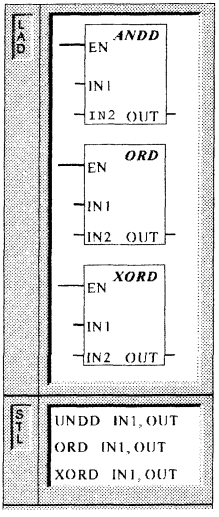
Операция *Логическое сопряжение слов через ИСКЛЮЧАЮЩЕЕ ИЛИ* логически связывает через *ИСКЛЮЧАЮЩЕЕ ИЛИ* соответствующие биты двух входных слов и загружает результат (OUT) в слово.

Операнды IN1, IN2:VW, T, C, IW, QW, MW, SMW, AC, AIW, константа, *VD, *AC, SW

OUT: VW, T, C, IW, QW, MW, SMW, AC, *VD, *AC, SW

Эти операции влияют на специальные маркеры: SM1.0 (нуль)

Логическое сопряжение двойных слов через И, ИЛИ и ИСКЛЮЧАЮЩЕЕ ИЛИ [1]



Операция *Логическое сопряжение двойных слов через И* логически связывает через И соответствующие биты двух входных двойных слов и загружает результат (OUT) в двойное слово.

Операция *Логическое сопряжение двойных слов через ИЛИ* логически связывает через ИЛИ соответствующие биты двух входных двойных слов и загружает результат (OUT) в двойное слово.

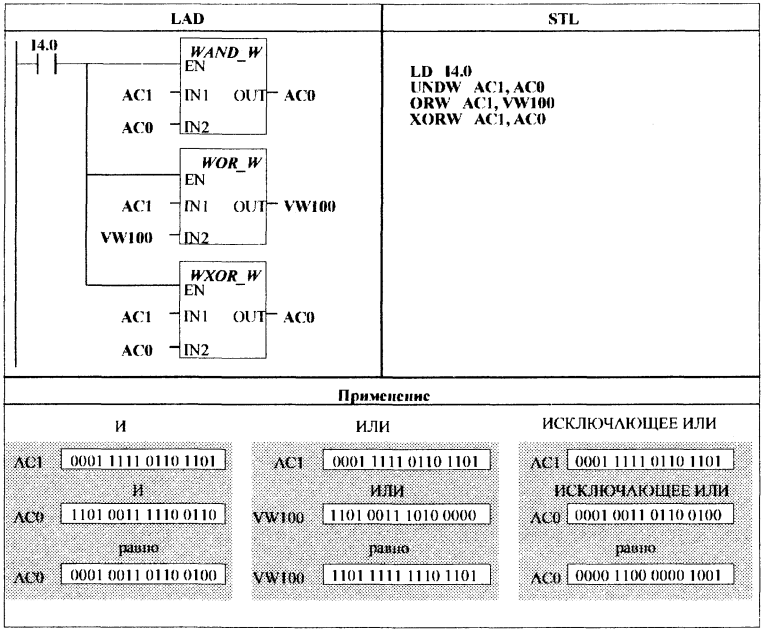
Операция *Логическое сопряжение двойных слов через ИСКЛЮЧАЮЩЕЕ ИЛИ* логически связывает через ИСКЛЮЧАЮЩЕЕ ИЛИ соответствующие биты двух входных двойных слов и загружает результат (OUT) в двойное слово.

Операнды IN1, IN2: VD, ID, QD, MD, SMD, AC, HC, константа, *VD, *AC, SD

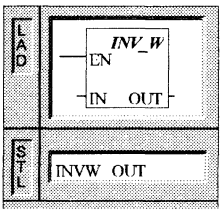
OUT: VD, ID, QD, MD, SMD, AC, *VD,*AC, SD

Эти операции влияют на специальные маркеры: SM1.0 (нуль)

Примеры логических операций



Образование дополнения до единицы для целого числа (16 бит) [1]



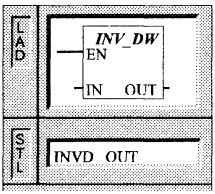
Операция *Образование дополнения до единицы для целого числа (16 бит)* образует дополнение до единицы для значения входного слова и загружает результат в слово (OUT).

Операнды IN: VW, T, C, IW, QW, MW, SMW, AC,AIW, константа, *VD, *AC, SW

OUT: VW, T, C, IW, QW, MW, SMW, AC,*VD, *AC, SW

Эти операции влияют на специальные маркеры: SM1.0 (нуль)

Образование дополнения до единицы для целого числа (32 бита) [1]



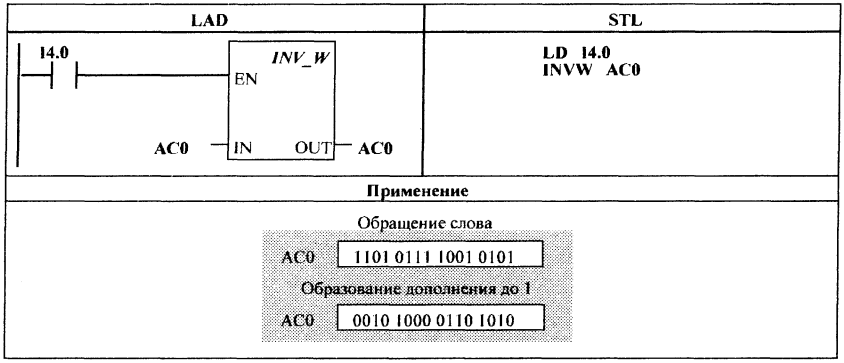
Операция *Образование дополнения до единицы для целого числа (32 бита)* образует дополнение до единицы для значения входного двойного слова и загружает результат в двойное слово (OUT).

Операнды IN: VD, ID, QD, MD, SMD, AC, HC, константа, *VD, *AC, SD

OUT: VD, ID, QD, MD, SMD, AC, *VD,*AC, SD

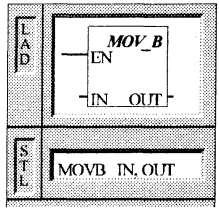
Эти операции влияют на специальные маркеры: SM1.0 (нуль)

Пример операции дополнения до единицы



1.8. Операции передачи, сдвига и циклического сдвига

Передача байта

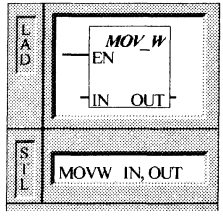


Операция *Передача байта* передает входной байт (IN) в выходной байт (OUT). Входной байт при этом не изменяется.

Операнды IN: VB, IB, QB, MB, SMB, AC, константа, *VD, *AC, SB

OUT: VB, IB, QB, MB, SMB, AC, *VD*AC, SB

Передача слова

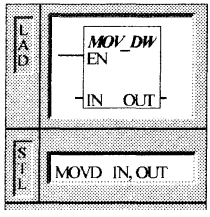


Операция *Передача слова* передает входное слово (IN) в выходное слово (OUT). Входное слово при этом не изменяется.

Операнды IN: VW, T, C, IW, QW, MW, SMW, AC, AIW, константа, VD, *AC, SW

OUT: VW, T, C, IW, QW, MW, SMW, AC, AQW, *VD, *AC, SW

Передача двойного слова

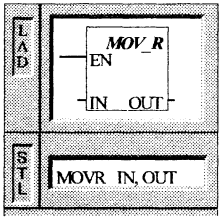


Операция *Передача двойного слова* передает входное двойное слово (IN) в выходное двойное слово (OUT). Входное двойное слово при этом не изменяется.

Операнды IN: VD, ID, QD, MD, SMD, AC, HC, константа, *VD, *AC, &VB, &IB,&QB, &MB, &T, &C, &SB, SD

OUT: VD, ID, QD, MD, SMD, AC, *VD, *AC, SD

Передача действительного числа

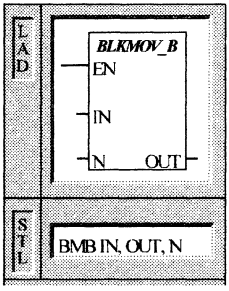


Операция *Передача действительного числа* передает входное действительное число (двойное слово 32 бита) (IN) в выходное двойное слово (OUT). Входное двойное слово при этом не изменяется.

Операнды IN: VD, ID, QD, MD, SMD, AC, HC, константа, *VD, *AC,SD

OUT: VD, ID, QD, MD, SMD, AC, *VD, *AC, SD

Блочная передача байтов

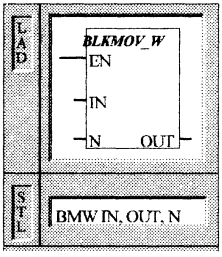


Операция *Блочная передача байтов* передает заданное количество байтов (N) из входного массива, начинающегося с IN, в выходной массив, начинающейся с OUT. N может лежать в диапазоне от 1 до 255.

Операнды IN, OUT: VB, IB, QB, MB, SMB, *VD, *AC, SB

N: VB, IB, QB, MB, SMB, AC, константа, *VD, *AC, SB

Блочная передача слов

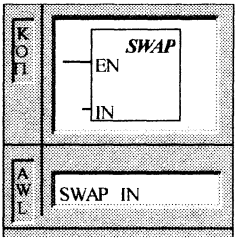


Операция *Блочная передача слов* передает заданное количество слов (N) из входного массива, начинающегося с IN, в выходной массив, начинающийся с OUT. N может лежать в диапазоне от 1 до 255.

Операнды: IN, OUT: VW, T, C, IW, QW, MW, SMW, AIW,*VD, *AC, SW

N: VB, IB, QB, MB, SMB, AC, константа, *VD, *AC, SB

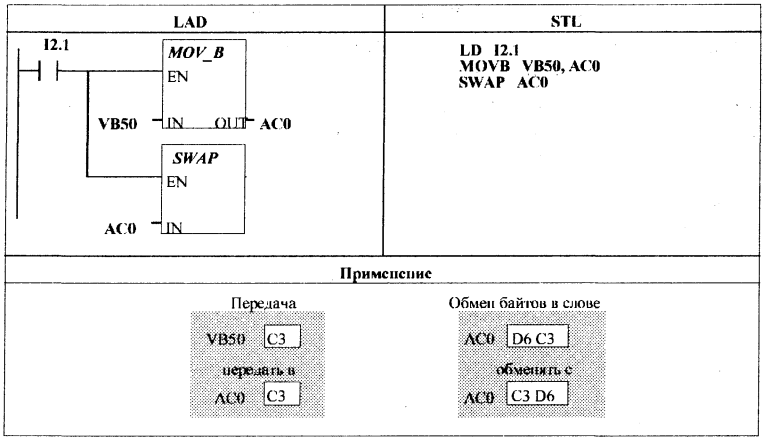
Обмен байтов в слове



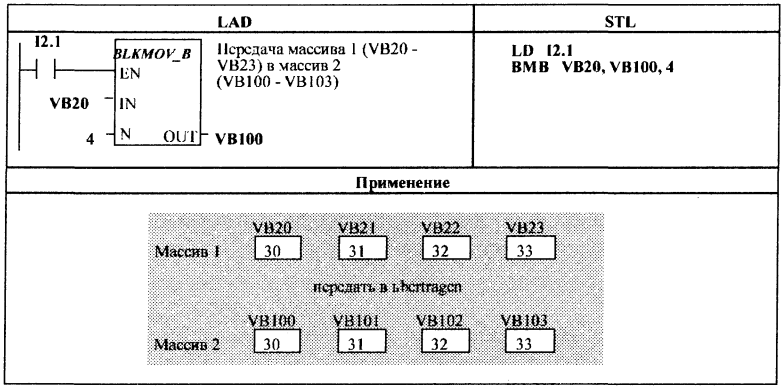
Операция *Обмен байтов в слове* меняет местами старший и младший байты в слове (IN).

Операнды: IN: VW, T, C, IW, QW, MW, SMW, SW,AC, *VD, *AC, SW

Пример операций передачи и обмена



Пример операции блочной передачи

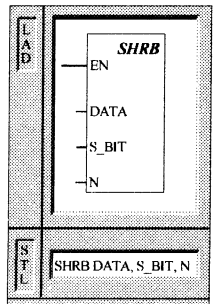


Ввод значения в регистр сдвига

Операция *Ввод значения в регистр сдвига* вводит значение DATA в регистр сдвига. S_BIT задает младший бит регистра сдвига. N показывает длину регистра сдвига и направление, в котором происходит сдвиг (положительный сдвиг = N, отрицательный сдвиг = –N).

Операнды DATA, S_BIT: I, Q, M, SM, T, C, V, S

N: VB, IB, QB, MB, SMB, AC, константа, *VD, *AC, SB



Описание операции «Ввод значения в регистр сдвига»

Операция «Ввод значения в регистр сдвига» предоставляет простой способ упорядочивания потока изделий или данных и управления ими. С помощью операции «Ввод значения в регистр сдвига» можно один раз за цикл сдвигать на один бит весь регистр сдвига. Регистр сдвига определяется младшим битом (S_BIT) и количеством битов, задаваемых длиной (N).

Адрес старшего бита в регистре сдвига (MSB.b) можно вычислить с помощью следующего уравнения:

MSB.b =[(байт, включающий S_BIT) + ([N] - 1 + (номер бита S_BIT в байте)) / 8]. [остаток от деления на 8]

Вы должны вычесть один бит, так как S_BIT принадлежит к битам регистра сдвига.

Пример: Если S_BIT = V33.4 и N = 14, то MSB.b = V35.1 или:

MSB.b = V33 + ([14] - 1 +4)/8 = V33 + 17/8 = V33 + 2 с остатком от деления 1 = V35.1

При отрицательной функции сдвига, отображаемой отрицательным значением длины (N), входные данные (DATA) вводятся в старший бит регистра сдвига. Младший бит (S_BIT) выводится из регистра сдвига.

При положительной функции сдвига, отображаемой положительным значением длины (N), входные данные (DATA) вводятся в младший бит регистра сдвига, отображаемый через (S_BIT). Старший бит выводится из регистра сдвига.

«Выдвигаемые» данные записываются в маркер переполнения (SM1.1). Регистр сдвига имеет максимальную длину 64 бита (положительную или отрицательную). На рис.П.2 показаны функции сдвига с положительным и отрицательным значениями длины N.

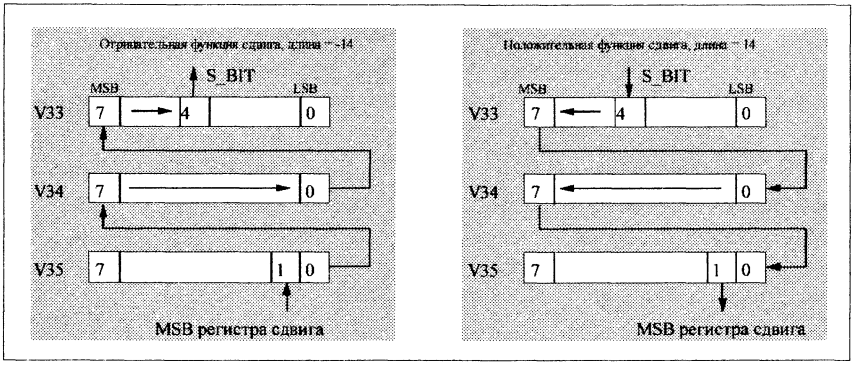
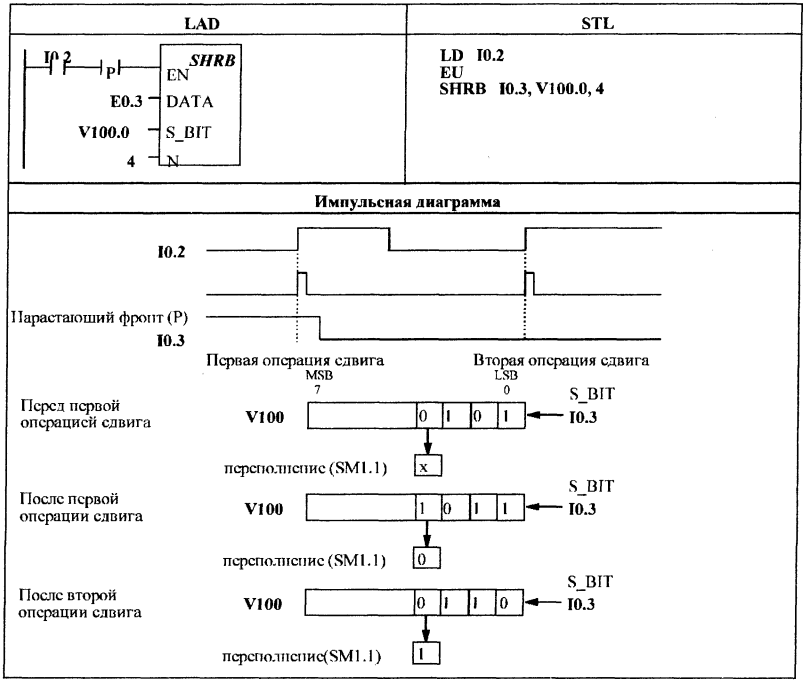


Рис.П.2

Пример операции “Ввод значения в регистр сдвига”



Сдвиг слова вправо и сдвиг слова влево [1]

Операции *Сдвиг слова вправо* и *Сдвиг слова влево* сдвигают значение слова (IN) на величину сдвига (N) вправо или влево и загружают результат в выходное слово (OUT).

Операнды IN: VW, T, C, IW, MW, SMW, AC, QW,AQW, константа, *VD, *AC, SW

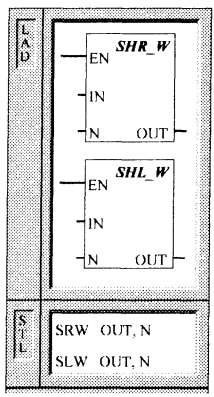
N: VB, IB, QB, MB, SMB, AC, константа, *VD, *AC, SB

OUT: VW, T, C, IW, QW, MW, SMW, AC,*VD, *AC, SW

Эти операции сдвига заполняют места “выдвигаемых” битов нулями.

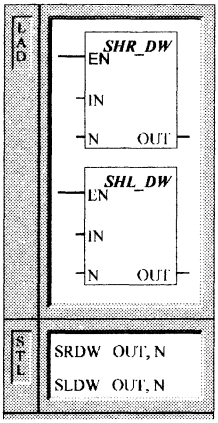
Если величина сдвига (N) больше или равна 16 (слово), то значение сдвигается максимум 16 раз. Если величина сдвига больше нуля, то маркер переполнения принимает значение «выдвигаемого» бита.

Операции «Сдвиг слова вправо» и «Сдвиг слова влево» не учитывают знака.



Эти операции влияют на специальные маркеры: SM1.0 (нуль); SM1.1 (переполнение)

Сдвиг двойного слова вправо и влево [1]



Операции *Сдвиг двойного слова вправо* и *Сдвиг двойного слова влево* сдвигают значение двойного слова (IN) на величину сдвига (N) вправо или влево и загружают результат в выходное двойное слово (OUT).

Операнды: IN: VD, ID, QD, MD, SMD, AC, HC, константа, *VD, *AC, SD

N: VB, IB, QB, MB, SMB, AC, константа, *VD, *AC, SB

OUT: VD, ID, QD, MD, SMD, AC, *VD, *AC, SD

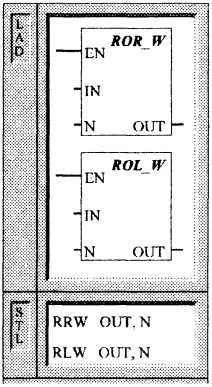
Эти операции сдвига заполняют места «выдвигаемых» битов нулями.

Если величина сдвига (N) больше или равна 32 (двойное слово), то значение сдвигается максимум 32 раза. Если величина сдвига больше нуля, то маркер переполнения принимает значение «выдвигаемого» бита.

Операции «Сдвиг двойного слова вправо» и «Сдвиг двойного слова влево» не учитывают знака.

Эти операции влияют на специальные маркеры: SM1.0 (нуль); SM1.1 (переполнение)

Циклический сдвиг слова вправо и влево [1]



Операции *Циклический сдвиг слова вправо* и *Циклический сдвиг слова влево* циклически сдвигают значение слова (IN) на величину сдвига (N) вправо или влево и загружают результат в выходное слово (OUT).

Операнды IN: VW, T, C, IW, MW, SMW, AC, QW,AIW, константа, *VD, *AC, SW

N: VB, IB, QB, MB, SMB, AC, константа, *VD, *AC, SB

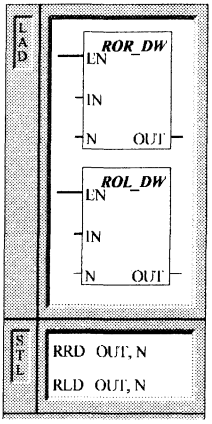
OUT: VW, T, C, IW, QW, MW, SMW, AC, *VD, *AC, SW

Если величина сдвига (N) больше или равна 16 (слово), то перед циклическим сдвигом выполняется операция по модулю 16. Отсюда получается величина сдвига в диапазоне от 0 до 15. Если величина сдвига равна нулю, то циклический сдвиг не происходит. Если циклический сдвиг происходит, то значение циклически «выдвигаемого» бита копируется в маркер переполнения.

Операции «Циклический сдвиг слова вправо» и «Циклический сдвиг слова влево» не учитывают знака.

Эти операции влияют на специальные маркеры: SM1.0 (нуль); SM1.1 (переполнение)

Циклический сдвиг двойного слова вправо и влево [1]



Операции *Циклический сдвиг двойного слова вправо* и *Циклический сдвиг двойного слова влево* циклически сдвигают значение двойного слова (IN) на величину сдвига (N) вправо или влево и загружают результат в выходное двойное слово (OUT).

Операнды IN: VD, ID, QD, MD, SMD, AC, HC, константа, *VD, *AC, SD

N: VB, IB, QB, MB, SMB, AC, константа, *VD, *AC, SB

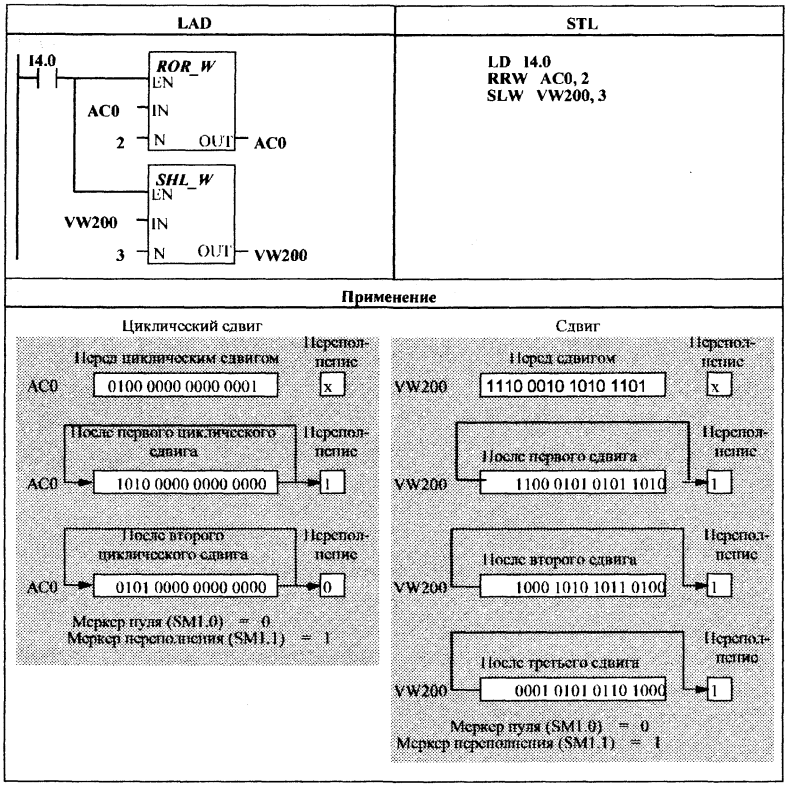
OUT: VD, ID, QD, MD, SMD, AC, *VD, *AC, SD

Если величина сдвига (N) больше или равна 32 (двойное слово), то перед циклическим сдвигом выполняется операция по модулю 32. Отсюда получается величина сдвига в диапазоне от 0 до 31. Если величина сдвига равна нулю, то циклический сдвиг не происходит. Если циклический сдвиг происходит, то значение циклически «выдвигаемого» бита копируется в маркер переполнения.

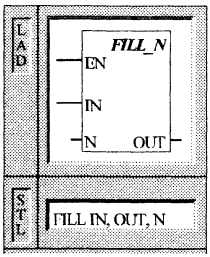
Операции «Циклический сдвиг двойного слова вправо» и «Циклический сдвиг двойного слова влево» не учитывают знака.

Эти операции влияют на специальные маркеры: SM1.0 (нуль); SM1.1 (переполнение)

Пример операций сдвига и циклического сдвига



Заполнение памяти битовой комбинацией



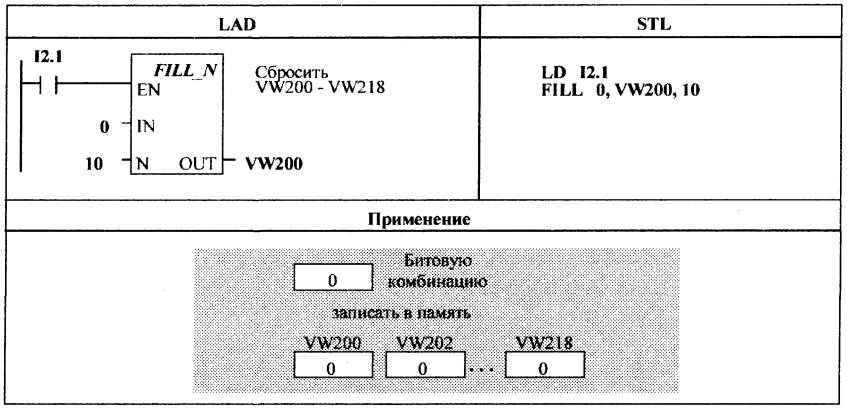
Операция *Заполнение памяти битовой комбинацией* заполняет область памяти, начинающейся с выходного слова OUT, битовой комбинацией входного слова IN для заданного количества слов N. N может лежать в диапазоне от 1 до 255.

Операнды IN: VW, T, C, IW, QW, MW, SMW, AIW, константа, *VD, *AC, SW

OUT: VW, T, C, IW, QW, MW, SMW, AQW,*VD, *AC, SW

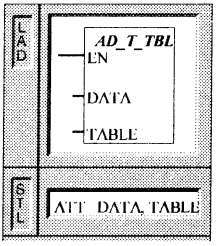
N: VB, IB, QB, MB, SMB, AC, константа, *VD, *AC, SB

Пример заполнения памяти битовой комбинацией



1.9. Табличные операции и операции поиска

Запись значения в таблицу



Операция *Запись значения в таблицу* вносит значения слов (DATA) в таблицу (TABLE)

Операнды DATA: VW, T, C, IW, QW, MW, SMW, AC, AIW, константа, *VD, *AC, SW

TABLE: VW, T, C, IW, QW, MW, SMW, *VD, *AC, SW

Первое значение в таблице задает максимальную длину таблицы (TL). Второе значение задает количество записей в таблице (EC). Новые данные добавляются в таблице после последней записи. Каждый раз, когда записываются новые данные, количество записей увеличивается на «1». Таблица может содержать максимум 100 записей, исключая параметры, задающие максимальную длину таблицы и фактическое количество записей.

Эта операция влияет на специальные маркеры: SM1.4 устанавливается в «1», если Вы пытаетесь записать в таблицу слишком много значений.

Пример операции «Запись значения в таблицу»

LAD	STL
I3.0 AD_T_TBL EN VW100 — DATA VW200 — TABLE	LD I3.0 ATT VW100, VW200

Применение

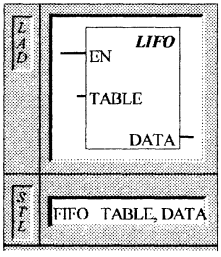
Перед выполнением операции ATT

VW100	1234	TL (макс. количество записей)
VW200	0006	EC (число записей)
VW202	0002	d0 (запись 0)
VW204	5431	d1 (запись 1)
VW206	8942	
VW208	xxxx	
VW210	xxxx	
VW212	xxxx	
VW214	xxxx	

После выполнением операции ATT

		TL (макс. количество записей)
VW200	0006	EC (число записей)
VW202	0003	d0 (запись 0)
VW204	5431	d1 (запись 1)
VW206	8942	d2 (запись 2)
VW208	1234	
VW210	xxxx	
VW212	xxxx	
VW214	xxxx	

LIFO



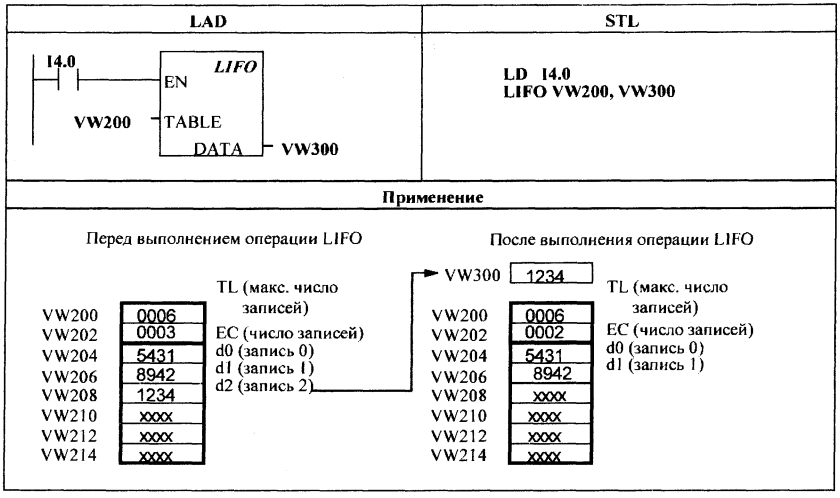
Операция *Стирание последней записи в таблице (LIFO)* стирает последнюю запись в таблице (TABLE) и выводит значение по адресу DATA. Каждый раз, когда выполняется данная операция, количество записей (EC) уменьшается на «1».

Операнды TABLE: VW, T, C, IW, QW, MW, SMW, *VD, *AC, SW

DATA: VW, T, C, IW, QW, MW, SMW, AC, AQW, *VD, *AC, SW

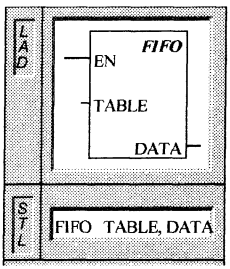
Эта операция влияет на специальные маркеры: SM1.5 устанавливается в «1», если Вы пытаетесь стереть запись в пустой таблице.

Пример операции LIFO



FIFO

Операция *Стирание первой записи в таблице (FIFO)* стирает первую запись в таблице (TABLE) и выводит значение по адресу DATA. Все остальные записи сдвигаются на одну позицию вверх. Каждый раз, когда выполняется данная операция, количество записей (EC) уменьшается на «1».

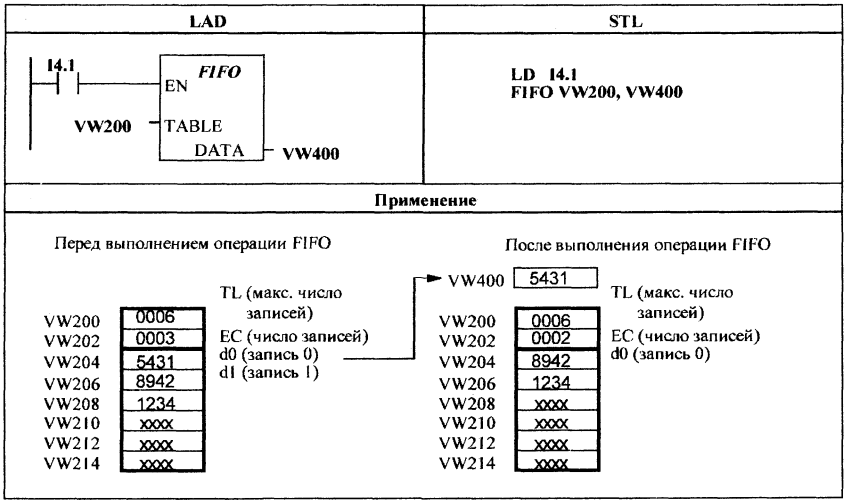


Операнды TABLE: VW, T, C, IW, QW, MW, SMW, *VD, *AC, SW

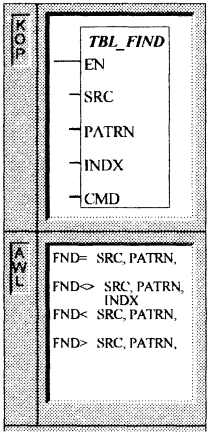
DATA: VW, T, C, IW, QW, MW, SMW, AC, AQW, *VD, *AC, SW

Эта операция влияет на специальные маркеры: SM1.5 устанавливается в «1», если Вы пытаетесь стереть запись в пустой таблице.

Пример операции FIFO



Поиск значения в таблице



Операция *Поиск значения в таблице* просматривает таблицу (SRC), начиная с записи таблицы, заданной параметром INDX, в поисках значения данных (PATRN), соответствующего заданным критериям =, ≠, < или >.

В LAD параметр CMD задает критерий числовым значением от 1 до 4, что соответствует критерию =, ≠, < или >.

Операнды SRC: VW, T, C, IW, QW, MW, SMW, *VD, *AC, SW

PATRN: VW, T, C, IW, QW, MW, SMW, AC, AIW, константа, *VD, *AC, SW

INDX: VW, T, C, IW, QW, MW, SMW, AC, *VD, *AC, SW

CMD: 1 (=) 2 (≠) 3 (<) 4 (>)

Если соответствующая запись в таблице найдется, то INDX указывает эту запись. Если в таблице нет подходящей записи, то значение INDX соответствует количеству записей в таблице. Для того, чтобы искать следующую запись, нужно сначала увеличить INDX на «1». Лишь тогда операция может быть вызвана снова.

Записи в таблице (область, где должен производиться поиск) пронумерованы от 0 до максимального значения. Таблица может содержать максимум 100 записей. Сюда не включены параметры, задающие максимальную длину таблицы и фактическое количество записей в таблице.

Если используют операции поиска в таблицах, составленных с помощью операций ATT, LIFO и FIFO, то количество записей и записи данных имеют прямое соответствие. В отличие от операций ATT, LIFO и FIFO, где максимальное количество записей задается в слове, операции поиска не используют данное слово. Поэтому операнд SRC операции поиска располагается на один адрес слова (два байта) выше, чем операнд таблицы, соответствующей операции ATT, LIFO или FIFO, как показано на рис.П.3

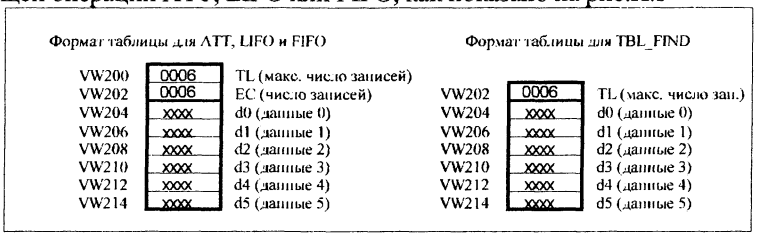
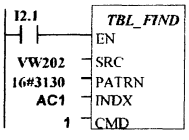


Рис.П.3

Пример операции поиска

LAD



Если активизирован I2.1 то в таблице ищется значение, равное 3130 HEX.

STL

```
ED  I2.1
FND= VW202, 16#3130, AC1
```

Применение

Это таблица, которую Вы просматриваете. Если таблица была создана с помощью операции ATT, LIFO или FIFO, то VW200 содержит максимально допустимое число записей и не требуется для операций поиска.

VW202	0006	EC (число записей)
VW204	3133	d0 (запись 0)
VW206	4142	d1 (запись 1)
VW208	3130	d2 (запись 2)
VW210	3030	d3 (запись 3)
VW212	3130	d4 (запись 4)
VW214	4541	d5 (запись 5)

AC1 0 — AC1 нужно сбросить в "0", чтобы вести поиск с самой верхней записи таблицы.

Просмотр таблицы

AC1 2 — AC1 содержит номер первой записи, соответствующей критерию поиска.

AC1 3 — Увеличить INDX на "1" перед просмотром остальных записей таблицы.

Просмотр таблицы

AC1 4 — AC1 содержит номер второй записи, соответствующей критерию поиска.

AC1 5 — Увеличить INDX на "1" перед просмотром остальных записей таблицы.

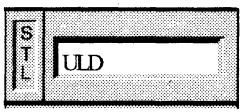
Просмотр таблицы

AC1 6 — AC1 содержит значение, соответствующее числу записей в Tabelle таблице. Вся таблица просмотрена, дальнейшие подходящие записи не найдены.

AC1 0 — Для получения возможности нового поиска в таблице нужно сбросить значение INDX в "0".

1.10. Стековые операции

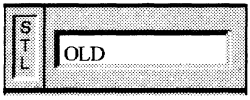
Логическое сопряжение через И первого и второго уровней стека



Операция *Логическое сопряжение через И первого и второго уровней стека* логически связывает через И первый и второй уровни стека. Результат загружается в вершину стека. После операции ULD стек содержит на один бит меньше.

Операнды: нет

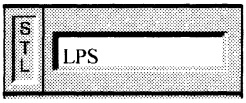
Логическое сопряжение через ИЛИ первого и второго уровней стека



Операция *Логическое сопряжение через ИЛИ первого и второго уровней стека* логически связывает через ИЛИ первый и второй уровни стека. Результат загружается в вершину стека. После операции OLD стек содержит на один бит меньше.

Операнды: нет

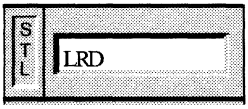
Дублирование вершины стека



Операция *Дублирование вершины стека* дублирует значение, находящееся в вершине стека, и загружает его в стек. Самое нижнее значение стека выталкивается из стека и теряется.

Операнды: нет

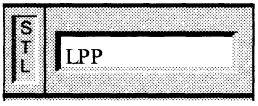
Копирование второго значения стека



Операция *Копирование второго значения стека* копирует второе значение стека в вершину стека. В стек ничего не загружается и не выталкивается, но предыдущее значение в вершине стека замещается новым значением.

Операнды: нет

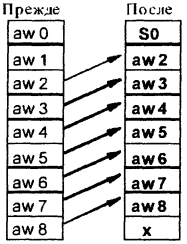
Выталкивание самого верхнего значения стека



Операция *Выталкивание самого верхнего значения стека* выталкивает самое верхнее значение из стека. Второе значение стека передвигается в вершину стека.

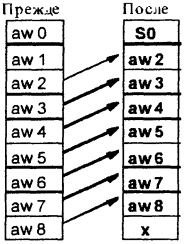
Операнды: нет

ULD
Логическое сопряжение через И первого и второго уровней стека



S0 = aw0 * aw1

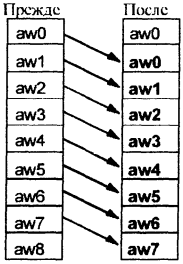
O LD
Логическое сопряжение через ИЛИ первого и второго уровней стека



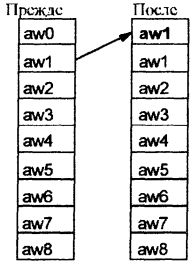
S0 = aw0 +aw1

Указание: x означает, что значение неизвестно (оно может быть равно "0" или "1").

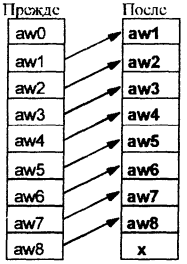
LPS
Дублирование вершины стека



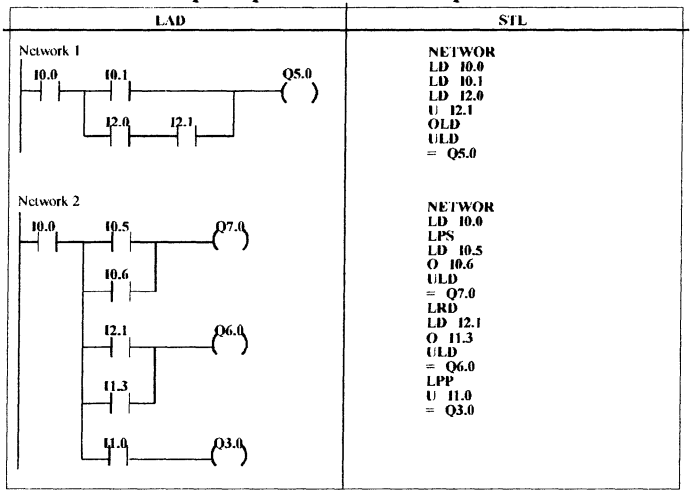
LRD
Копирование второго значения стека



LPP
Выталкивание верхнего значения стека

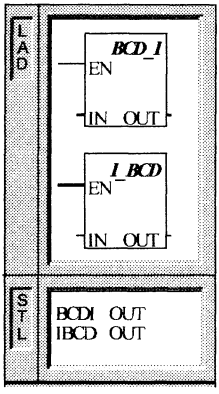


Примеры стековых операций



1.11. Операции преобразования

Преобразование BCD в целое число и преобразование целого числа в BCD [1]



Операция *Преобразование BCD в целое число* преобразует двоично-десятичное значение (IN) в целочисленное значение и загружает результат в OUT.

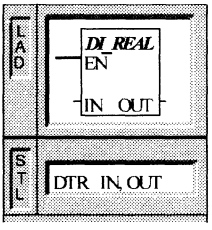
Операция *Преобразование целого числа в BCD* преобразует целочисленное значение (IN) в двоично-десятичное значение и загружает результат в OUT.

Операнды: IN: VW, T, Z, EW, AW, MW, SMW, AC, AEW, константа, *VD, *AC, SW

OUT: VW, T, Z, EW, AW, MW, SMW, AC, *VD, *AC, SW

Эти операции влияют на следующие специальные маркеры: SM1.6 (недействительное BCD–значение)

Преобразование целого числа (32 бита) в действительное число

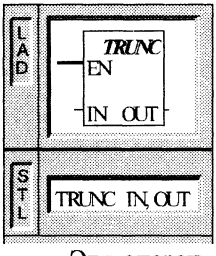


Операция *Преобразование целого числа (32 бита) в действительное число* преобразует целое число (32 бита) со знаком (IN) в действительное число (32 бита) (OUT).

Операнды: IN: VD, ED, AD, MD, SMD, AC, HC, константа, *VD, *AC, SD

OUT: VD, ED, AD, MD, SMD, AC, *VD, *AC, SD

Преобразование действительного числа в целое число (32 бита)



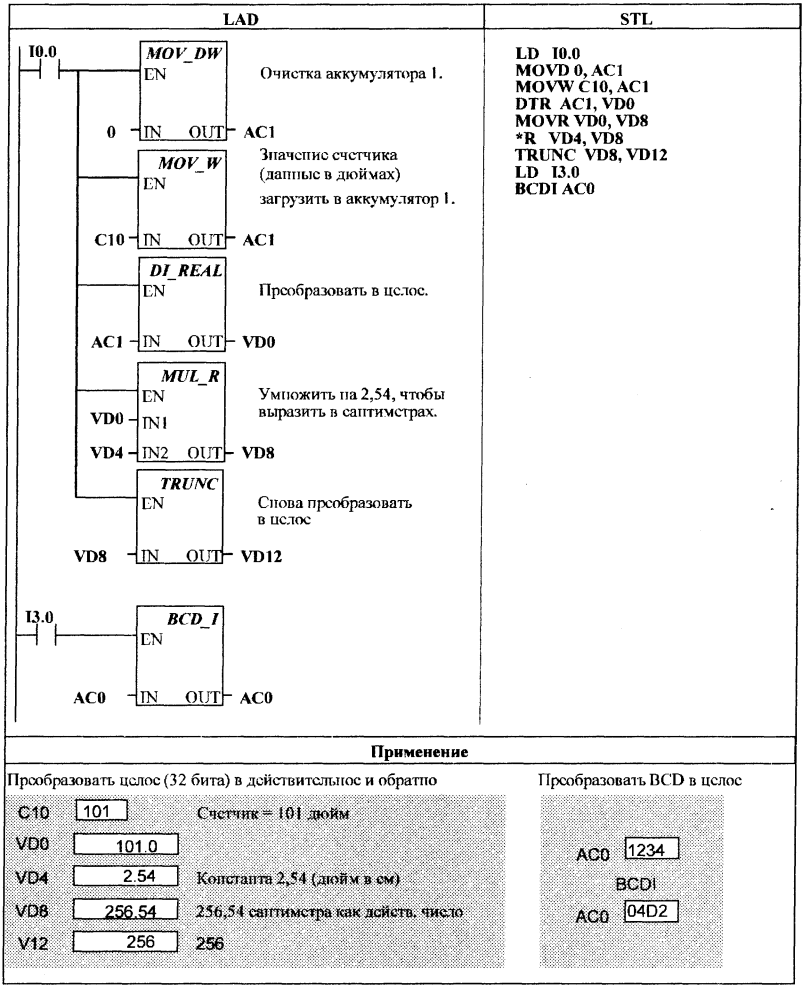
Операция *Преобразование действительного числа в целое число (32 бита)* преобразует действительное число (IN) в целое число (32 бита) (OUT). Преобразуется только целочисленная часть действительного числа (отбрасыванием знаков после десятичной точки).

Операнды: IN: VD, ED, AD, MD, SMD, AC, C, константа, VD, *AC

OUT: VD, ED, AD, MD, SMD, AC, *VD, *AC

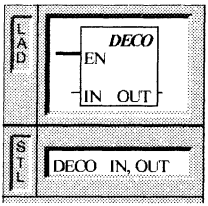
Эти операции влияют на следующие специальные маркеры: SM1.1 (переполнение)

Пример преобразования действительного числа



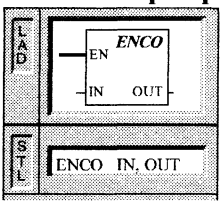
Преобразование бита в шестнадцатиричное число

Операция *Преобразование бита в шестнадцатиричное число* устанавливает в выходном слове (OUT) бит, номер которого (бит #) соответствует тому, который представлен младшим полубайтом (4 бита) входного байта (IN). Остальные биты выходного слова устанавливаются в «0».



Операнды: IN: VB, EB, AB, MB, SMB, AC, константа, *VD, *AC, SB
OUT: VW, T, Z, EW, AW, MW, SMW, AC, AAW, *VD, *AC, SW

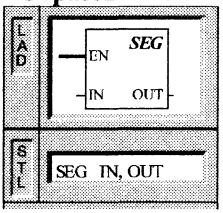
Преобразование шестнадцатиричного числа в бит



Операция *Преобразование шестнадцатиричного числа в бит* записывает номер младшего значащего бита (бит #) входного слова (IN) в младший полубайт (4 бита) выходного байта (OUT).

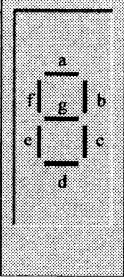
Операнды: IN: VW, T, Z, EW, AW, MW, SMW, AC, AEW, константа, *VD, *AC, SW
OUT:VB, EB, AB, MB, SMB, AC, *VD, *AC, SB

Образование битовой комбинации для семисегментного индикатора

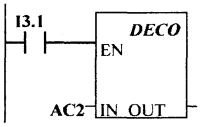


Операция *Образование битовой комбинации для семисегментного индикатора* образует битовую комбинацию (OUT), которая подсвечивает сегменты семисегментного индикатора. Подсвечиваемые сегменты представляют знак младшей цифры входного байта (IN).

Операнды: IN: VB, EB, AB, MB, SMB, AC, константа, *VD, *AC, SB
OUT: VB, EB, AB, MB, SMB, AC,*VD, *AC, SB

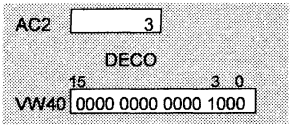
(IN) LSD	Отображ. сегментов	(OUT) -g f e d c b a			Отображ. сегментов	(OUT) -g f e d c b a
0	0	0011 1111		8	8	0111 1111
1	1	0000 0110		9	9	0110 01
2	2	0101 1011		A	A	0111 0111
3	3	0100 1111		B	b	0111 1100
4	4	0110 0110		C	C	0011 1001
5	5	0110 1101		D	d	0101 1110
6	6	0111 1101		E	E	0111 1001
7	7	0000 0111		F	F	0111 0001

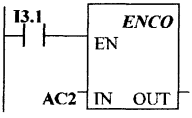
Примеры преобразования шестнадцатиричных чисел

LAD	STL
 Установка бита, соответствующего коду ошибки в аккумуляторе 2.	LD I3.1 DECO AC2, VW40

Применение

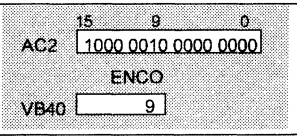
Аккумулятор 2 содержит код ошибки 3. Операция DECO устанавливает в VW40 бит, соответствующий этому коду.



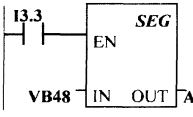
LAD	STL
 Преобразовать бит ошибки в аккумуляторе 2 в код ошибки в VB40.	LD I3.1 ENCO AC2, VB40

Применение

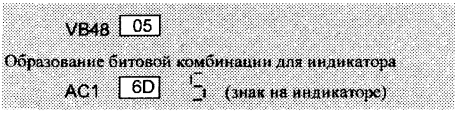
Аккумулятор 2 содержит бит ошибки. Операция ENCO преобразует младший значащий бит в код ошибки, который записывается в VB40.



Пример образования битовой комбинации для семисегментного индикатора

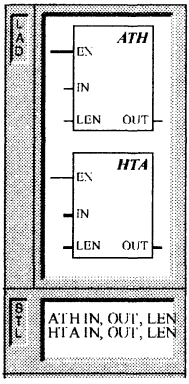
LAD	STL
	LD I3.3 SEG VB48, AC1

Применение



Преобразование строки символов ASCII–кода в шестнадцатиричное число и преобразование шестнадцатиричного числа в строку символов ASCII–кода

Операция *Преобразование строки символов ASCII–кода в шестнадцатиричное число* преобразует строку длиной LEN символов, начиная с



символа IN, в шестнадцатиричные цифры, которые начинаются с адреса OUT. Строка символов может быть длиной максимум 255 символов.

Операция *Преобразование шестнадцатиричного числа в строку символов ASCII–кода* преобразует шестнадцатиричные цифры, начиная с входного байта (IN), в строку символов ASCII–кода, которая начинается с адреса OUT. Количество шестнадцатиричных цифр, подлежащих преобразованию, задается длиной (LEN). Можно преобразовать максимум 255 шестнадцатиричных цифр.

Операнды: IN, OUT: VB, EB, AB, MB, SMB, *VD, *AC, SB

LEN:VB, EB, AB, MB, SMB, AC, константа, *VD, *AC, SB

Допустимыми ASCII–символами являются шестнадцатиричные значения от 30 до 39 и от 41 до 46.

Эти операции влияют на следующие специальные маркеры: SM1.7 (недопустимый ASCII–символ)

Примеры преобразования ASCII в HEX

LAD	STL
I3.2 — ATH: EN, VB30 — IN, 3 — LEN, OUT — VB40	LD I3.2 ATH VB30, VB40, 3

Применение

VB30 | 33 | 45 | 41 |
ATH
VB40 | 3E | AX |

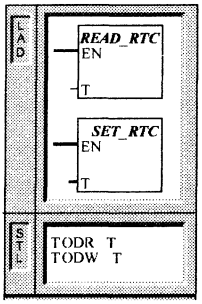
Указание: X указывает, что полубайт не был изменен.

1.12. Операции с часами реального времени

Чтение часов реального времени и запись в часы реального времени

Операция *Чтение часов реального времени* считывает текущее время суток и текущую дату из часов реального времени и загружает их в 8–байтный буфер (начальный адрес T).

Операция *Запись в часы реального времени* записывает текущее время суток и текущую дату, загруженные в 8–байтный буфер (начальный адрес T), в часы реального времени.



В STL операции Read_RTC и Set_RTC представляются посредством мнемокодов TODR (Чтение часов реального времени) и TODW (Запись в часы реального времени).

Операнды: T: VB, EB, AB, MB, SMB, *VD, *AC

После длительного прекращения подачи тока или после потери памяти часы реального времени запускаются со следующей датой и следующим временем:

Дата:	01-Jan-90
Время:	00:00:00
День недели:	воскресенье

Часы реального времени в S7–200 используют две младшие цифры для указания года. Поэтому 2000 год представляется как 00 (за 99 следует 00).

Необходимо кодировать дату и время в BCD–формате (например, 19#97 для года 1997). Используйте для этого следующие форматы данных:

год/месяц	jjmm	jj -	0 - 99	mm - 1 – 12
день/час	tthh	tt -	1 - 31	hh - 0 – 23
минут/секунд	mmss	mm -	0 - 59	ss - 0 – 59
день недели	000t	t -	0 - 7	1 = воскресенье 0 = день недели выключается.

S7–200 CPU не проверяет согласованность дня недели с датой. Вследствие этого могут приниматься недопустимые даты, такие как 30 февраля. Поэтому следует всегда убедиться в том, что дата введена правильно. Никогда не используйте операции TODR и TODW в главной программе и в программе обработки прерываний одновременно. В противном случае операция TOD больше не сможет выполняться в программе обработки прерываний при возникновении прерывания, если она уже была выполнена в главной программе. SM4.5 установится, указывая, что две операции попытались одновременно получить доступ к часам.

[1] При программировании в LAD можно указать, что IN1 совпадает с OUT. Таким образом экономится место в памяти.

СОДЕРЖАНИЕ