Ганс БЕРГЕР

ПРОГРАММИРОВАНИЕ УПРАВЛЯЮЩИХ УСТРОЙСТВ НА ЯЗЫКЕ STEP5

Том 1 Программирование основных функций

АКЦИОНЕРНОЕ ОБЩЕСТВО С И М Е Н С

Предисловие

Программируемые устройства управления заняли в последние годы в технике управления прочное место наряду с контакторными системами, системами на полупроводниках и УВМ для технологических процессов. Одним из таких примеров является система управления СИМАТИК S5.

С появлением мощных электронных интегральных схем эта система была подвергнута переработке в результате чего появилась система автоматизации СИМАТИКS5. В ней воплотился опыт, приобретенный при эксплуатации системы СИМАТИК S5 и проведены усовершенствования, ставшие возможными благодаря применению микропроцессоров появление которых не в последнюю очередь отразилось на языке программирования.

Язык программирования STEP5 является дальнейшим развитием языка STEP5. который зарекомендовал себя одним из самых мощных языков для программирования электронных устройств управления. Мнемотехнические сокращения этого языка программирования представление в виде таблицы команд отображают стремление западногерманских производителей электронных программируемых управляющих устройств к унификации, установленной нормами ДИН 19239.

Дополнительно язык программирования STEP 5 дает пользователю возможность представления программ в виде функциональной логической схемы или схемы релейно-контактной. Полученное таким образом широкое согласование метода программирования с представлением решаемой задачи управления упрощает обращение с этим языком программирования. В сочетании с простотой применения это означает широкому кругу пользователей подготовку, ввод и изменение программ на языке STEP 5.

Предлагаемая производителем управляющих устройств "библиотека" так называемых блоков функций является дальнейшим шагом в рационализации подготовки программ и тем самым снижения расходов на математическое обеспечение. Пользователи получают в свое распоряжение полностью

запрограммированными и отлаженными как сложные единичные функции, так и комплексные задачи, встречающиеся, например, в аналоговых функциях регулирования в связи с необходимостью управления и наблюдения за регулируемыми параметрами. Благодаря самостоятельно программируемым блокам функций, обладающим по сравнению с основными функциями значительно расширенным запасом операций, опытный пользователь получает в свои руки вспомогательное средство, позволяющее программировать на языке STEP 5 также сложные, индивидуальные задачи управления. Язык программирования STEP 5 перекрывает тем самым все задачи мощных программируемых устройств управления, охватывающие благодаря применению в качестве управляющего элемента микропроцессора функции, аналогичные выполняемым управляющими вычислительными машинами.

Пособие "Программирование управляющих устройств на языке STEP 5" разделено на 3 тома. В предлагаемом первом томе описываются основные функции языка программирования и представление программ в виде функциональной схемы, схемы контактов и таблицы команд. Второй том содержит обзор и описание предлагаемых изготовителем стандартных блоков функций, а также обращение с этими вспомогательными средствами программирования. Внутренняя структура и самостоятельная подготовка функциональных блоков с функциями, ориентированными на индивидуальные задачи управления, описываются в третьем томе.

ПРЕДИСЛОВИЕ К 5-му ИЗДАНИЮ

Широкое распространение системы автоматизации СИМАТИК S5 за очень небольшой промежуток времени сделало необходимым выпустить 3-е! издание этой книги. Сделаны дополнения, касающиеся запаса операций появившихся на сегодня устройств автоматизации. Кроме того, книга дополнена новым разделом, описывающим вычислительные функции, которые стали составной частью основных функций.

Эрланген, февраль 1982 г.

АКЦИОНЕРНОЕ ОБЩЕСТВО С И М Е Н С

ОГЛАВЛЕНИЕ

Введение

В традиционной технике управления задачи решаются путем монтажа индивидуальных релейно-контакторных схем, создаваемых для конкретных решаемых задач. Поэтому контакторные и релейные системы управления, а также системы на полупроводниках, собираемые из отдельных блоков, называют *жестко программируемыми управляющими устройствами*. Их программа заложена в монтажных соединениях.

В свободно программируемых устройствах управления, напротив, применяются стандартные серийные приборы. Необходимые функции управления осуществляются по программам, вводимым в соответствующее запоминающее устройство.

Система автоматизации СИМАТИК S5 является свободно программируемой системой. Она состоит из устройств автоматизации /контроллеров/, устройств программирования и языка программирования. Подлежащая решению задача управления содержится в накопителе программ устройства автоматизации в виде последовательности команд. Управляющее устройство последовательно считывает команды, интерпретирует их содержание и заботится об их исполнении.

В связи с такой *циклической обработкой программы* выход устройства автоматизации не реагирует с обычной для электронных систем управления скоростью на изменение сигнала на соответствующем входе. Изменение состояния сигнала на каком-либо входе обрабатывается управляющим устройством только тогда, когда будет обработана команда языка STEP 5! необходимая для запроса. Точно также воздействие на состояние сигнала на каком-либо выходе осуществляется только после обработки нужной команды.

Как правило, одна определенная команда обрабатывается во время прохождения программы только один раз. Время между двумя обработками одной команды называется *временем цикла*. Этот отрезок времени имеет большое значение для определения времени реакции свободно программируемого устройства, управления.

Время реакции устройства управления как минимум равно задержке сигнала на входах и выходах, а в самом сложном случае - сумме задержки сигнала и времени цикла. В устройствах автоматизации с отображением процесса время реакции в обоих случаях увеличивается на время цикла.

ЯЗЫК ПРОГРАММИРОВАНИЯ STEP 5

С помощью языка программирования STEP 5 формулируются различные задачи автоматизации, решаемые средствами автоматизации СИМАТИК S5. Три ориентированных на пользователя способа представления облегчают описание подлежащих решению задач и значительно облегчают изучение и применение этого языка программирования. Представление программы в виде функциональной /логической/ схемы (FUP, находится в стадии подготовки) символами на основе ДИН 40 700, а также представление в виде контактной схемы (КОР) наглядно описывают функцию управления, основываясь на имеющихся функциональных или принципиальных электрических схемах. Представление в виде таблицы команд (AWL) по ДИН 19239 /проект/ наиболее близко подходит к внутреннему отображению программы управления, машинному языку. Поэтому ему отдается предпочтение в простых устройствах программирования. Кроме того, это представление наилучшим образом подходит для ввода программ с помощью перфокарт /в очень больших системах управления/.

В разделе 1.1 приведены пояснения понятий, связанных с языком STEP 5. Далее следует обзор основных функций, программируемых с помощью языка STEP 5 /раздел 1.2/. Обзор, в соответствии с различными способами представления языка программирования, дан в виде четырех таблиц. Обзор дополнен кратким описанием каждой функции.

Устройства автоматизации имеют различный запас операций в соответствии с их мощностью. В разделе 1.3 дан обзор операций, используемых в конкретных устройствах автоматизации. Программаторы с дисплеем позволяют иметь удобное представление в виде функциональных и контактных схем. Однако, вместе с тем экран ограничивает представление графических изображений /раздел 1.4/.

Понятия

Программа

Программы, составляемые с помощью языка STEP 5, подразделяются на системные и прикладные.

Системные программы представляют собой совокупность всех команд и условий для реализации внутренних рабочих функций /например, защита данных при отключении питания, организационные функции при наложении блоков и т.д./. Системные программы хранятся в перепрограммируемых ПЗУ (ППЗУ=EPROM), которые не зависят от внешнего источника питания и располагаются в накопителе программ на центральном блоке. Пользователь доступа к системным программам не имеет.

Прикладные программы представляют собой совокупность всех команд и условий для обработки сигналов, через которые оказывается воздействие на управляемую установку /процесс/ в соответствии с задачей управления. Прикладные программы подразделяются на блоки.

Блок представляет собой часть программы, ограниченную функцией, структурой или целевым назначением. В языке программирования STEP 5 различают блоки, содержащие команды для обработки сигналов /организационные, программные и функциональные блоки/, и блоки, в которых хранятся данные /блоки данных/.

Организационные блоки(OB) служат для управления прикладной программой в форме перечня подлежащих обработке блоков программы. Имеются организационные блоки для циклической обработки программ, для обработки с управлением по прерываниям и для обработки с управлением по времени.

В программных блоках (PB) прикладная программа расчленяется на технологические группы, напр., на отдельные управляющие звенья.

С помощью функциональных блоков (PB) реализуются часто повторяющиеся или очень сложные функции. Функциональные блоки поставляются готовыми /стандартные функциональные блоки/ или программируются самим пользователем. Внутри функциональных блоков наряду с основными операциями имеются

дополнительные /"дополняющие"/ операции. Функциональные блоки параметризуются, т.е. функция, реализуемая каким-либо блоком, может протекать с различными операндами /параметрами блока/.

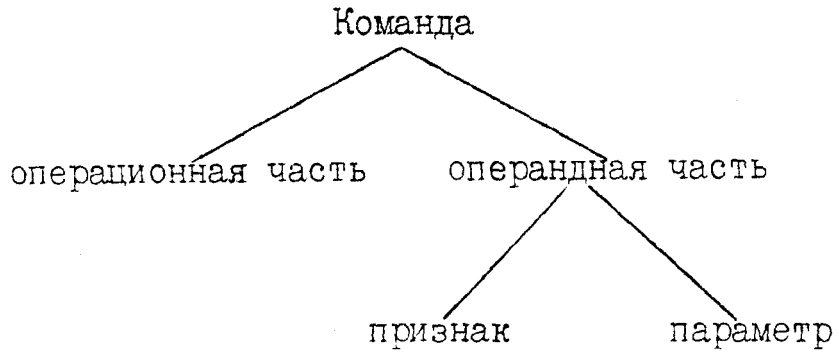
В блоках данных (DB) находятся данные, на которых работает программа пользователя.

Если нужно обработать какой-либо блок, его следует вызвать. Вызов может быть абсолютным или зависящим от результата логической операции. После окончания обработки блока программа продолжается с того места, где находился вызов блока, т.е. в "старшем" блоке.

Команда

Команда является самостоятельной наименьшей единицей программы. Она представляет собой рабочее указание для процессора. Команда состоит из операционной части и операндной части, а операндная часть - в свою очередь из признака и параметра.

Структуру команды можно представить следующим образом:



операционная часть описывает выполняемую функцию. Она говорит, что должен делать процессор.

Операндная часть содержит данные, необходимые для выполнения операции. Она говорит, с чем должен работать процессор. Язык программирования STEP 5 различает следующие области операндов:

Входы E	они представляют собой мест сопряжения процесса с устройством автоматизации /отображение процесса/.
Выходы A	они представляют собой место сопряжения устройства автоматизации с процессом /отображение процесса/.
Метки M	они предусматриваются для запоминания промежуточных результатов в двоичном виде.
Данные в время T	они предусматриваются для запоминания промежуточных результатов в цифровом виде, с их помощью реализуются временные функции.
Счетчики Z	с их помощью реализуются функции счета.
Периферия P,Q	для прямого обращениями периферийным устройствам /блокам ввода/вывода/
Константы K	Представляют собой постоянное заданное число.
Блоки OB,PB,PB,DB	служат для образования структуры программы.

Обозначение области операнда называется признаком операнда. Для обращения к определенному операнду в его области необходимо указать параметр.

Параметр указывает на адрес операнда. Адресование таких областей операнда как входы E, выходы A и метки M производится по байтам, т.е. указанное число относится к определенному байту этой области операнда /адрес байта/.

Обращение к этим областям операнда может производиться и по битам. Адрес бита отделяется от адреса байта
точкой.

Пример:

```
E      I0.3
A      7.0
M      I20  7
```

адрес бита
адрес байта
признак операнда
параметр
операнд

Адресование области операнда "периферия P" производится также по байтам, однако адреса бита не имеет.

Пример:

```
PB     I50
```

адрес байта = параметр
признак операнда
операнд

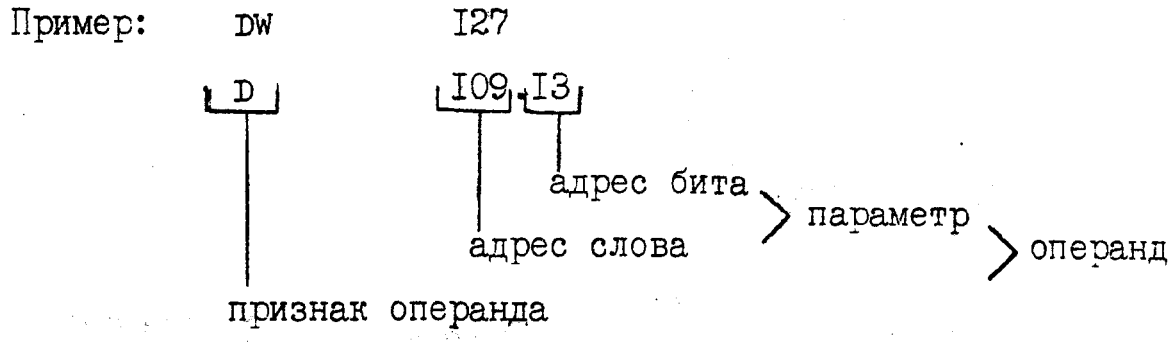
Адресование областей операнда T - время и z - счетчики производится по словам /адрес слова/. Эти области операнда адреса бита не имеют.

Пример:

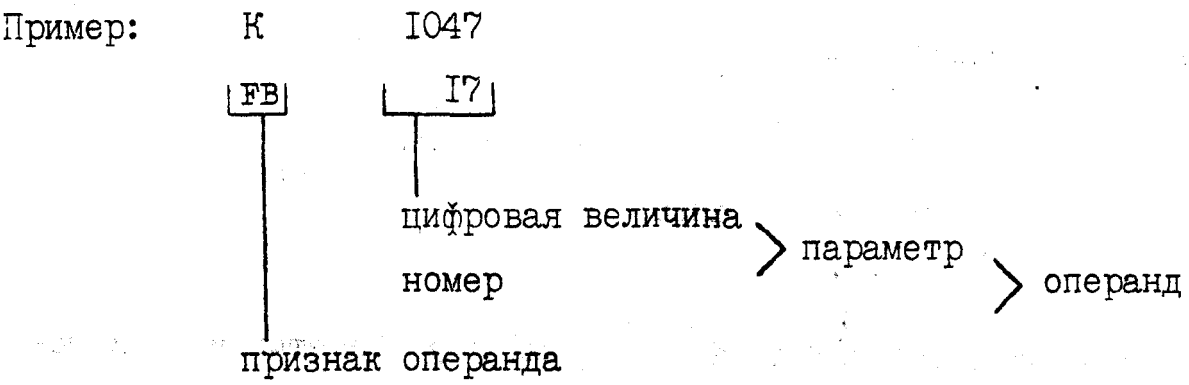
```
T      I5
Z      7
```

адрес слова = параметр
признак операнда
операнд

Область операнда D /данные/ может адресоваться, как по словам, так и по битам.



Параметр области операнда К /константа/ представляет собой число, которое должно обрабатываться как цифровая величина. Параметры областей операнда. OB,PB, FB и DB указывают номер операнда в этой области.



1.2 Обзор функций, краткое описание

Обзор функций разделен в соответствии с видами представления по разным таблицам.

Объем всех логических функций разделен на две таблицы, так как формы представления функциональной схемы и контактной схемы в этой области различаются /таблица I, таблица 2/. Все остальные символы функций в обоих видах представлений одинаковы. Они приведены в таблице 3. Перечень операций в виде таблиц команд приведен в таблице 4.

Таблица 1 содержит краткое описание логических операций при их изображении в виде функциональной схемы.

Таблица 2

содержит краткое описание логических операций при их изображении в виде контактной схемы /лестничной диаграммы - прим.перев./.

Таблица 3

содержит краткое описание символов функций, общих для функциональной и контактной схем.

Таблица 4

содержит краткое описание операций при их представлении в виде списка /таблицы/ команд.

Таблица 1 Символы функциональных схем /логические операции /

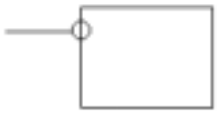
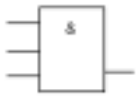
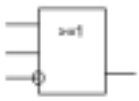
Изображение	Краткое описание
	Вход символа функции Результат опроса /состояние сигнала, которое воздействует непосредственно на символ функции/ равен «1», если состояние сигнала операнда, относящегося к данному входу, - «1». Если состояние сигнала операнда на входе
	Инвертированный вход символа функции Результат опроса /состояние сигнала, которое воздействует непосредственно на символ функции/ равен «1», если состояние сигнала операнда, относящегося к данному входу - «0». Если состояние сигнала операнда на входе равно «1», то результат опроса равен «0».
	Выход символа функции Операнд, находящийся на выходе символа функции, имеет состояние сигнала «1», если результат логической операции функции /= состоянию сигнала на выходе/ равен «1».
	функция "И" Состояние сигнала на выходе функции "И" равно "1",если в результате опроса на всех входах функции "И" получена «1». Если результат опроса на входе равен "0", то состояние сигнала на выходе - «0». Количество входов
	ФУНКЦИЯ "ИЛИ" Состояние сигнала на выходе функции "ИЛИ" равно «1», если результат опроса входа функции "ИЛИ" равен «1». При результате опроса всех входов равном "0", состояние сигнала на выходе - «0». Количество входов не ограничено.

Таблица 2

Символы контактных схем /логические соотношения/

Изображение	Краткое описание
`----] [----`	Нормально открытый контакт Контакт "замкнут", если состояние сигнала относящегося к нему операнда равно «1». При состоянии сигнала соответствующего операнда равном «0», контакт "разомкнут".
`----]/[----`	Нормально закрытый контакт Контакт "замкнут", если состояние сигнала относящегося к нему операнда равно "0". При состоянии сигнала соответствующего операнда равном «1», контакт "разомкнут".
`----( )----`	Обмотка реле или контактора /в конце цепи/ Состояние сигнала операнда, относящегося к к обмотке, равно «1», если в цепи есть ток. Операнд 'имеет состояние сигнала "0", если ток по цепи не идет.
`--] [--+--] [--+-- --+--( )--[`	Последовательная схема Ток течет по цепи, если все контакты замкнуты. Ток не течет, если один из контактов разомкнут. Количество контактов ограничено шириной экрана дисплея. Параллельные схемы также можно
`--] [--+-- --+-- --+--( )--]`	Параллельная схема Ток течет по цепи, если один из контактов замкнут. Ток не течет, если все контакты разомкнуты. Количество параллельных включений не должно превышать две с половиной высоты экрана дисплея. Допускается также параллельное включение последовательных соединений.

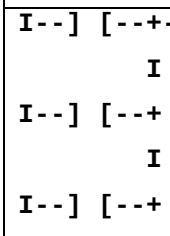
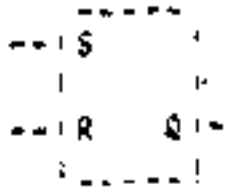
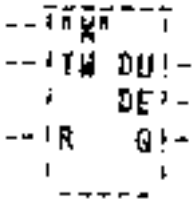
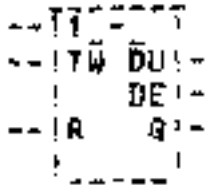
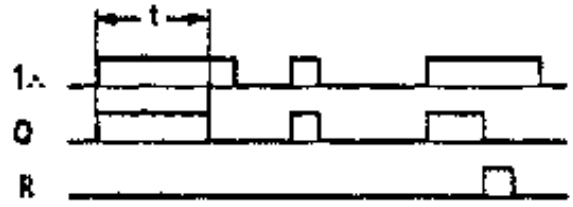
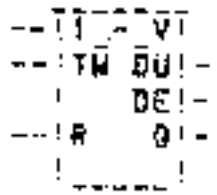
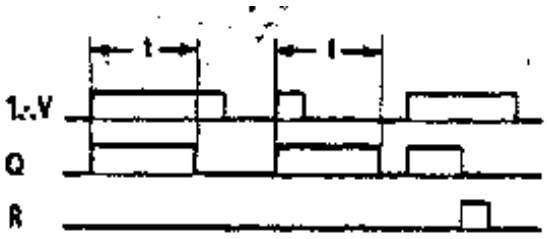
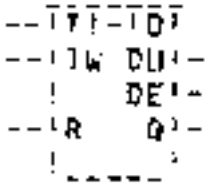
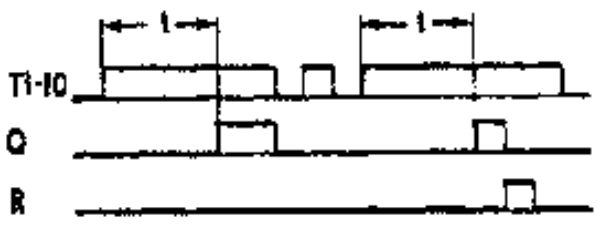
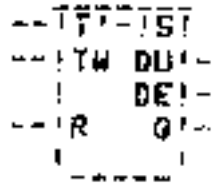
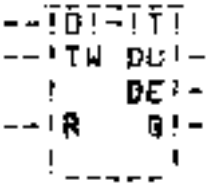
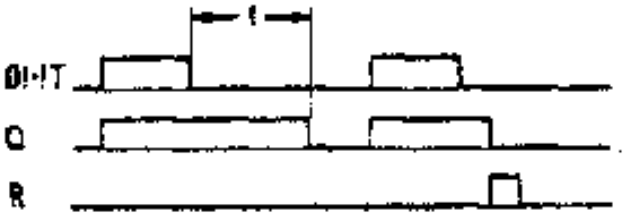
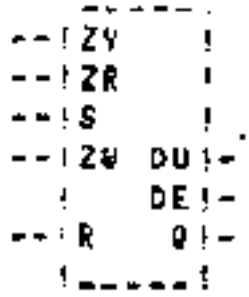
Изображение	Краткое описание
	Параллельная схема Ток течет по цепи, если один из контактов замкнут. Ток не течет, если все контакты разомкнуты. Количество параллельных включений не должно превышать две с половиной высоты экрана дисплея. Допускается также параллельное включение последовательных соединений.

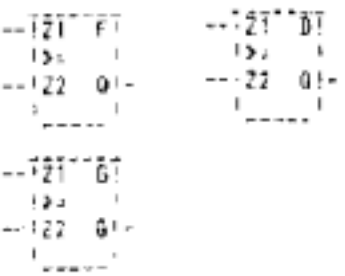
Таблица 3

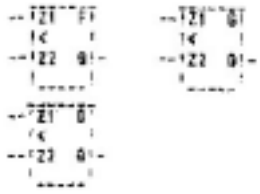
Символы, общие для функциональных и контактных схем

Изображение	Краткое описание
	Функция запоминания с приоритетом сброса При появлении на входе S в результате логических операций "I" происходит установка памяти. На выходе 0, состояние сигнала также будет "I". При появлении "I" на входе R происходит сброс памяти. Выход Q будет иметь тогда состояние сигнала "O". Логический "O" на входах изменения состояния сигнала на выходе не вызывает. Сброс имеет приоритет, если состояние сигнала на обоих входах равно "I". На выходе Q состояние сигнала тогда будет равно "O".
	Функция запоминания с приоритетом установки аналогична вышеописанной. Если на обоих входах состояние сигнала "I", приоритет принадлежит установке. Тогда выход Q будет иметь состояние сигнала "I".
	Функция времени. Общая Вход обозначенный символом "X", является входом запуска времени. Здесь представляется соответствующая характеристика времени /см. ниже/.

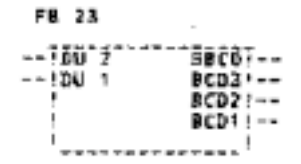
Изображение	Краткое описание
	На входе т-W находится значение времени /3 декады в двоично-десятичном коде/. Вход R является входом сброса. Состояние сигнала "1" на этом входе вызывает состояние "0" на выходе Q. Выход Q имеет состояние сигнала "1" в соответствии с диаграммой времени /см.ниже/. Выходы DU и DE являются цифровыми выходами. На выходе DU значение времени дано в двоичном коде, на выходе DE - в двоично-десятичном с определенным растром /ценой делений/.
	Функция времени, короткий импульс t установленное время. 
	Функция времени, удлиненный импульс t установленное время 
	Функция времени, задержка включения t установленное время 

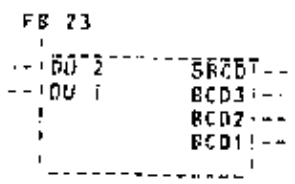
Изображение	Краткое описание
	Функция времени, задержка включения запоминанием t установленное время 
	Функция времени, задержка отключения t установленное время 
	Функция_счета Входы ZV и ZR являются входами прямого и обратного счета. На этих входах учитывается только смена состояния сигнала с "О" на "I". На входе s, 'который служит для установки счетчика на числовое значение ZW (5 декады в двоично-десятичном коде),также учитывается только смена состояния сигнала с "О" на "I". Вход R для сброса действует статически. На выходе Q будет "О", если счетчик стоит на нуле, и "I", если числовое значение больше нуля. Выходы du и DE являются цифровыми. На выход DU цифровая величина поступает в двоичном коде, на выход DE - в двоично-десятичном.

Изображение	Краткое описание
	Функция сравнения на «равно» На выходе Q состояние сигнала равно "1", если комбинации битов операндов на входах Z1 и Z2 одинаковы.
	Функция сравнения на «не равно» На выходе Q состояние сигнала равно "1", если комбинации битов операндов на входах Z1 и Z2 не одинаковы.
	Функция сравнения на «больше» На выходе Q состояние сигнала равно "I", если значение операнда на входе Z1 больше, чем значение операнда на входе z 2. Величина операндов интерпретируется в соответствии с представлением числа.
	Функция сравнения на «больше или равно» На выходе Q состояние сигнала равно "1", если значение операнда на входе Z1 больше или равно значению операнда на входе Z2. Величина операндов интерпретируется в соответствии с представлением числа.

Изображение	Краткое описание
	Функция сравнения на «меньше» На выходе Q состояние сигнала разно "I", если значение операнда на входе Z1 меньше значения операнда на входе Z2. Величина операндов интерпретируется в соответствии с представлением числа.
	Функция сравнения на «меньше или равно» На выходе Q состояние сигнала равно "1", если значение операнда на входе Z1 меньше или равно значению операнда на входе Z2. Величина операндов интерпретируется в соответствии с представлением числа.
+ F + G	Сложение Содержания обоих аккумуляторов суммируются в соответствии с представлением чисел F или G. Результат хранится в аккумуляторе до последующей обработки.
- F - G	Вычитание Содержание основного аккумулятора вычитается из вспомогательного аккумулятора в соответствии с представлением чисел F• или G . Результат поступает в аккумулятор для последующей обработки.

Изображение	Краткое описание
XF XG	Умножение Содержания обоих аккумуляторов перемножаются в соответствии с представлением чисел F или G. Результат хранится в аккумуляторе до последующей обработки.
:F :G	Деление Содержание вспомогательного аккумулятора делится на содержание основного /в соответствии с представлением чисел F или G /. Результат поступает в аккумулятор для последующей обработки.
L-	Функция загрузки Значение операнда, указанного при функции загрузки, загружается в аккумулятор.
T-	Функция переноса Содержимое аккумулятора переносится в операнд, указанный при функции переноса
SPA PBn	Абсолютный вызов одного из блоков программы Линейная обработка программы прекращается. Она будет продолжена в указанном блоке программы PBn.

Изображение	Краткое описание
SPB PBn	Условный вызов блока программы Если результат логической операции предыдущего символа Функции равен "1", или если предыдущая цепь была токопроводящей, линейная обработка программы прекращается. Она будет продолжена в программном блоке PBn. Если результат логической операции символа функции равен "0" или если предыдущая цепь не была токопроводящей, вызов не выполняется и обработка программы продолжается в обычном порядке.
	Абсолютный вызов функционального блока Функциональный блок обрабатывается постоянно. Операнды на входах и на выходах /параметры блока/ обрабатываются в соответствии с внутренней функцией блока.

Изображение	Краткое описание
	Условный вызов функционального блока Если результат логического сопряжения предыдущего символа функции равнялся "1" или если предыдущая цепь была токопроводяшей, происходит обработка функционального блока. Операнды на входах и на, выходах /параметры блока/ обрабатываются в соответствии с внутренней функцией блока. Если результат равен "0" или если цепь тока не проводила, вызов не выполняется и продолжается обработка программы.
A DB n	Вызов блока данных С вызовом блока данных происходит выбор области данных. Все последующие операции с областью операнда данных в будут относиться к блоку данных DB N.

Изображение	Краткое описание
BE	Конец блока После вызова только что обработанного блока продолжается обработка программы в "старшем" блоке.
BEB	Условное окончание блока Если результат логического сопряжения предыдущего символа функции равен "1" или если предыдущая цепь была токопроводящей, после вызова только что обработанного блока продолжается обработка программы в "старшем" блоке. Если результат равен "0" или если цепь тока не проводила, операция не выполняется и продолжается обработка программы.
STP	Стоп Обработка программы прерывается. Процессор останавливается. Новый или повторный пуск возможен только вручную.

Таблица 4

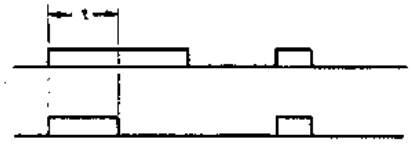
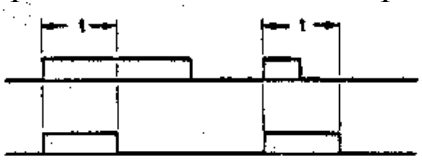
Обзор операций при представлении программы в виде таблицы команд

Изображение	Краткое описание
U -	Логическая операция «И», опрос на состояние сигнала «1». Результат опроса равен "1", если находящийся при этой операции операнд имеет состояние сигнала "1". Если операнд имеет "0", то и результат опроса будет равен "0". Результат опроса по функции "И" сопрягается с результатом "логического сопряжения, находящимся в процессоре. При первичном опросе процессор принимает результат опроса в качестве результата логического сопряжения.
UN -	Логическая операция «И», опрос на состояние сигнала «0». Результат опроса равен "1", если находящийся при этой операции операнд имеет состояние сигнала. "0". Если состояние сигнала операнда равно "1", то результат опроса "0". Результат опроса по Функции "И" сопрягается с результатом логического сопряжения, находящимся в процессоре. При первичном опросе процессор принимает результат опроса в качестве результата логического сопряжения.

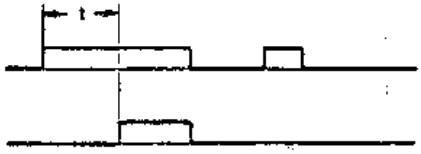
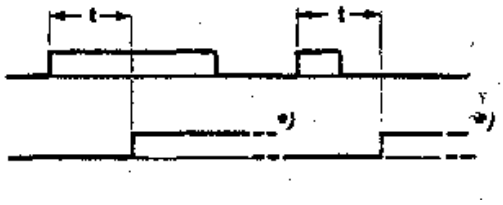
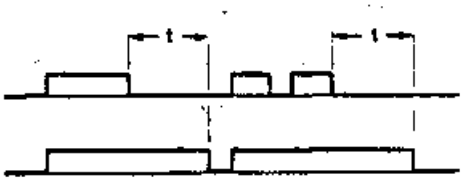
Изображение	Краткое описание
O -	Логическая операция «ИЛИ», опрос на состояние сигнала «1». Результат опроса равен "1", если находящийся при этой операции операнд имеет состояние сигнала "1". Если состояние сигнала операнда равно "0", то и результат опроса равен "0". Результат опроса по функции "ИЛИ" сопрягается с результатом логического сопряжения, находящимся в процессоре. При первичном опросе процессор принимает результат опроса в качестве результата логического сопряжения.
ON -	Логическая операция «ИЛИ», опрос на состояние сигнала «0». Результат опроса равен "1", если находящийся при этой операции операнд имеет состояние сигнала "0". Если операнд имеет состояние сигнала "1", результат опроса равен "0". Результат опроса по функции ИЛИ сопрягается с результатом логического сопряжения, находящимся в процессоре. При первичном опросе процессор принимает результат опроса в качестве результата логического сопряжения.

Изображение	Краткое описание
O	Логическое сопряжение «ИЛИ» функции «И» Результат сопряжения последующей операции И сопрягается по функции ИЛИ с предыдущим результатом логических операций.
U(	Логическое сопряжение «И» с выражением в скобках Результат логических операций выражения в скобках сопрягается с предыдущим логическим результатом по функции И.
O(	Логическая операция «ИЛИ» с выражением в скобках Результат логический операций выражения в скобках сопрягается с предыдущим логическим результатом по функции ИЛИ.
)	Скобку закрыть Этой операцией закрывается выражение в скобках

Изображение	Краткое описание
= -	Присвоение Операнд, находящийся при этой операции, имеет состояние сигнала "I", если результат логического сопряжения равен "I". При логическом "О" состояние сигнала операнда "О".
S -	Установка /функция запоминания/ Операнд, находящийся при этой операции, имеет состояние сигнала "I", если результат логического сопряжения равен "I". Логический "О" влияния на эту операцию не оказывает.
S Zn	Установка /счетчика / Счетчик Zn, находящийся при этой операции, при смене логического результата с "О" на "I" устанавливается на значение, указанное в следующей команде. Логический "О" и статическая "I" не влияют на эту операцию.
R -	Сброс /функция запоминания/ Операнд, находящийся при этой операции, имеет состояние сигнала "0", если логический результат в переключателе "I". Логический "О" влияния на эту операцию не оказывает.

Изображение	Краткое описание
R -	Операнд, находящийся при этой операции. сбрасывается, если логический результат "I". Опросы этого операнда дают состояние сигнала "О". Логический "О" влияния на эту операцию не оказывает.
SI T n /Указание длительности/ VKE SI Состояние сигнала времени Tn	Запуск_времени_как_короткого импульса Запуск времени происходит с указанной ниже продолжительностью периода. 
SV T n / Указание длительности/ VKE при SV Состояние сигнала времени Tn	Запуск_времени_как_удлиненного_импульса Запуск времени происходит с указанной ниже продолжительностью периода. 

VKE результат логической операции

Изображение	Краткое описание
ST T n VKE При SE Tп Состояние сигнала времени Tn	Зацуск_времени_как_задержки_включенкя Запуск времени происходит с указанной ниже продолжительностью периода. 
SS T n VKE При SS Tп Состояние сигнала времени Tn	Запуск_времени_как_задезжки_включения запоминанием Запуск времени происходит с указанной ниже продолжительностью периода. 
SA T n VKE При SA Tп Состояние сигнала времени Tn	Запуск_времени_как_задержки отключения Запуск времени происходит с указанной ниже продолжительностью периода. 

Изображение	Краткое описание
ZV Zn	Счетчик прямого счета Если логический результат при этой операции меняется с "О" на "I", то показания счетчика Zn увеличиваются на 1. Логический "О" и статическая "I" влияния не имеют.
ZR Zn	Счетчик обратного счета Смена логического результата при этой операции с "О" на "I" ведет к уменьшению показания счетчика Zn .на I. Логический "О" и статическая "1" влияния 'не имеют.
!=F !=D !=G	Сравнение на «равно» Содержимое аккумулятора, сравнивается со значением операнда, стоящего при этой операции. При равенстве результат логического сопряжения будет "I", при неравенстве - "О".
><F ><D ><G	Сравнение на «не равно» Содержимое аккумулятора сравнивается со значением операнда, стоящего при этой операции. При неравенстве результат логического сопряжения будет "I", в противном случае -"О".

Изображение	Краткое описание
>F >D >G	Сравнение на больше Содержимое аккумулятора сравнивается со значением операнда, стоящего при этой операции. Если содержимое аккумулятора больше этого значения, тогда результат логического сопряжения будет "I", в противном случае -"O" .
>=F >=D >=G	Сравнение на больше или равно Содержимое аккумулятора сравнивается со значением операнда, стоящего при этой операции. Если содержимое аккумулятора больше или равно этому значению, результат логического
<F <D <G	Сравнение на меньше Содержимое аккумулятора сравнивается со значением операнда, стоящего при этой операции. Если содержимое аккумулятора меньше этого значения, то результат логического сопряжения будет "I", в противном случае -"O".

Изображение	Краткое описание
>=F >=D >=G	Сравнение на «меньше» или «равно» Содержимое аккумулятора сравнивается со значением операнда, стоящего при этой операции. Если содержимое аккумулятора меньше или равно этого значения, то результат логического сопряжения будет "I", в противном случае -"O".
+F +G	Сложение Содержания обоих аккумуляторов суммируются в соответствии с представлением числа. Результат хранится в аккумуляторе до дальнейшей обработки.
-F -G	Вычитание Содержание основного аккумулятора вычитается из вспомогательного /в соответствии с представлением числа/. Результат хранится в аккумуляторе до дальнейшей обработки.
XF XG	Умножение Содержания обоих аккумуляторов перемножаются в соответствии с представлением числа. Результат хранится в аккумуляторе до дальнейшей обработки.
:F :G	Деление Содержание вспомогательного аккумулятора делится на содержание основного /в соответствии с представлением числа/. Результат хранится в аккумуляторе до дальнейшей обработки.

Изображение	Краткое описание
L-	Загрузка /вообще/ Значение операнда, стоящего при операции загрузки, вводится в аккумулятор.
L-	Загрузка /значения времени/ показания счетчика/ Указанное при этой операции значение времени или состояния счетчика загружается в двоичном коде в аккумулятор /без бита состояния и растра времени/.
LC-	Загрузка кодированная / значения времени/показания счетчика, Указанное при этой операции значение времени или состояния счетчика загружается в аккумулятор в двоично-десятичном коде /без бита состояния, однако с растром времени/.
T-	Перенос Содержимое аккумулятора переносится к операнду, стоящему при операции переноса.
SPA PBn	Абсолютный вызов блока программы Линейная обработка программы прекращается. Она будет продолжена в указанном блоке программы PBn.

Изображение	Краткое описание
SPB PBn	Условный вызов программного блока При логическом результате "I" линейная обработка программы прекращается и продолжается в указанном блоке программы PBn . При результате логического сопряжения "O" эта операция не выполняется и обработка программы продолжается.
SPA SBn	Абсолютный вызов шагового блока Линейная обработка программы прекращается и будет продолжена в указанном шаговом блоке SBn
SPB SBn	Условный вызов шагового блока При логическом результате "I" линейная обработка программы прекращается и продолжается в указанном шаговом блоке. При результате 0" эта операция не выполняется и обработка программы продолжается
SPA FBn /Указание параметра блока/	Абсолютный вызов функционального блока Линейная обработка программы прекращается. Она будет продолжена, в указанном функциональном блоке FBn.

Изображение	Краткое описание
SPB FBn /Указание параметра блока/	Условный вызов функционального блока Если результат логической операции равен "I", линейная обработка программы прекращается и будет продолжена в указанном Функциональном блоке PBn. Если результат "O", эта операция не выполняется и обработка программы продолжается в обычном порядке.
A DBn	Вызов блока данных С вызовом блока данных происходит выбор области данных. Все последующие операции с областью операнда данных D будут относиться к блоку данных DBn.
BE	Конец блока Обработка программы в "старшем" блоке будет продолжена после вызова только что обработанного блока.
BEB	Условное окончание блока Если результат логической операции равен "I", линейная обработка программы прекращается и будет продолжена в "старшем" блоке после вызова только что обработанного блока. Если результат "O", эта функция не выполняется и обработка программы продолжается в обычном порядке.

Изображение	Краткое описание
STP	Стоп Обработка программы прерывается. Процессор останавливается. Новый или повторный запуск возможен только вручную.

1.3 Запас операций, выполняемых устройством автоматизации

В предлагаемой таблице показано распределение функций языка программирования STEP 5 в соответствии с объемом возможностей устройств автоматизации /программируемых контроллеров ПК/

Операции	Обозначение на языке STEP 5	У-ва автоматизации S5- 110A	130A	130W 100S	150A 150K	150S
Логические	U-, O-, UN-, ON- U(,), O O(	x x*	x x	x x x	x x x	x x x
Установки	=-, S-, R-	x	x	x	x	x
Времени и счета	SI-, SV-, R-, ZR- SE-, SS-, SA-, ZV-		x	x x	x x	x x
Загрузки и переноса	L-, T-, LC-		x**	x x	x x	x x
Сравнения	!=F, >F, <F <=F, >=F, ><F !=D, >D, <D, <=D >=D, ><D, !=G, >G, <G, <=G, >=G, ><G			x	x x	x x x
Вычислительные	+F, -F XF, :F, +G, -G, XG, :G			x	x	x x x
С блоками	BE BEB SPA-, SPB-, A-	x x	x x	x x x	x x x	x x x
Организационные	STP			x	x	x

- Начиная с центрального блока *6ES5900-7AD21*
- В устройствах 130 А и 130 К операции загрузки и переноса зависят от результата логической операции.

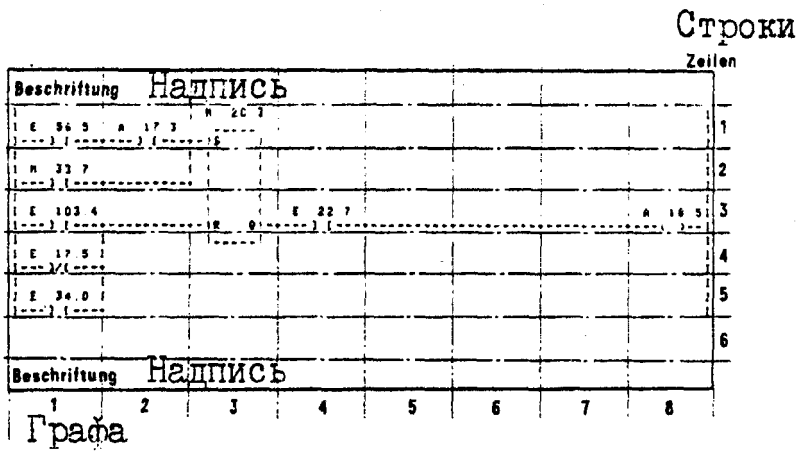
1.4 Графическое изображение на экране дисплея

Функциональная схема

Изображение на экране разделено по ширине на 8 зон /"уровней"/. Функциональный символ или обозначение входа или выхода функционального символа занимают на экране один уровень. Надпись, обозначающая: входы и выходы, стоит непосредственно у символа функции. Все соотношение начинается с левого края экрана.

Изображение на экране можно продвигать вверх. Одно логическое соотношение /операция/, - например, для какого-то выхода занимает по объему максимально 2 1/2 экрана.

Контактная схема [1)]



Поверхность экрана, занимаемая изображением, делится на 8 граф и 6 строк. Величина этих 48 полей составляет 10 знаков /ширина/ на 3 строки /экрана/. Логическое соотношение занимает 7 первых граф, а выходы - 8-ю графу. Контакты одного сопряжения изображаются у левого края экрана или на одной ветви.

В каждом из полей помещается символ контакта и относящееся сюда обозначение операнда. Вертикальные связи между контактами /ответвления/ изображаются вдоль границы между полями.

Для сложных функциональных символов, в зависимости от величины, отводят несколько полей.

Изображение на экране можно продвигать вверх.

На каждую цепь /каждый выход/ могут программироваться макс.

до 16 строк.

[1)] Некоторые специалисты называют такое изображение лестничной диаграммой. - Прим.переводчика.

ЛОГИЧЕСКИЕ ОПЕРАЦИИ

В данной главе рассматриваются логические соотношения И и ИЛИ, а также комбинации этих двух функций. Эти Функции в языке программирования STEP 5 имеют три принципиально различающихся между собой способа изображения. Поэтому эти три способа изображения - функциональная схема, таблица команд и контактная схема - приводятся и описываются параллельно. Описание строится в зависимости от рассматриваемого способа изображения.

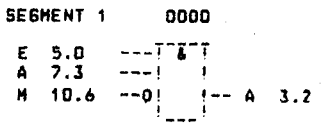
В разделе *2.1* описывается функция И /последовательная схема/, в разделе *2.2* - функция ИЛИ /параллельная схема/. В разделах 2.3 - 2.5 содержатся комбинации этих обеих функций.

Примеры в разделах 2.1 - 2.5 описаны так, как если бы к устройству автоматизации /ПК/ были подключены внешние нормально открытые - НО - контакты. Если к устройству автоматизации подключены внешние нормально закрытые - НЗ - контакты, это должно быть учтено при программировании. В разделе 2.6 приведен пример, где в качестве датчиков выступают как НО- так и НЗ - контакты.

Не все встречающиеся логические соотношения допускают прямое программирование на языке STEP-5. В разделе 2.7 дано руководство по преобразованию таких функций. Глава "Логические операции" завершается одним из примеров программирования. В разделе 2.8 на примере одной из задач управления описываются решения этой задачи с помощью функциональной схемы, таблицы команд и контактной схемы. В устройстве автоматизации s 5-150S возможен .опрос данных в двоичном виде. Область операнда простирается от D 0.0 до D255.I5. При пользовании операндом D следует обращать внимание на правильный предварительный выбор блока данных. Операнд данных D всегда обращается к только что вызванному блоку данных.

2.1 Функция И

Функциональная схема



Графически функция И изображается прямоугольником с символом " &". Слева находятся входы этой функции. На этих входах происходит опрос операндов: в нашем примере это вход Е 5.0, выход А,7.3 и метка М 10.6.

На соединенных входах /Е 5.0 и А 7.3/ происходит опрос операндов на состояние сигнала «1», т.е. если эти операнды имеют состояние сигнала «1», то и результат опроса будет «1». Результат опроса является тем состоянием сигнала, который имеет "прямое" воздействие на символ функции. Если эти операнды имеют состояние сигнала «0», то результат опроса «0».

На входах со знаком отрицания /М 10.б/ операнды опрашиваются на состояние сигнала «0», т.е. результат опроса будет равен «1», если эти операнды имеют состояние сигнала «0». При состоянии сигнала на входах этих операндов равном "J" результат опроса будет равен «0».

Если все входы функции И имеют результат опроса «1», эта функция "выполнена". Тогда на находящемся справа выходе появится результат логической операции «1». В нашем примере выход А 3.2 имеет состояние сигнала «1», если вход Е 5.0 и выход А 7.3 имеют состояние сигнала «1», а метка MIO.6 - состояние сигнала «0».

Если один или несколько входов функции И имеют результат опроса «0», то эта функция не выполнена. На выходе появится результат сопряжения «0». В нашем примере тогда на выходе А 3.2 состояние сигнала «0».

Количество входов одной функции И и количество функций И теоретически может быть любым. Точно также любыми могут быть последовательность и соотношение смешиваемых инвертируемых и не инвертируемых входов.

Таблица или список команд

```
SEGMENT 1
0000      :U   E 5.0        UND-Funktion
0001      :U   A 7.3
0002      :UN  M 10.6
0003      :=   A 3.2
0004      :***
```

Б таблице команд операнды опрашиваются поочередно и результат опроса сопрягается по функции И. Опрос на состояние сигнала «1» и сопряжение опрошенного состояния сигнала по функции И обозначается как операция "И". При этой операции стоит операнд, который указывает, что будет опрашиваться. В нашем примере это вход Е 5.0 и выход А 7.3. Если состояние сигнала этих операндов «1», то и результат опроса «1». Результат опроса является тем состоянием сигнала, над которым будет производиться логическая операция /сопряжение/. Если состояние сигнала этих операндов «0», результат опроса равен «0».

Опрос на состояние сигнала «0» и сопряжение опрошенного состояния сигнала по функции И обозначается как операция "UN". В нашем примере на состояние сигнала «0» опрашивается метка М 10. Результат опроса равен «1», если этот операнд имеет состояние сигнала «0». Если операнд, опрошенный на «0», имеет состояние сигнала «1», результат опроса будет «0».

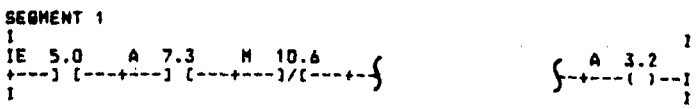
В первой команде процессор опрашивает вход Е 5.0. ("Первичный опрос", см. также раздел 2.3). Результат, опроса запоминается. По следующей команде опрашивается выход А 7.3. Результат этого опроса логически соотносится с уже находящимся в процессоре результатом первого опроса по функции И и образуется новый результат логического сопряжения (VKE - сокращение немецкого слова Verknupfungsergetmis).

Этот логический результат запоминается и сопрягается с результатом следующего опроса. В конце логической операции по сопряжению получают результат, предназначаемый для дальнейшей обработки.

Сопряжение выполнено, если результат логических операций равен «1», и не выполнено, если он равен «0». Функция И считается выполненной тогда, когда все опросы этой функции дают в результате «1». Если один или несколько опросов дадут в результате «0», логическое сопряжение не выполнено. С помощью результата логического сопряжения можно, например, управлять каким-либо выходом. В нашем примере результат логического сопряжения по функции И

присвоен выходу А 3.2 (=А3.2). При выполненном сопряжении происходит установка выхода А 3.2 /т.е. состояние его сигнала равно «1»/. В нашем примере это имеет место тогда, когда вход Е 5.0 и выход А 7.3 имеют состояние сигнала «1», а метка М 10.6 - состояние сигнала «0». Если логическое сопряжение не выполнено, происходит сброс выхода А 3.2, т.е. состояние сигнала будет равно «0».

Контактная схема



Функция И изображается на контактной схеме как схема последовательных включений. Опрошенные операнды представляются как символ контакта. Признак операнда и параметр находятся над контактом. Если символ контакта обозначает нормально открытый контакт, то относящиеся к нему операнды /в нашем примере Е 5.0 и А 7.3/ опрашиваются на состояние сигнала «1». При состоянии сигнала этих операндов «1» происходит "срабатывание" НО-контакта, т.е. символ контакта можно представить себе как "закрытый". При состоянии сигнала этих операндов «0» НО - контакт не "срабатывает", Символ контакта остается прежним - НО.

Если символ контакта обозначает нормально закрытый контакт, операнды /в нашем примере М 10.6/ опрашиваются на состояние сигнала «0». При состоянии сигнала этих операндов «1» происходит "срабатывание" НЗ - контакта, т.е. символ контакта меняется на НО. При состоянии сигнала этих операндов "0" НЗ - контакт не " срабатывает". Символ контакта остается прежним - НЗ.

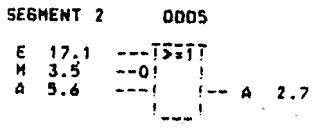
При замыкании всех контактов последовательной схемы в цепи появится электрический ток. Этот ток окажет воздействие на выход А 3.2. При замыкании всех контактов на выходе А.3.2 будет состояние сигнала «1» /реле "сработало"/. Если один из контактов открыт, состояние сигнала на выходе А 3.2 будет «0» /реле "отпало"/

В нашем примере выход А 3.2 только тогда - имеет состояние сигнала «1», если вход Е 5.0 и выход А 7.3 имеют состояние сигнала «1» /т.е. НО-контакты замкнуты/, а метка М 10.6 имеет состояние сигнала «0» /НЗ-контакты остаются закрытыми/.

Количество контактов, а также количество цепей теоретически может быть любым. Практически же они могут быть ограничены, например, шириной экрана дисплея.

2.2 Функция ИЛИ

Функциональная схема



Функция ИЛИ графически изображается прямоугольником с символом „>= "1". Слева изображаются входы к этой функции. На этих входах происходит опрос операндов. В нашем примере это вход E 17.1, метка M 3.5 и выход A 5.6.

Операнды E 17.1 и A 5.6 опрашиваются на состояние сигнала «1». Если эти операнды имеют состояние сигнала «1», результат опроса равен «1».

Операнд M 3.5 опрашивается на состояние сигнала «0». Результат опроса на этом, входе будет равен «1», если операнд имеет состояние сигнала «0».

Если один или несколько входов в результате опроса дают «1», функция ИЛИ выполнена. Если результат опроса всех входов «0», функция ИЛИ не выполнена. Выполнение функции ИЛИ дает на выходе A 2.7 состояние сигнала «1». При невыполненной Функции ИЛИ состояние сигнала на выходе A 2.7 равно «0».

В нашем примере состояние сигнала на выходе A 2.7 равно «0» только тогда, когда вход E 17.1 и выход A 5.6 имеют состояние сигнала «0», а метка M 3.5 - «1».

Количество входов в одной функции ИЛИ и количество функций ИЛИ теоретически может быть любым. Точно также любыми могут быть последовательность и соотношение инвертированных и не инвертированных входов.

Таблица команд

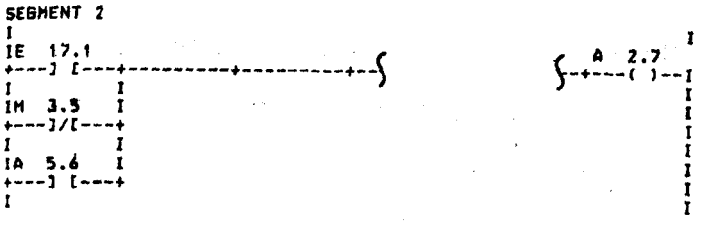
```
SEGMENT 2
0005      :O    E 17.1      ODER
0006      :ON   M 3.5
0007      :O    A 5.6
0008      :=    A 2.7
0009      :BE
```

В таблице команд операнды опрашиваются последовательно и результат опроса обрабатывается по функции ИЛИ. Опрос на состояние сигнала «1» и сопряжение результата опроса по функции ИЛИ обозначается как операция «0». В нашем примере с операцией «0» связаны операнды E.17.1 и A 5.6. Если состояние сигнала этих операндов «1», то и результат опроса равен "1".

Опрос на состояние сигнала «0» и сопряжение опрошенного состояния сигналов по Функции ИЛИ обозначается как операция "ON". В нашем примере это метка M 3.5. Если состояние сигнала метки «0», то результат опроса - "1".

Обработка команд в процессоре происходит последовательно. Результат опроса первой команды /OEГ 17.I/ запоминается процессором и соотносится с результатом опроса следующей команды по (функции ИЛИ /"Первичный опрос", см. также раздел S.3/. Вытекающий из этой операции результат вновь запоминается процессором и соотносится с результатом опроса следующей команды. Стоящий в конце операции результат может обрабатываться дальше. Функция ИЛИ считается выполненной, если один или несколько операндов в результате опроса дают «1». Если все операнды этой функции в результате опроса дают «0», она считается невыполненной. В нашем примере выход A 2.7 устанавливается только при условии выполнения функции ИЛИ и снова сбрасывается, если функция ИЛИ не выполнена. Выход A 2.7 только тогда имеет состояние сигнала «0», если вход E 17.1 и выход A 5.6 имеют состояние сигнала «0», а метка M 3.5 - «1».

Контактная схема



Функция ИЛИ на контактной схеме изображается как схема параллельных включений. Опрошенные операнды представляются как символ контакта и обозначаются над контактом.

Если символ обозначает НО - контакт, то относящиеся к нему операнды /в нашем примере вход E 17.1 и выход A 5.6/ опрашиваются на состояние сигнала «1». НО - контакт "сработал" " /закрыт/, если соответствующие операнды имеют состояние сигнала «1».

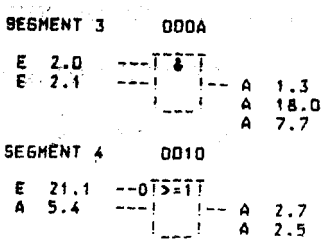
Если символ обозначает НЗ - контакт, то операнды /в нашем примере метка M 3.5/ опрашиваются на состояние сигнала «0». КЗ - контакт "сработал" /открыт/, если соответствующие операнды имеют состояние сигнала «1».

Если в параллельной схеме замкнуты один или несколько контактов, в цепи появляется электрический ток. Этот ток окажет воздействие на выход A 2.7. Если, как минимум, замкнут один контакт, состояние сигнала на выходе A 2.7 будет «1». Если все контакты открыты, состояние сигнала на выходе A 2.7 «0». В нашем примере выход A 2.7 имеет состояние сигнала «0» только тогда, когда вход E 17.1 и выход A 5.6 имеют состояние сигнала «0», а метка M 3.5 - «1».

Количество, контактов, а также количество цепей теоретически может быть любым. На практике оно ограничивается, например, шириной экрана дисплея.

2.3 Управление несколькими выходами

Функциональная схема



Результат логической операции на выходе символа функции может управлять несколькими операндами в первом примере - выходы A 1.3, A 18.0 и A 7.7/. Эти выходы устанавливаются один за другим и все реагируют одинаково.

Список команд

```
SEGMENT 3
0009        :U    E 2.0
000B        :U    E 2.1
000C        :=    A 1.3
000D        :=    A 18.0
000E        :=    A 7.7
000F        :***

SEGMENT 4
001D        :ON   E 21.1
0011        :O    A 5.4
0012        :=    A 2.7
0013        :=    A 2.5
0014        :***
```

Результатом этого соотношения могут управляться несколько выходов. Операции для выходов просто программируются одна за другой.

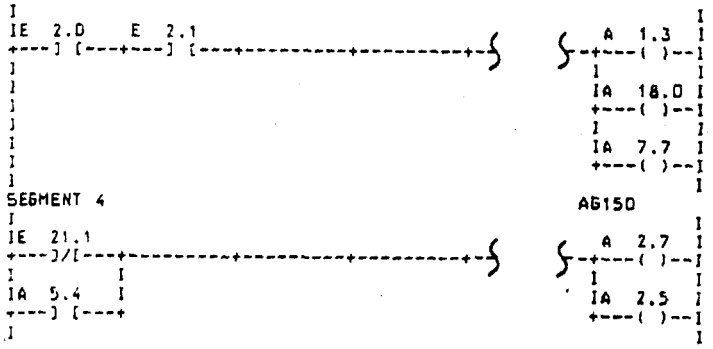
В первом примере результатом соотношения, образованным с по мощью функции И, управляются выхода А 1.3, А 18.0 и А 7.7. Все эти выходы реагируют одинаково.

До тех пор, пока идет обработка операций для управления выходами, результат логического соотношения не изменяется. Новый результат образуется только лишь с последующими опросами. В нашем примере логический результат образуется через команды U E2.0 и U E2.1 и присваивается выходам А1.3 и А7.7. В течение присвоения результат логического соотношения не изменяется и действует без ограничений.

Только лишь со следующим опросом /команда ONE 21.I/ процессор начинает образование нового логического соотношения. Этот опрос называется первичным опросом. Первичный опрос это всегда первый' опрос после операции, исполнение которой зависит от результата логического соотношения "условная" операция.

Результат первичного опроса запоминается процессором без логического сопряжения, которое произойдет лишь при следующем опросе. Поэтому указание по логическому сопряжению при первичном опросе не является определяющим. Например, допускается замена п на 0 и наоборот, и соответственно UN на ON и наоборот. Однако, чтобы не вносить путаницы в программирование, этого следует избегать.

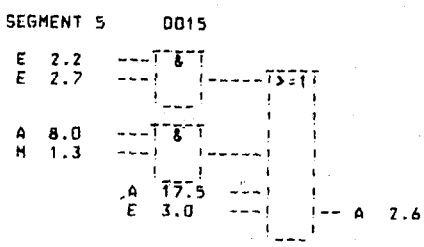
Контактная схема



Одна цепь может управлять несколькими "реле". В первом примере это выходы A 1.3, A 18.0 и A 7.7. Эти выходы устанавливаются один за другим и реагируют все одинаково.

2.4 Логическая операция И перед ИЛИ

Функциональная схема



В этой логической операции, составленной из .функций И и ИЛИ, выходы функций И ведут к функции ИЛИ. Результаты логических соотношений функций И сопрягаются с входами функции ИЛИ A 17.5 и E 3.0 по функции ИЛИ. Если одна из функций И выполнена, или если один из входов A 17.5 или E 3.0 имеет состояние сигнала «1», на выходе A 2.6 появляется состояние сигнала «1».

Таблица команд

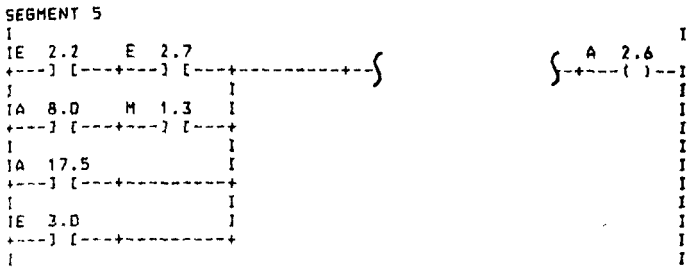
```
SEGMENT 5
0015      :U   E 2.2
0016      :U   E 2.7
0017      :O
0018      :U   A 8.0
0019      :U   M 1.3
001A      :O   A 17.5
001B      :O   E 3.0
001C      :=   A 2.6
001D      :***
```

I-я функция И
операция ИЛИ над функциями И
2-я функция И
функция ИЛИ

Это составленное из функций И и ИЛИ соотношение допускает в Булевой алгебре запись без скобок. Определяют, что "сначала" обрабатываются Функции И. После этого результаты сопряжения функций И сопрягаются по функции ИЛИ. Эта обработка "И перед ИЛИ" производится в специальном процессоре двоичных разрядов устройства автоматизации.

Первая функция И /E2.2 и E2.7/ связана со следующей функцией И /AO.8 и MI.3/ одним единственным 0 /обозначающим функцию ИЛИ/. Эта операция всегда необходима в тех случаях, когда функцию И ставят "перед" Функцией ИЛИ. Единственное 0 программируется перед функцией И ; после функции И оно больше не нужно. Входы, ведущие непосредственно к функции ""ИЛИ, программируются, как описано в разделе 2.2 /ОД 17.5 и ОЕ 3.0/. После этого получают общий результат всех логических операций, который присваивается выходу А 2.6.

Контактная схема

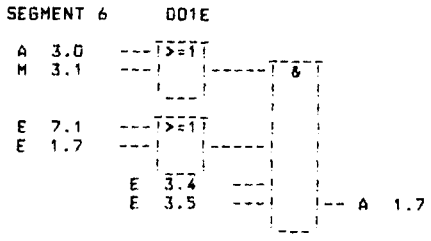


В этом сопряжении, построенном из последовательной и параллельной схемы, контакты параллельно включенных "ветвей" располагаются последовательно. Если " ток ", как минимум, течет по одной ветви, то он течет и по всей цепи, которая управляет выходом А 2.6. Тогда на выходе А 2.6 состояние сигнала будет «1».

Чтобы "ток" мог течь по одной из ветвей, нужно чтобы соответствующие контакты были "замкнуты", т.е. соответствующие операнды имели состояние сигнала «1». Е противном случае выход А..2.6 будет иметь состояние сигнала «0».

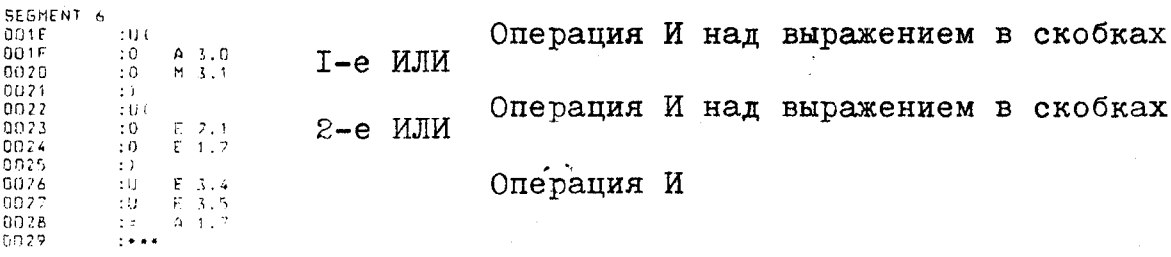
Логическая операция ИЛИ перед И.

Функциональная схема



В этой логической операции, составленной из функций И и ИЛИ, выходы функций ИЛИ ведут к одной функции И. Результаты функций ИЛИ увязываются с входами Е 3.4 и Е 3.5 по Функции И. Если обе функции ИЛИ выполнены, а оба входа Е 3.4 и Е 3.5 имеют «1», то на выходе А 1.7 появляется «1».

Таблица команд

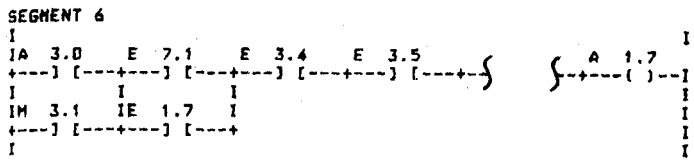


Это логическое соотношение, составленное из функций И и ИЛИ, по Булевой алгебре нужно писать со скобками, чтобы показать, что функции ИЛИ обрабатываются "перед" функцией И.

В языке программирования STEP 5 функции ИЛИ тоже пишутся в скобках. "Открытая скобка" всегда "комбинируется" с функцией И.

Программирование выражений в скобках происходит так, как описано Б разделе 2.2. Входы, непосредственно ведущие к функции И, программируются, как описано в разделе 2.1. , После чего результат логических увязок учитывается в целом и присваивается ВЫХОДУ А 1.7.

Контактная схема



В этом соотношении, составленном из последовательной и параллельной схем, внутри последовательной схемы находятся контакты, включенные по параллельной схеме. "Ток" пойдет по общей цепи, если будут замкнуты не менее одного из параллельно включенных контактов, а также оба последовательно включенных контакта /Е 3.4 и Е 3.5/. Соответствующие операнды тогда имеют состояние «1», также как и выход А 1.7. «0» на выходе А 1.7 будет в том случае, если открыты два из параллельно включенных контактов /напр., Е 7.1 и Е I.7/ или один из последовательно включенных контактов.

2.6 Учёт особенностей датчиков

При составлении программы, независимо от того, делается - ли она в виде Функциональной схемы, таблицы команд или контактной схемы, должно учитываться исполнение датчиков. Перед началом программирования должно быть известно, является ли используемый датчик НЗ - или НО - контактом.

Если датчик, подключенный к одному из входов, действует как НО - контакт, при воздействии на датчик на входе будет сигнал «1». Если датчик является НЗ - контактом, то при срабатывании на входе будет сигнал «0».

Устройство автоматизации не может распознавать, какой из типов контактов находится на входе. Оно различает только сигналы «1» или «0».

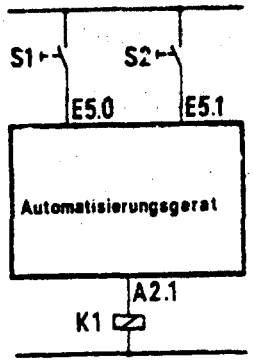
Так как программирование ориентируется на функцию датчика, следовательно, оно будет разным при разных исполнениях датчиков. Вход с НО - контактом должен рассматриваться иначе, чем вход с НЗ - контактом.

Используются, как правило, НО - контакты. Примеры, приводимые и рассматриваемые в этой книге, построены так, как если бы к входам были подсоединены НО - контакты. Случаи применения НЗ - контактов рассматриваются особо.

Однако, при реализации некоторых Функций управления не всегда можно использовать НО - контакт. Во многих случаях, например, в цепях готовности, применение НЗ - контактов является неизбежным. Различие в программировании наглядно показано в следующем примере: два датчика, подключенных к входам Е5.0 и Е 5.1, должны управлять выходом А 2.1. На выходе А 2.1 /К I/ сигнал «1» может быть только в том случае, если датчик на входе Е 5.0 (s1) сработал, а датчик на входе Е 5.1 (s2) - не сработал.

Случай (а)

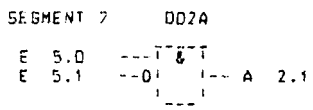
Оба датчика (s1) и (s2) являются НО - контактами



У-во автоматизации

Реле КI притягивает только в том случае, если кнопка S1 нажата, а кнопка S2 - нет.

Функциональная схема



Сигнал «1» на выходе А 2.1 будет только при условии выполнения функции "И". При срабатывании датчика S1 на входе Е 5.0 будет сигнал «1». Поэтому он

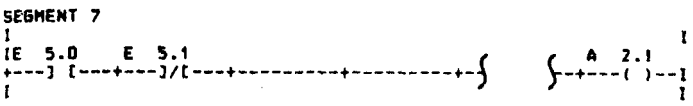
непосредственно идет на выполнение функции И. Если датчик S2 не задействован, на входе E 5.1 будет сигнал «0». Для выполнения функции И этот вход должен быть инвертирован .

Таблица команд

```
SEGMENT 7
0024      :U    E 5.0
0026      :UN   E 5.1
0028      :=    A 2.1
002A      :***
```

Сигнал «1» на выходе A 2.1 будет только в том случае, если результат логической операции, запрограммированной перед присвоением, равен «1». При функции И /как в нашем примере/ это будет иметь место в том случае, если в результате всех опросов получим «1». При срабатывании датчика S1 на входе E 5.0 будет сигнал «1», поэтому он опрашивается на состояние «1». Если датчик s2 остается открытым, на входе E 5.1 будет «0». Чтобы получить в результате опроса «1», этот вход нужно опрашивать на состояние «0».

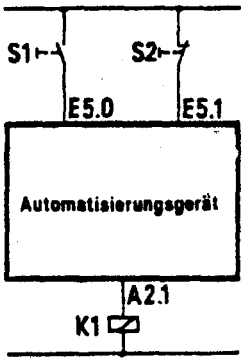
Контактная схема



На выходе A 2.1 сигнал «1» будет только в том случае, если -? в пепи есть "ток". В последовательной схеме /как в нашем примере/ это имеет место, если замкнуты все контакты. При срабатывании датчика S1 на входе E 5.0 появляется сигнал «1». Поэтому он программируется как НО - контакт. /Вход при сигнале «1» закрыт/. Если датчик S2 не задействован, на входе будет сигнал «0». Поэтому, чтобы "ток" мог проходить и через него, этот датчик должен программироваться как НЗ - контакт. /Вход при сигнале «0» закрыт/.

Случай (б).

Датчик S1 на входе E 5.0 является НО - контактом, а датчик S2 на входе E 5.1 является НЗ - контактом



У-во автоматизации

Реле К I притягивает только в том случае, если кнопка S1 нажата, а кнопка S2 - нет.

Функциональная схема

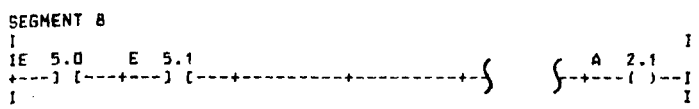
Сигнал «1» на, выходе А 2.1 будет только при условии выполнения Функции И. Если датчик s1 задействован, на входе Е 5.0 будет сигнал «1». Поэтому он непосредственно идет на выполнение Функции И. Если датчик S2 остается открытым, то на входе Е.5.I будет сигнал «1». Поэтому его тоже можно непосредственно вывести на функцию И.

Таблица команд

```
SEGMENT 8
002E        :U   E 5.0
002F        :U   E 5.1
0030        :=   A 2.1
0031        :***
```

Сигнал «1» на выходе А 2.1 будет только при условии, если результат всех опросов даст «1». При срабатывании датчика S1 на входе Е 5.0 будет сигнал «1». Поэтому он опрашивается на сигнал «1». Если датчик s 2 остается незадействованным, на входе Е 5.1 будет сигнал «1». Поэтому он также опрашивается на состояние «1».

Контактная схема

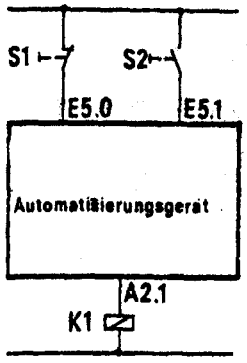


Сигнал «1» на выходе А 2.1 будет только в том случае, если

все контакты замкнуты. При срабатывании датчика S1 вход Е 5.0 имеет состояние «1». Поэтому он программируется как НО - контакт. Если датчик S2 остается незадействованным, на входе Е 5.1 также будет «1». Поэтому его тоже программируют как НО - контакт.

Случай (в)

Датчик S1 на входе E 5.0 является НЗ - контактом, а датчик S2 на входе E 5.1 является НО - контактом



У-во автоматизации

Реле К I должно притягивать только в том случае, если кнопка S1 нажата, а кнопка S2 - нет.

Функциональная схема

```
SEGMENT 9      0032
E  5.0   --0|  &  |
E  5.1   --0|     |-- A  2.1
            |_____|
```

Сигнал «1» на выходе А 2.1 будет только при условии выполнения функции И. Если датчик s1 задействован, "на входе E 5.0 будет сигнал «0». Поэтому его нужно вести к функции И в инвертированном виде. Если датчик s 2 остается открытым, на входе E 5.1 будет сигнал «0». Поэтому он точно также подается на функцию И в инвертированном виде.

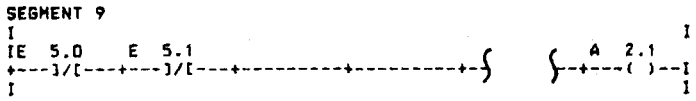
Таблица команд

```
SEGMENT 9
0032      :UN  E 5.0
0033      :UN  E 5.1
0034      :=   A 2.1
0035      :***
```

Сигнал «1» на выходе А 2.1 будет только при условии, если все опросы дадут результат «1». Если датчик S1 задействован, вход E 5.0 имеет сигнал «0». Поэтому он должен опрашиваться на «0».

Если датчик S2 остается незадействованным, то на входе E 5.1 будет сигнал «0». Поэтому он также должен опрашиваться на состояние «0».

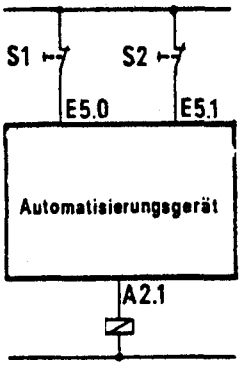
Контактная схема



Сигнал «1» на выходе A 2.1 будет только в том случае, если все контакты замкнуты. При срабатывании датчика S1 на входе E 5.0 будет сигнал «0». Поэтому он должен программироваться как НЗ - контакт. Если датчик S2 остается незадействованным, на входе E 5.1 будет сигнал «0». Поэтому этот датчик также должен программироваться как НЗ - контакт.

Случай(г)

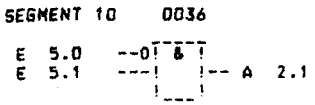
Оба датчика (S1 и S2) являются НЗ - контактами



У-во автоматизации

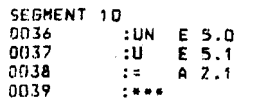
Реле К I притягивает только в том случае, если кнопка S1 нажата, а S2 -нет.

Функциональная схема



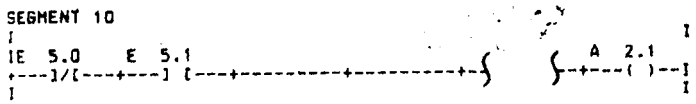
Сигнал «1» на выходе А 2.1 появляется только при условии выполнения функции И. Если датчик S1 задействован, на входе Е 5.0 будет сигнал «0». Поэтому к функции И его нужно вести в инвертированном виде. Если датчик S2 остается незадействованным, на входе Е 5.1 будет «1». Его к функции И можно вести напрямую.

Таблица команд



Сигнал «1» на выходе А 2.1 будет только в том случае, если результат всех опросов даст «1». Если датчик S1 задействован, на входе Е 5.0 будет сигнал «0». Поэтому он должен опрашиваться на состояние «0». Если датчик S2 остается незадействованным, на входе Е 5.1 будет сигнал «1». Он опрашивается на состояние «1»

Контактная схема



Сигнал «1» на выходе А 2.1 будет только при условии, что все контакты замкнуты. Если датчик s1 сработал, то на входе Е 5.0 будет сигнал «0». Поэтому он должен программироваться как НЗ-контакт. Если датчик S2 остается незадействованным, на входе Е 5.1 будет сигнал «1». Датчик программируется как НО-контакт.

Выводы

Общим для всех трех способов представления программ является тот такт. что программировать нужно с учетом состояния сигналов на входах. При этом безразлично, является ли датчик НО - или НЗ - контактом, срабатывает ли он или нет. Тем не менее можно вывести следующие общие правила:

Если датчик является НО - контактом и сработал или НЗ – контактом и не сработал т.е. сигнал на входе «1», этот вход выводится непосредственно на символ функции и, соответственно, должен опрашиваться на состояние «1» или программироваться как НО - контакт.

Если датчик является НО - контактом и не сработал или НЗ - контактом и сработал т.е. сигнал на входе "0", этот вход выводится на символ функции в инвертированном виде и, соответственно, должен опрашиваться на состояние «0» или программироваться как НЗ - контакт.

Тип датчика	Состояние датчика	Сигнал на входе	Способ представления программы		
			Функц. схема	Табл. команд	Контактная схема
НО-конт.	закрыт	"I"	—|	U- O-	┤├
НО-конт.	открыт	"0"	—o|	UN- ON-	┤/├
НЗ-конт.	закрыт	"0"	—o|	UN- ON-	┤/├
НЗ-конт.	открыт	"I"	—|	U- O-	┤├

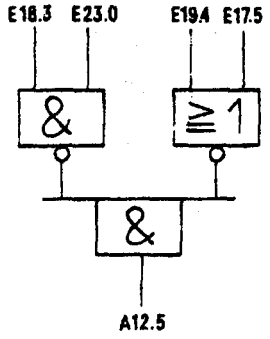
2.5 Преобразование логических операции

Не всякую исходную функциональную или принципиальную электрическую схему можно непосредственно перевести на язык STEP 5.

В функциональней схеме это может быть, к примеру, функция НЕ-И или НЕ-ИЛИ, а в электрической схеме - контакт между двумя цепями. Ниже показано одно из возможных представлений обоих случаев на языке программирования STEP 5.

Функция НЕ-И или НЕ-ИЛИ

Исходная функция:

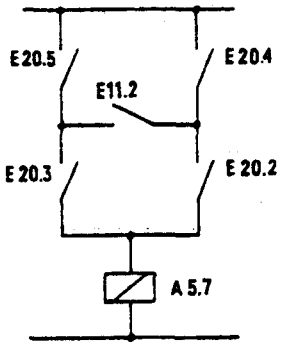


Для программирования этой задачи используются промежуточные метки, как описано в разделе 3.3. Результат логической операции над функцией И или ИЛИ запоминается в виде меток, которые опрашиваются на состояние «0».

```
A
   SEGMENT 11     003A
     E  18.3  ---|&|
     E  23.0  ---|  |-- M  100.0
                 |__|

   SEGMENT 12     003E
     E  19.4  ---|>=1|
     E  17.5  ---|  |-- M  100.1
                 |__|

   SEGMENT 13     0042
     M  100.0 --0|&|
     M  100.1 --0|  |-- A  12.5
                 |__|
```

Контакт между двумя цепями

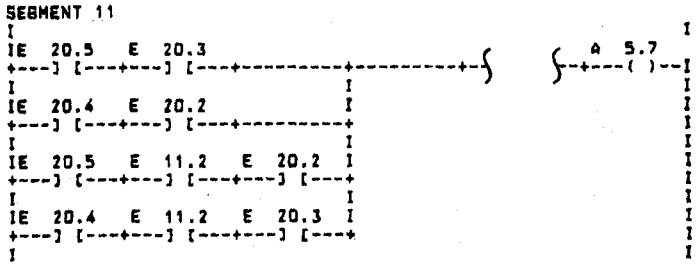
Исходная функция:



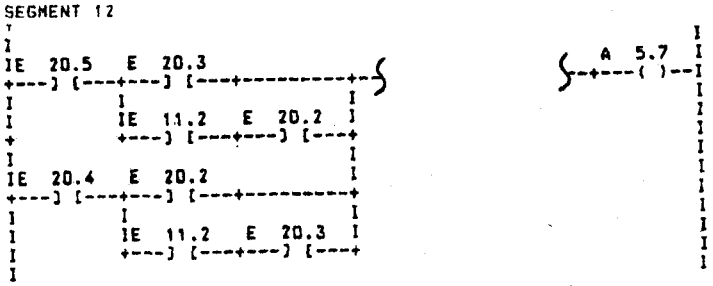
Реле A 5.7 сработает, если

- ▷ контакты E 20.5 и E 20.3 замкнуты или если
- ▷ контакты E 20.4 и E 20.2 замкнуты или если
- ▷ контакты E 20.5 и E II.2 и E 20.2 замкнуты или если
- ▷ контакты E 20.4 и E II.2 и E 20.3 замкнуты.

Вытекающая отсюда контактная схема выглядит следующим образом:



Или в обобщенном виде:



2.8 Пример программирования Контроль работы вентиляторов

Сигнальная лампа горит постоянно, если в рабочем режиме установки из 3 вентиляторов работают не менее двух. Сигнальная лампа должна мигать с частотой 0,5 Гц, если работает только один вентилятор, и 2 Гц, если не работает ни один. При выключении рабочего режима сигнальная лампа гаснет.

Сигнальная лампа	"Работа"	Работающие
не горит	откл.	-
горит постоянно	вкл.	2 или 3
мигает с частотой 0,5 Гц	Вкл.	1
мигает с частотой 2 Гц	вкл.	0

Функциональный план. Таблица команд

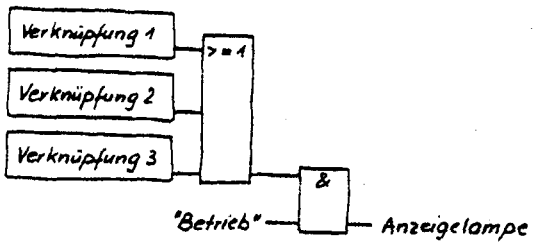
В предложенном примере есть три режима горения сигнальной лампы, в зависимости от количества работающих вентиляторов. Для решения этой задачи нам понадобятся три различных логических сопряжения:

- одно сопряжение, при котором лампа горит постоянно, если работают не менее 2 вентиляторов
- одно сопряжение, при котором лампа мигает с частотой 2 Гц, если ни один из вентиляторов не работает, и
- одно сопряжение, при котором лампа мигает с частотой 0,5 Гц, если работает только один вентилятор.

Сигнальная лампа будет гореть, если выполнена логическая Функция 1 или функция 2 иди функция 3. Результаты этих трех сопряжении охватываются таким образом одной функцией ИЛИ.

Сигнальная лампа должна гореть только, если выполнена Функция ИЛИ и включен режим "Работа". Если режим "Работа" не включен, сигнальная лампа не горит, независимо от количества работающих вентиляторов. Эта "деблокировка" сигнальной лампы в режиме "Работа" реализуется через Функцию И на выходе функции ИЛИ.

Схема, по которой мы будем программировать, выглядит тогда следующим образом:

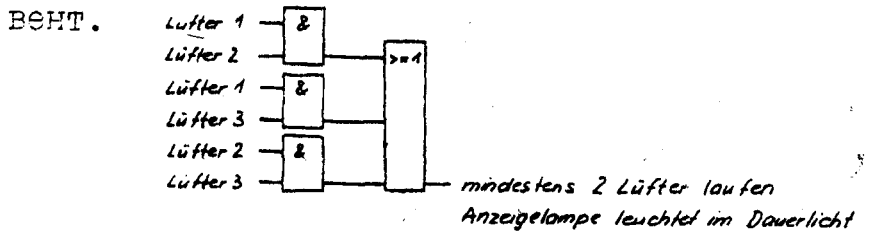


Лог.операция 1
Лог.операция 2
Лог.операция 3

"Работа" Сигн. лампа

Операция 1 считается выполненной, если работают минимум 3 вентилятора,. Таким образом, должны работать вентилятор 1 и вентилятор 2 или вентилятор 1 и вентилятор 3 или вентилятор 2 и вентилятор 3.

Изобразив это в виде функциональной схемы , мы получим



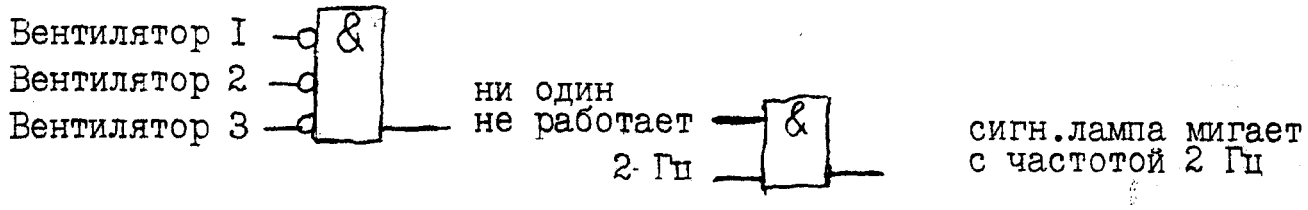
работав не менее 2 вентиляторов Сигнальная лампа горит постоянно

При работе всех трех вентиляторов это логическое соотношение также считается выполненным, так как в этом случае все функции И имеют сигнал «1».

На выходе логической операции 2 с частотой 2 Гц должен меняться сигнал «0» и «1», если ни один из вентиляторов не работает. Это значит:

вентилятор 1 не работает и вентилятор 2 не работает и вентилятор 3 не работает.

Если эта операция выполнена, по функции И включается частота 2 Гц. Таким образом логическую операцию 2 можно представить в виде следующего функционального плана:



По логической операции 3 определяют, когда работает только один вентилятор. Это имеет место, если не выполнено логическое сопряжение 1 и не выполнено условие "не работает ни один вентилятор". При выполнении функции И на выход логической операции 3 подается частота 0,5 Гц.

Для этого сопряжения нам требуется конечный результат логической операции 1 и промежуточный результат логической операции 2. Поэтому в соответствующих местах устанавливаются промежуточные метки /М 100.0 и М 100.1/, которые нужны для запоминания этих результатов /см.раздел 3.3/. Остальные операнды определяются по списку присвоений /распределения/.

Список присвоений:

E 33.1 вентилятор 1

E 38.2 вентилятор 2 сигнал «1», если вент.работает

E 38.3 вентилятор 3

M 99.1 частота 2 Гц M 99.2 частота. 0,5 Гц

M 100.0 промежуточная метка, имеет сигнал «1», если работают не менее 2 вентиляторов

M 100.1 промежуточная метка, имеет сигнал «1», если ни один вентилятор не работает

A 42.4 "Работа" при сигнале «1» A 51.7 Сигнальная лампа, горит при сигнале «1»

Если составить теперь таким образом полученные логические соотношения по вышеописанной схеме, мы получим полное изображение.

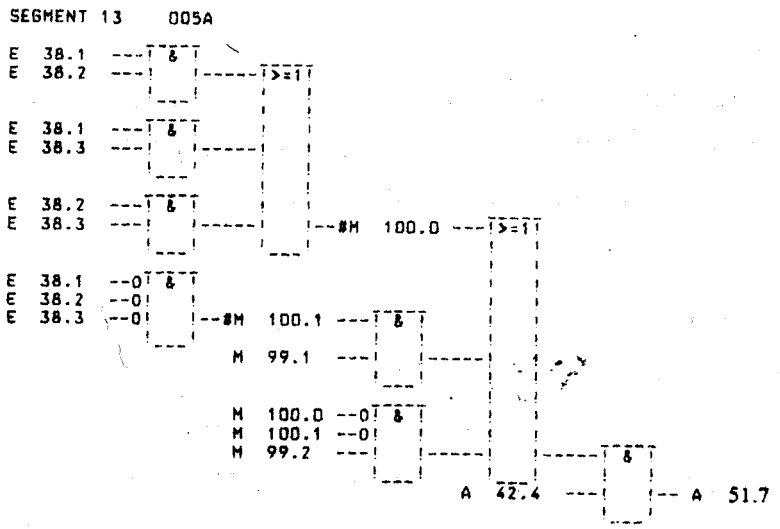
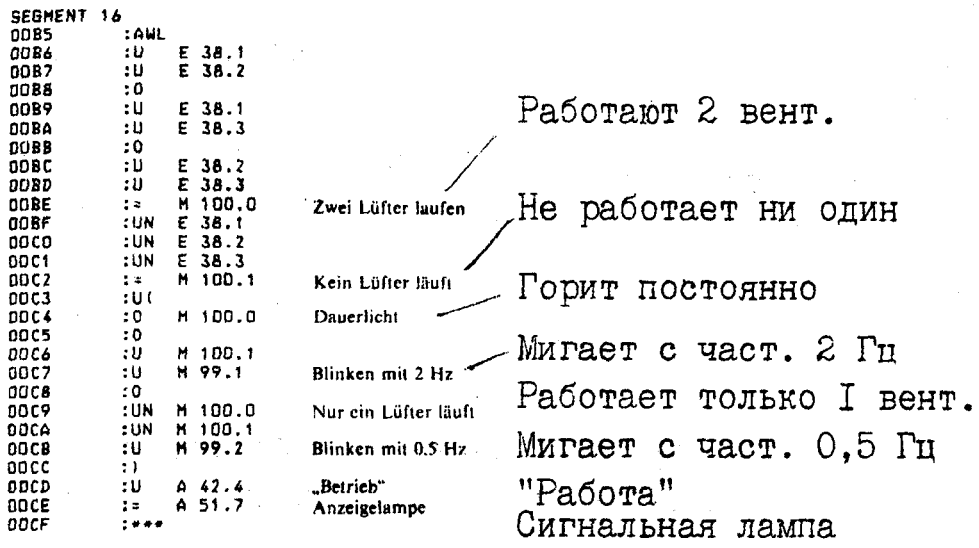


Таблица команд

Из только что полученного функционального плана можно непосредственно вывести таблицу команд:



В рассматриваемом примере есть три режима горения сигнальной лампы, в зависимости от количества работающих вентиляторов. Для решения этой задачи нам нужны три различных логических сопряжения /три цепи/:

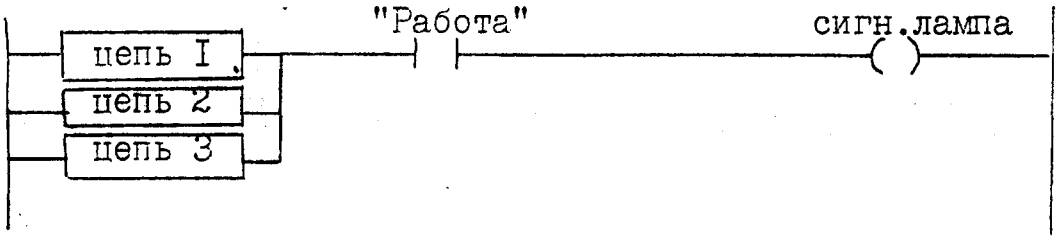
- Одно сопряжение, при котором лампа горит постоянно, если работают не менее 2 вентиляторов /цепь I/,
- одно сопряжение, при котором лампа мигает с частотой 2 Гц, если ни один из вентиляторов не работает /цепь 2/, и
- одно сопряжение, при котором лампа мигает с частотой 0,5 Гц, если работает только один вентилятор /цепь 3/

Сигнальная лампа горит, если цепь 1 иди цепь 2 иди цепь 3 пропускают "ток". Поэтому эти три цепи включаются параллельно.

Сигнальная лампа должна гореть только в том случае, если одна из трех цепей пропускает "ток" и включен режим "Работа". Если режим "Работа" не включен, лампа не горит, независимо от количества работающих вентиляторов. Для "деблокировки" сигнальной лампы к параллельной цепи последовательно подключается НО - контакт "Работа", при замыкании которого и происходит загорание лампы.

Схема, по которой мы будем программировать, выглядит теперь следующим образом:

Контактная схема



"Ток" в цепи 1 будет в том случае, если работают не менее 2 или 3 вентиляторов.

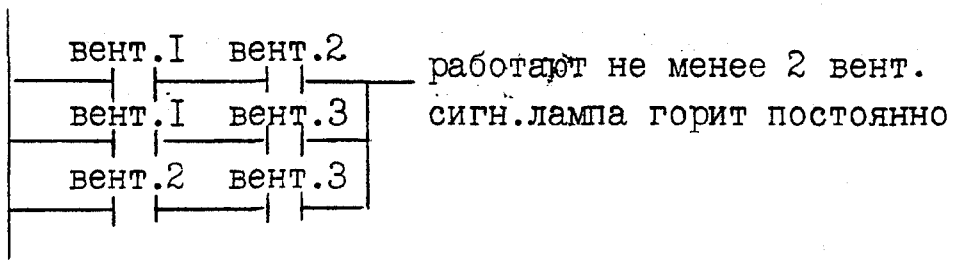
Таким образом, должны работать

вентилятор 1 и вентилятор 2 иди

вентилятор 1 и вентилятор 3 или

вентилятор 2 и вентилятор 3

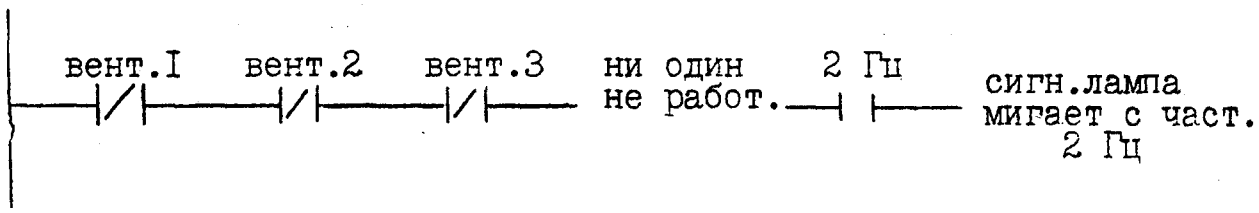
Представив в виде контактного плана, мы получим:



Если все три вентилятора работают, то это логическое соотношение также считается выполненным, так как все контакты замкнуты.

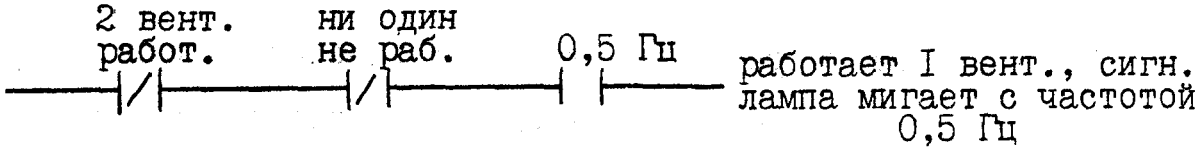
В цепи 2 с частотой 2 Гц чередуется наличие и отсутствие "тока", если не работает ни один вентилятор. Это можно представить: вентилятор 1 не работает и вентилятор 2 не работает и вентилятор 3 не работает.

В это соотношение последовательно включается контакт, закрывающийся и открывающийся с частотой 2 Гц. Если ни один вентилятор не работает, сигнальная лампа мигает с частотой 2 Гц. В виде контактной схемы это выглядит следующим образом:



В 3 цепи "ток" будет течь, если работает только один вентилятор. Это имеет место, когда не выполняются условия цепь 1 не пропускает "ток" и "не работает ни один вент".

Поэтому в цепи 1 и 3 устанавливаются метки промежуточных результатов /"вспомогательное реле"/ /см.раздел 3.3/. Размыкающие контакты этих реле включаются последовательно. В свою очередь последовательно к ним включается контакт с частотой 0,5 Гц, и мы получаем:



Эти цепи включаются согласно установленной вначале схеме и в сочетании с нижеследующим списком присвоений мы получим полную контактную схему.

Список присвоений

E 38.1 вентилятор 1

E 38.2 вентилятор 2 сигнал «1», если вент. работает

E 38.3 вентилятор 3

M 99.1 частота 2 Гц

M 99.2 частота 0,5 Гц

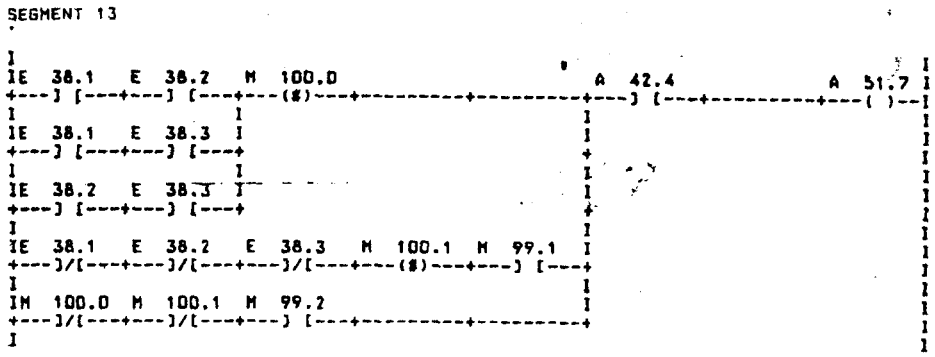
M 100.0 промежуточная метка, имеет сигнал «1», если работает не менее 2 вентиляторов

M 100.1 промежуточная метка, имеет сигнал «1», если ни один вентилятор не работает

A 42.4 "Работа" при сигнале «1»

A 51.7 Сигнальная лампа, горит при сигнале «1».

Контактная схема



3 ФУНКЦИИ ПАМЯТИ

К основным Функциям языка программирования STEP5 относятся также Функции памяти. Они изображаются одним и тем же графическим символом как при представлении программы в виде Функциональной схемы, так и в виде контактной схемы. Чистая функция памяти, RS-триггер, описывается в разделе 3.1

Принятый в электрических схемах прием получения запоминания путем самоподхвата разъясняется в разделе 3.2.

Функции памяти выступают не только как "статические" накопители. При сложных логических соотношениях иногда возникает необходимость в запоминании промежуточных результатов для обращения к ним в ходе последующего программирования. В качестве накопителя промежуточных результатов имеется область операндов "метки". Метку можно устанавливать, сбрасывать и опрашивать как выход /раздел 3.3/.

Оценка фронта сигнала позволяет распознавать программными средствами смену состояния сигнала. Реализация оценки фронта сигнала, которая на контактной схеме соответствует импульсному контакту, описывается в разделе 3.4. В разделе 3.5 раскрыто использование обработки фронта на примере D-триггера.

В разделе 3.6 поясняется программирование преобразователей двоичных сигналов в зависимости от того или иного способа представления программы: в виде функциональной схемы, контактной схемы или таблицы команд. Соответственно этому дано описание трех различных способов программирования.

Область операндов "метки" в устройствах автоматизации, как правило, защищена от исчезновения напряжения через батарею, т.е. метки характеризуются "реманентностью" /сохранностью/. При исчезновении напряжения они сохраняют состояние своих сигналов, которое можно будет опросить после восстановления напряжения. Путем обхода через метку можно получить реманентную характеристику и для выходов /раздел 3.7/.

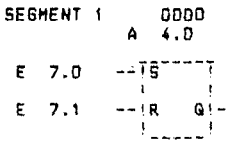
В устройствах автоматизации S5-110S,S5-130W,S5-150A,S5-150K и S5-150S для входов имеются регистры отображения процесса. В целях имитации входных сигналов можно устанавливать и сбрасывать отдельные биты в регистре отображения процесса /раздел 3.8/.

В разделе 3.9 показан случай использования Функций памяти. Описывается программирование отдельного звена управления в виде функциональной схемы, таблицы команд и контактной схемы с разными способами представления.

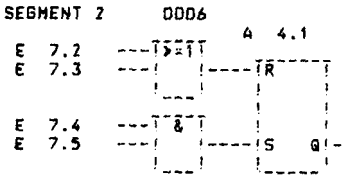
В устройствах автоматизации S5-150S можно производить установку, сброс и присвоение данных в двоичном виде. Область операндов простирается от D 0.0 до D 255.15. При использовании операнда данных D следует обращать внимание на правильность предварительного выбора блока данных. Операнд данных D всегда обращается в только что выбранный блок данных.

3.1 Rs-триггер

Функциональная схема Функция запоминания с приоритетом сброса



Если на обоих входах функции памяти сигнал «1», то приоритет отдается входу сброса. Функция запоминания будет или останется сброшенной. Сигнал «0» на обоих входах изменения на выходе не вызывает.



Если на обоих входах символа Функции сигнал «1», то приоритет отдается входу установки. Функция запоминания будет или сохранится установленной. Сигнал «0» на обоих входах изменения на выходе не вызывает.

Функция запоминания общая

Функции памяти на функциональных планах изображаются как прямоугольники с входами и 1 выходом. Установочный вход обозначается буквой S) сбрасывающий вход - R. Q обозначает выход функции памяти.

Если на входе установки «1», происходит установка функции запоминания. В нашем примере выход A 4.0 будет установлен, если на входе E 7.0 будет сигнал «1». "Выход установлен" - это значит, что на выходе сигнал «1».

Если на входе установки /при остановленной функции запоминания/ сигнал «0», функция памяти остается установленной. Она не изменит также своего сигнала, если вновь поступят сигналы «1» и «0».

Функция запоминания будет сброшена, если на вход сброса поступит сигнал «1». В нашем случае произойдет сброс выхода A 4.0, если на входе E 7.1 будет сигнал «1». "Выход сброшен" - это значит, что на выходе сигнал «0».

Если на входе сброса /при сброшенной функции запоминания/ сигнал «0», функция памяти остается сброшенной. Она не изменит также своего сигнала, если вновь поступят состояния «0» и «1».

Состояние сигналов на входах может представлять собой результат опроса /как в примере выше/ или результат логической операции. Таким образом и двоичные логические пункции можно ставить перед входами Функции памяти /пример ниже/.

Одной логической операции может соответствовать только, один элемент памяти. Его нельзя программировать в сочетании с функциями времени, счета и сравнения.

Таблица команд [1)]

Функция запоминания с приоритетом сброса

```
SEGMENT 1
0000      :U   E 7.0    Rücksetzen vorrangig      Приоритет сброса
0001      :S   A 4.0       Setzen                 Установить
0002      :U   E 7.1
0003      :R   A 4.0       Rücksetzen
0004      :NOP 0                                  Сбросить
0005      :***
```

При обработке как операций установки, так и сброса, с результатом логических сопряжении «1» выполняются обе операции. В данном случае значение имеет последовательность программирования /и тем самым последовательность обработки/. В рассматриваемом примере сначала выполняется операция установки; устанавливается выход A 4.0. После этого выполняется операция сброса, выход вновь сбрасывается.

Выход A 4.0 останется сброшенным до конца обработки программы.

Эта кратковременная установка выхода A 4.0 выполняется только в отображении процесса /исключение: ПК S5-110 A, см.раздел 10/. Во время обработки программы состояние сигнала на соответствующем блоке вывода не меняется. Информация из регистров отображения процесса на выходах переносится в блоки вывода только в конце программы.

1) Команда NOP0 обозначает входы и выходы на графических символах, не занятых операндами. На выполнение запрограммированных функций эта команда не влияет. При отказе от графического изображения можно опускать.

Функция запоминания с приоритетом установки

```
SEGMENT 2
0006      :O    E 7.2    Setzen vorrangig     Приоритет установки
0007      :O    E 7.3
0008      :R    A 4.1        Rücksetzen       Сбросить
0009      :U    E 7.4
000A      :U    E 7.5
000B      :S    A 4.1        Setzen
000C      :NOP  0                             Установить
000D      :***
```

При обработке как операций установки, так и сброса, с результатом логических сопряжении «1» выполняются обе операции. В данном случае значение имеет последовательность программирования и тем самым последовательность обработки/. В рассматриваемом примере сначала выполняется операция сброса ; сбрасывается выход А 4.1. После этого выполняется операция установки ;устанавливается выход А 4.1.

Выход А 4.1 остается установленным до конца обработки программы.

Этот кратковременный сброс выхода А 4.1 выполняется только в отображении процесса /исключение: ПК S5-110 А, см.раздел 10/. Во время обработки программы состояние сигнала на соответствующем блоке вывода воздействию не подвергается. Отображение процесса на выходах переносится в блоки вывода только в конце программы.

Функция запоминания общая

В языке программирования STEP 5 также известны операции для функций памяти. Операция s - "установка" и R -"сброс".

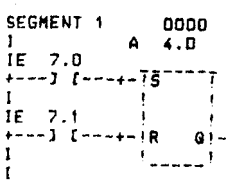
Если операция s обрабатывается с результатом сопряжения «1», происходит установка операнда. В нашем примере устанавливается выход А 4.0, если на входе Е 7.0 будет «1». "Выход установлен" - означает, что сигнал на выходе «1».

Если операция установки /при установленных операндах/ обрабатывается с результатом логических сопряжении «0», т.е. если на входе Е 7.0 вновь сигнал «0», операнд /выход А 4.0/ остается установленным. Он не изменит также своего сигнала, если вновь будет обрабатываться с результатом логического сопряжения «1» или «0». Операнд сбрасывается, если операция сброса обрабатывается с логическим результатом «1». В нашем примере сброс выхода А 4.0 произойдет в

том случае, если вход E 7.1 имеет сигнал «1». "Выход сброшен" - это означает, что на выходе сигнал «0». Если операция сброса /при сброшенных операндах/ обрабатывается с логическим результатом «0», операнд остается сброшенным. Он не изменит также своего сигнала, если повторить операцию сброса с логическим результатом «1» или «0». Перед операцией установки или сброса могут также стоять двоичные логические функции. Результат этих сопряжении скажется тогда при соответствующей операции.

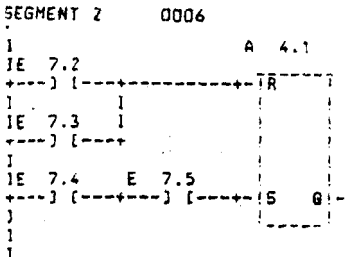
Контактная схема

Функция запоминания с приоритетом сброса



Если на обоих входах функции памяти сигнал «1», т.е. оба контакта E 7.0 и E 7.1 замкнуты, приоритет отдается входу сброса. Функция запоминания будет или останется сброшенной. Сигнал «0» на обоих входах изменения на выходе не вызывает.

Функция запоминания с приоритетом установки



Если на обоих входах символа функции сигнал «1», т.е. "ток" идет в оба входа, приоритет отдается входу установки. Функция запоминания будет или останется установленной. Сигнал «0» на обоих входах изменения выхода не вызывает.

Функция запоминания общая

Контактная схема также позволяет изображать функцию памяти в графическом виде с помощью того же символа, что и на функциональном плане. Символ имеет вход установки-S, вход сброса -R. Q обозначает выход функции памяти.

Если на входе установки сигнал «1», т.е. "ток" идет через вход установки, происходит установка функции запоминания. В нашем примере выход А 4.0 установится, если замкнут контакт Е 7.0. "Выход установлен" - это значит, что выход Q символа функции пропускает "ток".

Если на входе установки /при установленной функции запоминания/ сигнал «0», функция запоминания остается установленной и не изменит также своего состояния при повторном поступлении сигнала «0» и «1».

Функция запоминания сбрасывается, если на входе сброса будет сигнал «1», т.е. по входу сброса идет "ток". В нашем примере произойдет сброс выхода А 4.0, если замкнуть контакт Е 7.1.

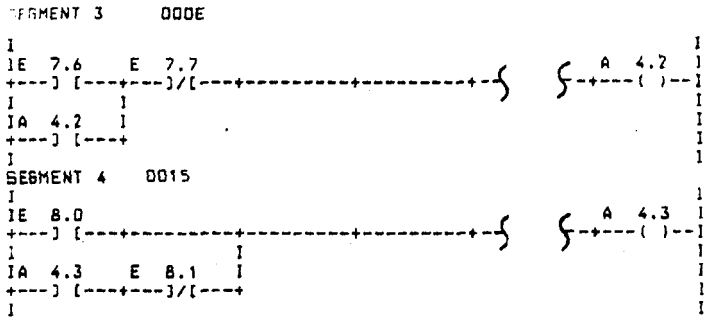
"Выход сброшен" - это означает, что "тока" на выходе Q символа функции нет.

Если на вход сброса /при сброшенной функции запоминания/ поступит сигнал "С", Функция памяти останется сброшенной. Не изменит она своего состояния и при повторном поступлении сигналов «0» и «1».

На одну цепь допускается изображение только одного элемента памяти. Его нельзя программировать в сочетании с функциями времени, счета и сравнения.

3.2 Запоминание путём самоподхвата

Контактная схема



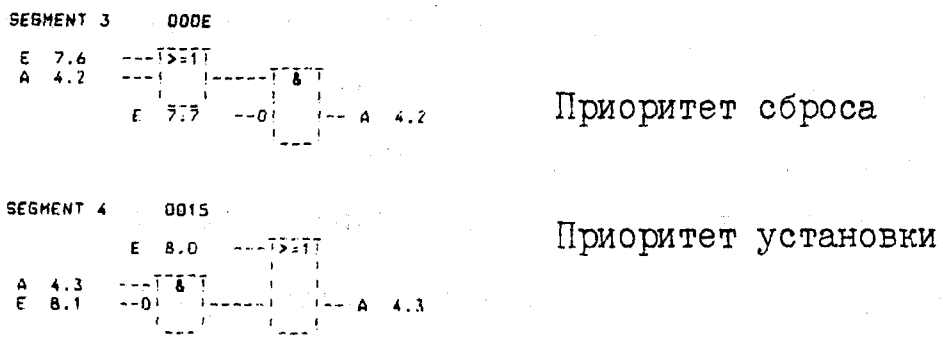
Принятое на принципиальных электрических схемах изображение функции памяти реализуется через самоподхвата управляемого выхода. Эта реализация: может быть перенесена и на контактную схему. Однако в отличие от изображения, описанного в разделе 3.1, ее недостаток в том, что функция памяти не распознается с первого взгляда. Контакт "выходного реле" при этом методе выводится параллельно к цепи, содержащей логическое сопряжение для установки выхода. Как только цепь для установки выхода /напр., вход E 7.6/ становится проводящей "ток", происходит установка выхода /выход A 4.2/; "реле" срабатывает. Теперь через параллельный контакт реле /выход A 4.2/ цепь перемкнется. Реле удерживает само себя, независимо от того проводит ли цепь для установки "ток" или нет

Для сброса выхода есть два варианта, в зависимости от приоритетов сброса и установки. Если приоритет за сбросом /верхний пример/^ цепь для сброса /вход E 7.7/ включается последовательно к операции для установки и самоподхвата. При необходимости произвести сброс выхода цепь прерывается. В нашем примере "реле" "отпадет", если контакт E 7.7 размыкает, т.е. если на входе E 7.7 сигнал «1». Если на обоих входах E 7.6 и соотв. E 7.7 сигнал «1», приоритет отдается сбросу. Реле /выход A 4.2/ отпадает или остается в этом состоянии, так как цепь прерывается контактом E 7.7.

Во втором варианте /нижний пример/ цепь для сброса включена последовательно с самоподхватом. В случае необходимости сброса выхода производится прерывание цепи. При отсутствии условия на установку "реле" "отпадает". Если на обоих

входах E 8.0 и E 8.1 сигнал «1», происходит установка выхода A 4.3, так как цепь для установки ведет непосредственно к реле A 4.3. В этом варианте приоритет отдается установке.

Функциональная схема



Функцию памяти можно реализовать также через самоподхват управляемого входа. Это изображение, принятое на принципиальных электрических схемах, обычно не применяется в функциональных схемах. Здесь пользуются функцией памяти, описанной в разделе 3.1, с тем преимуществом, что эта функция распознается с первого взгляда. Однако принципиальная реализация функции памяти через самоподхват возможна и на Функциональных схемах. В связи с чем мы и приводим ее описание /опираясь на изображение в виде контактной схемы/.

Как и в разделе 3.1,запоминание можно программировать как с приоритетом установки, так и с приоритетом сброса. На верхнем примере показано программирование с приоритетом сброса. С сигналом «1» на входе E 7.6 происходит установка выхода A 4.2 /при условии, что на входе E 7.7 сигнал «0»/. Состояние сигнала на выходе A 4.2 по функции ИЛИ сопрягается с условиями установки /вход E 7.6/, благодаря чему происходит самоподхват установленного выхода. Если на входе E 7.6 вновь появится сигнал «0», выход A 4.2 сохранит свое состояние.

Сброс выхода A 4.2 произойдет в том случае, если на входе E 7.7 будет сигнал «1». Так как тогда результат опроса этого входа будет «0» и логическая функция И не выполняется. Это будет иметь место и в том случае, когда условие установки выполнено; таким образом, приоритет за сбросом.

В нижнем примере приоритет отдан установке. Условие установки /вход Е 8.0/ управляет выходом А 4.3. Как только выход установлен, он удерживает себя сам. Самоподхвата не произойдет, если на входе Е 8.1 сигнал «1». Если результат-опроса на этом входе «0», логическое сопряжение И не выполняется и- выход сбрасывается. Если дополнительно к условию сброса выполняется условие установки, доминирует условие установки. Выход будет или останется установленным.

В нижнем примере приоритет отдан установке. Если на входе Е 8.0 сигнал «1», то и на выходе А 4.3 будет сигнал «1», независимо от состояния сигнала на входе сброса /вход Е 81/. Установленный выход может удерживаться через обратную связь, даже если условие установки больше не выполняется. Если в таком случае действует условие сброса /сигнал «1» на входе Е 8.1/, выход А 4.3 сбрасывается.

3.2 Запоминание двоичных промежуточных результатов

При сложных логических операциях иногда возникает необходимость в запоминании промежуточных результатов и их опроса и дальнейшей обработки в ходе последующей программы. Для этого промежуточного запоминания в распоряжении имеется область операндов "метки". Программно метка обрабатывается так же, как и выход. Однако она не имеет выхода "наружу". Кроме того, эта область операндов через имеющуюся в аппарате батарею получает питание даже в случае отключения напряжения, благодаря чему обеспечивается сохранность /реманентность/ записанной информации /см.раздел 3.7/. Для запоминания промежуточных результатов нельзя: брать регистры отображения процесса на выходах. Программно-техническое влияние на отображение процесса таких выходов у которых нет присвоенных им блоков вывода, учитывается процессором и интерпретируется им как сбой. Эта мера предусмотрена как защита от ошибочного программирования /или неисправных блоков/ и охватывает в устройствах автоматизации S5-150A, S5-150 K из5-150S не только выходы, но и входы.

Метка, используемая для промежуточного запоминания результатов логических операций, называется "промежуточной меткой".

Таблица команд

```
SEGMENT 3
000E      :U(
000F      :O    E 7.6
0010      :O    A 4.2
0011      :)
0012      :UN   E 7.7
0013      :=    A 4.2
0014      :***

SEGMENT 4
0015      :O    E 8.0
0016      :O
0017      :U    A 4.3
0018      :UN   E 8.1
0019      :=    A 4.3
001A      :***
```

Приоритет сброса
Самоподхват

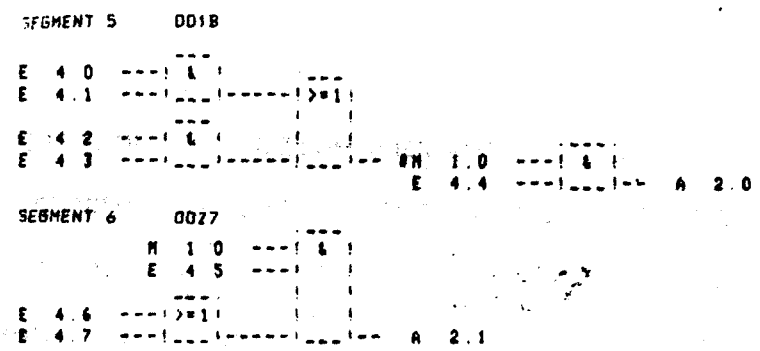
Приоритет установки
Само подхват

Функция памяти может программироваться также через присвоение, когда для операндов устанавливается "обратная связь". В этом случае состояние сигнала установленного операнда используется для самоподхвата. Хотя в принципе такая реализация функции памяти и возможна, тем не менее лучше использовать программирование через операции S -и R. Тогда запоминание распознается с первого взгляда. Опираясь на изображение в виде контактной схемы, мы познакомимся и с этой возможностью программирования. Этот метод также позволяет получать как преимущество для операций установки, так и приоритет для операций сброса. В верхнем примере приоритет отдан сбросу. Если условие сброса /вход Е 7.7/ имеет сигнал «1», результат опроса равен «0». Тем самым, как обусловлено функцией И, результат логического сопряжения также «0». Выход А 4.2 будет сброшен независимо от состояния сигнала условия установки /вход Е 7.6/. Выход может быть установлен только в том случае, если условие сброса больше не выполняется. Выход остается установленным даже в том случае, если после одного выполненного условия установки следующее условие установки больше не выполняется.

Метки, используемые в качестве промежуточных, могут неоднократно применяться внутри программы. При этом следует помнить, что промежуточная метка сначала устанавливается и потом опрашивается, и что при повторной установке одной и той же метки "старый" промежуточный результат теряет свое значение для дальнейшей обработки программы. Промежуточные метки, опрашиваемые общей программой, могут использоваться только один раз. Можно программировать метки и с запоминающей характеристикой. Здесь действуют те же правила, что и описанные в разделе 3.1 для области операндов выхода. Как и для выходов, запоминающее поведение меток достигается за счет самоподхвата.

Промежуточные метки

Функциональная схема



Внутри логических операций могут устанавливаться так называемые "промежуточные метки". Такие промежуточные метки обозначаются значком ..№" В этой промежуточной метке удерживается результат последней выполненной логической операции. Этот результат в свою очередь, в том числе многократно, может опрашиваться и увязываться с другими операциями.

В нашем примере производится промежуточное запоминание в метке М 1.0 результата логической операции по функции И – перед - ИЛИ /входы Е 4.0, Е 4.1, Е 4.2 и Е 4.3/. Результат этой операции сопрягается по функции И с входом Е 4.4 с воздействием на выход А 2.0. Этот результат сопрягается также с входом Е 4.5 и через функцию ИЛИ с входами Е 4.6 и Е 4.7 по функции И. В этом случае экономится повторное программирование функции И – перед - ИЛИ.

Промежуточную метку можно обозначить не только внутри логической операнды, стоящие в конце операции, также могут сопровождаться значком "#".

Таблица команд [1)]

```
SEGMENT 14
0098      :AWL
0099      :U   E 4.0
009A      :U   E 4.1
009B      :O
009C      :U   E 4.2
009D      :U   E 4.3
009E      :=   M 1.0     Zwischenergebnisspeicher
009F      :U   M 1.0     Abfrage des Zwischenergebnisses
00A0      :U   E 4.4
00A1      :=   A 2.0
00A2      :U   M 1.0     Abfrage des Zwischenergebnisses
00A3      :U   E 4.5
00A4      :U(
00A5      :O   E 4.6
00A6      :O   E 4.7
00A7      :)
00A8      :=   A 2.1
00A9      :***
```

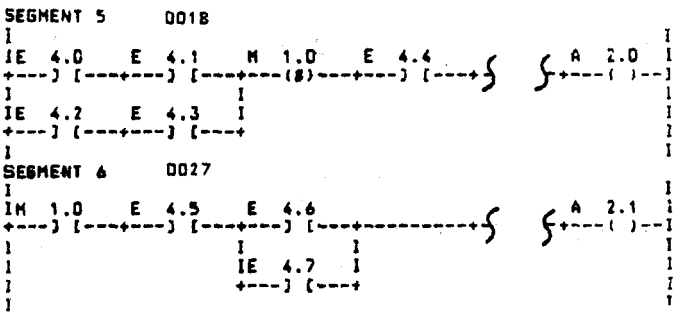
Промежуточный результат
Опрос промежуточного результата

Опрос промежуточного результата

В таблице команд промежуточная метка специального обозначения не имеет. Если какой-то определенный результат собираются использовать многократно, то устанавливают обычную метку /в нашем примере М 1.0/. Эта метка может теперь опрашиваться при дальнейших логических операциях.

1) Приведенная в данном примере таблица команд не соответствует графической функции функциональной или контактной схемы.

Контактная схема

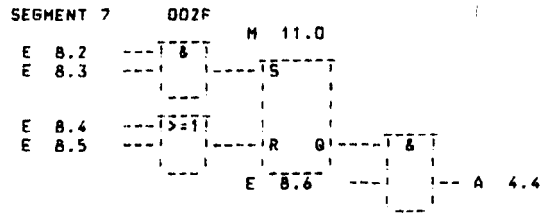


Внутри логических операций могут устанавливаться промежуточные метки. Такие промежуточные метки обозначаются значком "#". Они изображаются в виде "катушек реле". В этих промежуточных метках запоминается результат последней произведенной логической операции. Этот результат в свою очередь, в том числе многократно, может опрашиваться и увязываться с другими операциями. Опрос изображается в виде символа контакта. В нашем примере выход А 2.0 образован путем последовательного включения входа Е 4.4 и схемы из входов Е 4.0, Е 4.1, Е 4.2 и Е 4.3. Результат логического сопряжения этой схемы записан в метке М 1.0. В сочетании с входами Е 4.5,. Е4.6 и Е 4.7 он будет использован для управления выходом А 2.1.

Промежуточная метка ставится не только внутри логической операции. Операнды, стоящие в конце операции, также могут сопровождаться значком "#". К этому приему прибегают в тех случаях, когда нужно включить последовательно более 7 контактов /а ширины экрана дисплея не хватает/. Тогда изображение прерывается промежуточной меткой и продолжается со следующей цепи.

Метка с запоминанием

Функциональная схема



В указанном примере содержится метка с запоминанием /метка М 11.0/. Метка М 11.0 устанавливается в том случае, если вход Е 8.2 и Е 8.3 имеют сигнал «1». Она сбрасывается, если сигнал «1» на входе Е 8.4 или входе Е 8.5. Выход А 4.4 имеет сигнал «1» при установленной метке М 11.0 и сигнале «1» на входе Е 8.6. Символ функции запоминания с приоритетом установки приведен в разделе 3.1.

1) При этом приоритет за функцией сброса.

Таблица команд

```
SEGMENT 7
002F      :U(
0030      :U    E 8.2
0031      :U    E 8.3
0032      :S    M 11.0    Setzen des Zwischenergebnisspeichers
0033      :O    E 8.4
0034      :O    E 8.5
0035      :R    M 11.0    Rücksetzen des Zwischenergebnisspeichers
0036      :U    M 11.0    Abfragen des Zwischenergebnisspeichers
0037      :)
0038      :U    E 8.6
0039      :=    A 4.4
003A      :***
```

Установка промежуточного результата

Сброс промежуточного результата

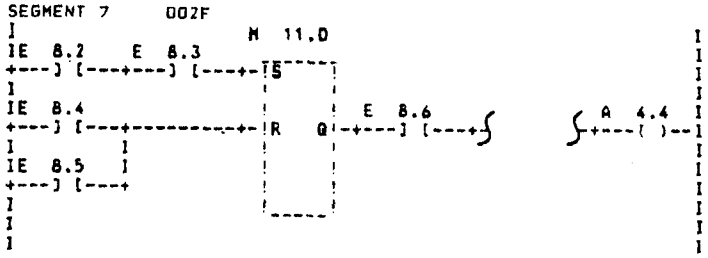
Опрос промежуточного результата

В указанном примере содержится метка с запоминанием
/метка М 11.0/.

Метка М 11.0 устанавливается в том случае, если вход Е 8.2 и Е 8.3 имеют сигнал «1». Она сбрасывается, если сигнал «1» на входе Е 8.4 или входе Е 8.5. Приоритет за функцией сброса, так как она запрограммирована после установки.

Выход A 4.4 имеет сигнал «1» при установленной метке M 11.0 и сигнале «1» на входе E 8.6.

Контактная_схема



В указанном примере содержится метка с запоминанием /метка M 11.0/. Метка M 11.0 устанавливается при условии, что цепь с контактами E 8.2 и E 8.3 проводит "ток". Она сбрасывается, если "ток" течет по цепи с контактами E 8.4 и E 8.5, при чем приоритет за функцией сброса. Установленная метка M 1.1.0 имеет на выходе Q "ток", в результате чего при закрытом контакте E 8.7 срабатывает реле A 4.4. Символ функции запоминания с приоритетом установки приведен в разделе 3.1.

3.4 Обработка фронтов, импульсный контакт

Обработка фронтов нужна для регистрации и оценки нарастающего или падающего фронта сигнала. "Фронтом" называют изменение состояния сигнала /напр., на входе/. Нарастающим фронтом называется изменение состояния сигнала с «0» на «1». В противном случае говорят о падающем Фронте. /Нарастающий фронт называют также передним, а падающий - задним фронтом сигнала. - Прим.перев./.

На принципиальной электрической схеме эквивалентом опенки фронта является импульсный контакт. Выдача импульса этим контактом при включении реле соответствует нарастанию фронта. Импульс от контакта при отключении соответствует падению фронта.

Распознание изменения сигнала происходит программными средствами. При каждом прохождении его через накопитель программ имеет место проверка состояния сигнала /напр., входа/ на изменение по сравнению с предыдущим прохождением. Чтобы иметь возможность сравнивать старое состояние сигнала на входе с новым, первое должно храниться в памяти. Эта задача возлагается на метки /"метки фронтов"/.

Отклонение состояния сигнала метки фронта от состояния сигнала на входе свидетельствует о наличии фронта, после чего происходит установка второй метки /"импульсной метки"/. Эта импульсная метка служит в качестве накопителя промежуточного результата и, соответственно, может неоднократно использоваться. При возникновении фронта состояние сигнала импульсной метки будет «1». Это состояние может в дальнейшем опрашиваться и использоваться в логических операциях. Импульсная метка непосредственно соответствует импульсному контакту.

После распознания, изменения фронта метка фронта принимает состояние сигнала на входе, с тем чтобы при следующем цикле обработки процессор не сделал бы повторного определения фронта. Импульсная метка как раз и служит для такой синхронизации сигналов.

Структура программы обработки Фронта, соответственно, импульсного контакта будет выглядеть следующим образом:

Начало

На входе другое состояние сигнала, чем на метке фронта?

Да

Нет

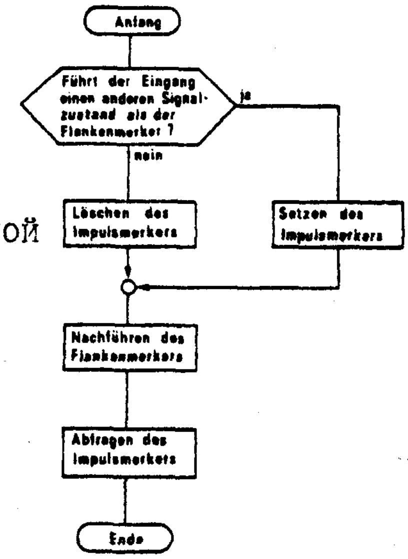
Сброс импульсной метки

Установка импульсной метки

Синхронизация метки фронта

Опрос импульсной метки

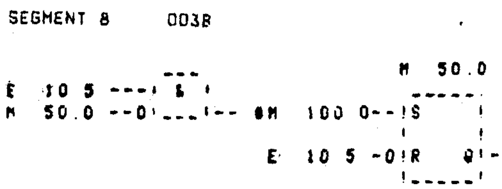
Конец



Обработка нарастающего фронта

Импульсный контакт при включении

Функциональная схема



Нарастание фронта имеет место, если на входе E 10.5 сигнал «1», а на метке фронта M 50.0 сигнал «0». В этом случае происходит установка метки импульса M 100.0. При установленной импульсной метке устанавливается также метка фронта M 50.0. При сигнале «0» на входе E 10.5 метка фронта M 50.0 снова сбрасывается.

Метка фронта, таким образом, все время приводится в соответствие. с состоянием сигнала на входе. Тем самым при следующем цикле обработки Функция И больше не выполняется. Поэтому метка импульса М 100.0 только в течение одного цикла имеет состояние сигнала «1».

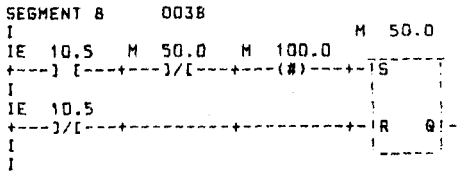
Таблица команд [1] /см.примечание на стр. 80/

```
SEGMENT 8
003B        :U   E 10.5
003C        :UN  M 50.0
003D        :=   M 100.0     Impulsmerker
003E        :U   M 100.0
003F        :S   M 50.0      Setzen des Flankenmerkers
0040        :UN  E 10.5
0041        :R   M 50.0      Rücksetzen des Flankenmerkers
0042        :NOP 0
0043        :***
```

Импульсная метка
установка метки фронта
сброс метки фронта

Фронт нарастает, если на входе Е 10.5 сигнал «1», а на метке фронта М 50.0 - сигнал «0». Тогда и на метке импульса М 100.0 будет сигнал «1». При установленной импульсной метке устанавливается также метки фронта М 50.0. При сигнале «0» на входе Е 10.5 метка Фронта М 50.0 снова сбрасывается.

Контактная схема

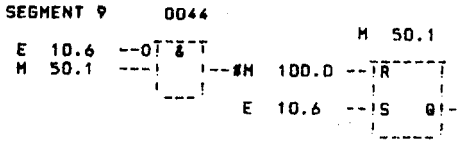


Если контакт Е 10.5 замкнут, а метка фронта М 50.0 имеет сигнал «0», происходит установка импульсной метки М 100.0 /импульсного контакта/. Одновременно устанавливается метка фронта М 50.0. Тем самым контакт М 50.0 снова размыкается, в результате чего импульсная метка только в течение одного цикла имеет состояние «1». При появлении на входе Е 10.5 сигнала «0» метка фронта М 50.0 снова сбрасывается.

Обработка падающего фронта

Импульсный контакт при отключении

Функциональная схема



Падение фронта имеет место, если на входе E 10.6 сигнал «0», а на метке фронта M 50.1 сигнал «1». Тогда происходит установка импульсной метки M 100.0. При установленной импульсной метке метка фронта M 50.1 сбрасывается. При сигнале «1» на входе E 10.6 метка фронта M 50.1 снова устанавливается.

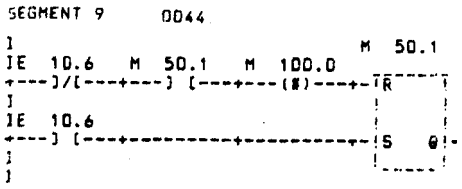
Таким образом, метка фронта всегда приводится в соответствие с состоянием сигнала на входе. Тем самим в следующем цикле обработки Функция И больше не выполняется. Поэтому импульсная метка только в течение одного цикла имеет состояние «1».

Таблица команд

```
SEGMENT 9
0044      :UN  E 10.6
0045      :U   M 50.1
0046      :=   M 100.0    Impulsmerker                    Импульсная метка
0047      :U   M 100.0
0048      :R   M 50.1     Rücksetzen des Flankenmerkers   Сброс метки фронта
0049      :U   E 10.6
004A      :S   M 50.1     Setzen des Flankenmerkers       Установка метки фронта
004B      :NOP 0
004C      :***
```

Фронт будет падающим, если на входе E 10.6 сигнал «0», а на метке фронта M 50.1 - сигнал "1". Тогда и на импульсной метке M 100.0 Будет сигнал «1». При установленной импульсной метке происходит сброс метки фронта M 50.1. При сигнале «1» на входе E 10.6 метка фронта M 50.1 снова устанавливается.

Контактная схема

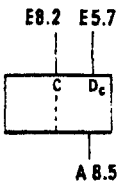


Если на входе E 10.6 сигнал «1», метка фронта M 50.1 установлена. Таким образом, контакт M 50.1 замкнут. Если теперь вход E 10.6 изменит свое состояние с «1» на «0», размыкающий контакт /НЗ/ E 10.6 замкнется и произойдет установка импульсной метки M 100.0 /импульсного контакта/. Одновременно метка фронта M 50.1 сбросится. Тем самым контакт M 50.1 снова "разомкнется" и, таким образом, импульсная метка M 100.0 только в течение одного цикла будет иметь состояние «1».

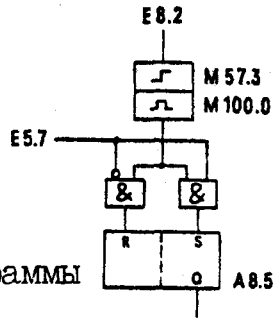
3.5 D- триггер

Элемент памяти D имеет два входа, один вход тактовых импульсов и один вход D. При смене сигнала на входе тактовых импульсов происходит запись состояния сигнала на входе D в элемент памяти.

Программирование охватывает обработку фронта и последующее программирование элемента памяти. Если на входе D сигнал «1», происходит установка памяти. Если на входе D сигнал «0», память сбрасывается.

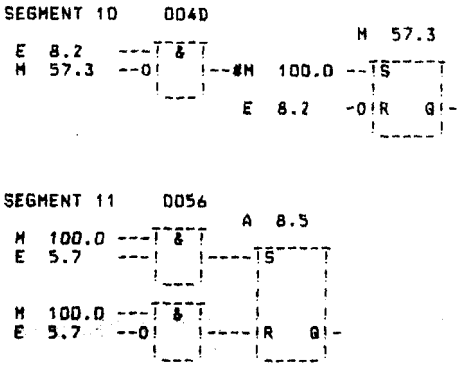


Схематическое изображение



Структура программы

Функциональная схема



Обработка фронта /нарастающего/ запрограммирована через метку фронта М 57.3. Импульсной меткой является метка М 100.0. Выход А 8.5 будет установлен в том случае, если есть фронт сигнала от входа Е 8.2, а на входе Е 5.7 - состояние «1». Если на входе Е 5.7 сигнал «0» при фронте от входа Е 8.2, выход А 8.5 будет сброшен.

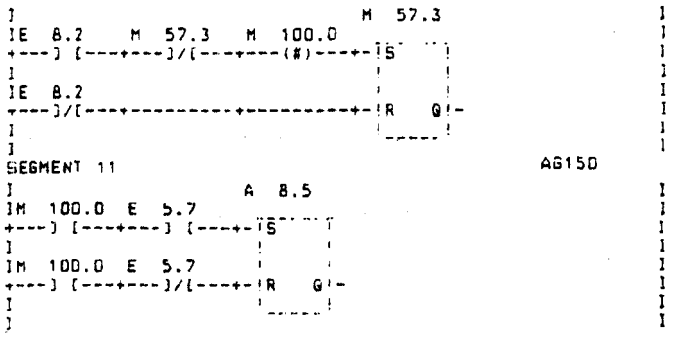
Таблица команд

```
SEGMENT 10
0040      :U   E 8.2      Flankenauswertung
004E      :UN  M 57.3
004F      :=   M 100.0        Impulsmerker
0050      :U   M 100.0
0051      :S   M 57.3         Setzen des Flankenmerkers
0052      :UN  E 8.2
0053      :R   M 57.3         Rücksetzen des Flankenmerkers
0054      :NOP 0
0055      :***

SEGMENT 11
0056      :U   M 100.0    Speicherglied
0057      :U   E 5.7
0058      :S   A 8.5          Setzen des Speichers
0059      :U   M 100.0
005A      :UN  E 5.7
005B      :R   A 8.5          Rücksetzen des Speichers
005C      :NOP 0
005D      :***
```

Обработка фронта
Импульсная метка
Установка метки фронта
Сброс метки фронта
Элемент памяти
Установка памяти
Сброс памяти

Обработка фронта /нарастающего/ запрограммирована через метку Фронта М 57.3. Импульсной меткой является метка М 100.0. Выход А 8.5 будет установлен в том случае, если есть фронт сигнала от входа Е 8.2, а на входе Е 5.7 - состояние «1». Если на входе Е 5.7 сигнал «0» при фронте от входа F 8.2, выход А 8.5 будет сброшен.

Контактная схема

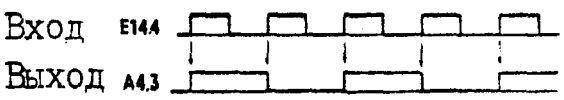


Обработка фронта /нарастающего/ запрограммирована через метку фронта М 57.3. Импульсным контактом является метка М 100.0. Выход А 8.5 будет установлен, если импульсный контакт М 100.0 замкнется, а на входе Е 5.7 будет сигнал «1» /НО-контакт Е 5.7 будет тогда замкнут/. Если на входе Е 5.7 сигнал «0» /НЗ-контакт Е 5.7 остается тогда замкнутым/, а импульсный контакт М 100.0 замкнут, выход А 8.5 сбрасывается.

3.6 Триггер со счетным входом /пересчетная схема/

Пересчетная логическая схема представляет собой функциональный элемент с одним входом и одним выходом. При изменении состояния сигнала на. входе, напр., с «0» на «1», изменится также состояние сигнала на выходе. Это /новое/ состояние будет теперь сохраняться до следующей, в нашем примере положительной, смены состояния. После этого состояние сигнала на выходе вновь изменится. Таким образом, на выходе этого функционального элемента появится только половина входной частоты.

Временная диаграмма импульсов



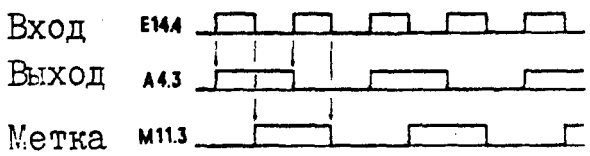
Существуют разные методы решения этой задачи. Ниже будут показаны 3 различных метода, в приложении к каждому способу изображения программы.

При изображении в виде функциональной и контактной схем одновременно приводится программирование в виде таблицы команд. Пересчетная логическая схема, в том виде, как она дана в "таблице команд", не поддается графическому представлению. Однако такое программирование значительно упрощает реализацию счетных функций /например, в устройствах автоматизации, которые не содержат счетчиков; см. раздел 10/.

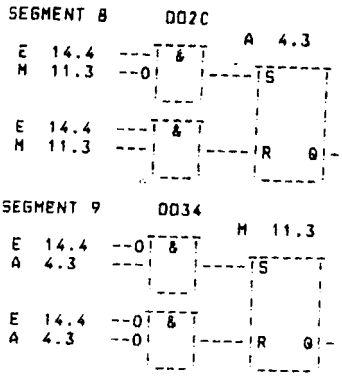
Функциональная схема

Если на входе E 14.4 сигнал «1», происходит установка выхода A 4.3. При изменении сигнала на входе E 14.4 с «1» на «0» /при установленном выходе A 4.3/ одновременно произойдет установка вспомогательной метки /M 11.3/. Если на входе E 14.4 вновь появится «1», то, так как метка M 11.3 установлена, произойдет сброс выхода A 4.3 Метка M 11.3 будет сброшена как только на входе E 14.4 появится «0» Теперь выход и метка занимают основное положение. При следующем изменении состояния сигнала на входе произойдет установка выхода и т.д.

Временная диаграмма импульсов



Программирование



Соответствующая программа в виде таблицы команд гласит:

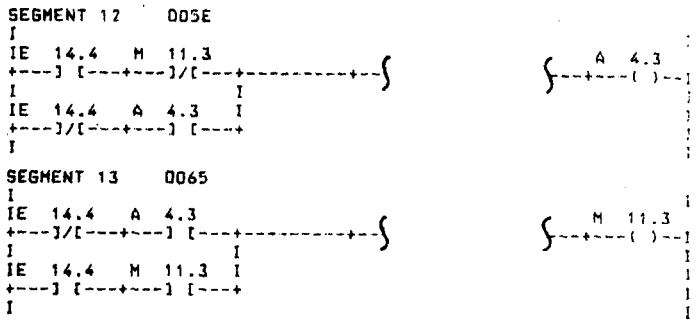
```
SEGMENT 8
002C        :U   E 14.4        Ausgang          Выход
002D        :UN  M 11.3
002E        :S   A 4.3               Setzen     Установить
002F        :U   E 14.4
0030        :U   M 11.3
0031        :R   A 4.3               Rücksetzen
0032        :NOP 0                               Сбросить
0033        :***                                 Вспомог.метка
SEGMENT 9                                        Установить
0034        :UN  E 14.4        Hilfsmerker
0035        :U   A 4.3
0036        :S   M 11.3              Setzen
0037        :UN  E 14.4
0038        :UN  A 4.3
0039        :R   M 11.3              Rücksetzen  Сбросить
003A        :NOP 0
003B        :***
```

Контактная схема

Принцип пересчетной логической схемы в этом примере реализован так же, как и в предыдущем случае, т.е. здесь тоже необходима метка /M 11.3/. И здесь справедлива та же составленная диаграмма в виде функциональной схемы.

В отличие от изображения в виде функциональной схемы здесь запоминающая функция выхода и метки реализуется, как это принято в принципиальных электрических схемах, за счет самоподхвата. Главная цепь содержит соответствующую Функцию установки, а параллельная цепь располагает контактом для самоподхвата.

Программирование



Соответствующая программа в виде таблицы команд гласит:

```
SEGMENT 12
005E      :U   E 14.4
005F      :UN  M 11.3
0060      :O
0061      :UN  E 14.4
0062      :U   A 4.3
0063      :=   A 4.3        Ausgang Выход
0064      :***

SEGMENT 13
0065      :UN  E 14.4
0066      :U   A 4.3
0067      :O
0068      :U   E 14.4
0069      :U   M 11.3
006A      :=   M 11.3        Hilfsmerker Вспомогательная метка
006B      :***
```

Таблица команд

Обработке подвергается нарастающий фронт входа E 14.4 /см.раздел Если выход сброшен, а импульсная метка имеет состояние «1», происходит установка выхода. Если выход был уже установлен, а метка имеет состояние «1», выход сбрасывается.

Программирование

```
SEGMENT 13
0088      :AWL
0089      :U    E 14.4      Flankenauswertung
008A      :UN   M 11.3
008B      :=    M 15.7            Impulsmerker
008C      :U    M 15.7
008D      :S    M 11.3            Setzen des Flankenmerkers
008E      :UN   E 14.4
008F      :R    M 11.3            Rücksetzen des Flankenmerkers
0090      :U    M 15.7      Ausgang
0091      :UN   A 4.3
0092      :S    A 4.3             Setzen
0093      :R    M 15.7            Rücksetzen des Impulsmerkers
0094      :U    M 15.7
0095      :U    A 4.3
0096      :R    A 4.3             Rücksetzen
0097      :***
```

Обработка фронта
Импульсная метка

Установка метки фронта

Сброс метки фронта

Выход
установка
сброс импульсной метки

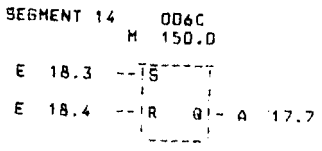
Сброс

если устанавливается выход А 4.3, следует сбросить импульсную метку, чтобы немедленно предотвратить сброс выхода А 4.3. В связи с этим сбрасыванием импульсной метки такая программа не поддается графическому изображению.

3.7 Энергонезависимые ОЗУ

В устройствах системы СИМАТИК S5 область меток имеет буферное питание. Благодаря этому метки сохраняют свое исходное состояние при отключении сети не менее 4 недель. Выходы же, напротив, при восстановлении питания от сети сбрасываются. Если хотят, чтобы выходы сохраняли свое состояние сигналы, которые должны быть
на выходах после восстановления напряжения, отображают в виде меток. В этом случае эти выходы примут состояние сигналов меток.

Функциональная схема



Метка М 150.0 устанавливается через вход Е 18.3 и сбрасывается через вход Е 18.4. Состояние сигнала метки присваивается выходу А 17.7.

Таблица команд

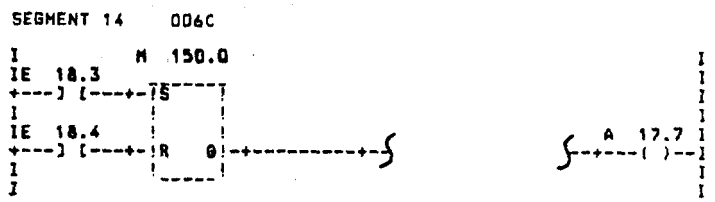
```
SEGMENT 14
006C      :U   E 18.3      Setzen des Hilfsmerkers
006D      :S   M 150.0
006E      :U   E 18.4
006F      :R   M 150.0     Rücksetzen des Hilfsmerkers
0070      :U   M 150.0     Abfrage des Hilfsmerkers
0071      :=   A 17.7      Nachführen des Ausgangs
0072      :***
```

Установка вспомогат.метки

Сброс вспомогательной метки
Опрос вспомогательной метки
Присвоение выходу состояния вспомогательной метки

Метка M 150.0 устанавливается через вход E 18.3 и снова сбрасывается через вход E 18.4. Состояние сигнала метки присваивается выходу A 17.7.

Контактная схема



Метка M 150.0 устанавливается через вход E 18.3 и снова сбрасывается через вход E 18.4. Состояние сигнала метки присваивается выходу A 17 ..7.

3.8 Установка входов

Функции памяти могут быть также применены и для области операндов входов. Однако это возможно только при условии, что в распоряжении имеется отображение процесса на входах. Отображение процесса представляет собой одну из зон памяти данных системы. Всякий раз в начале циклической обработки программы в эту зону памяти загружаются состояния сигналов входных блоков. После этого процессор работает только с этой зоной памяти. Таким отображением процесса для входов располагают только устройства автоматизации S5-110S, S5-130W, S5-150A, S5-150K и S5-150S.

Установка и сброс входов, а также присвоение результата логического сопряжения здесь происходит так же, как это делается с выходами или метками. Как и при установке выходов, здесь тоже должны присутствовать соответствующие блоки ввода, чтобы процессор не выдал сигнала об "ошибке в программировании".

Функциональная схема

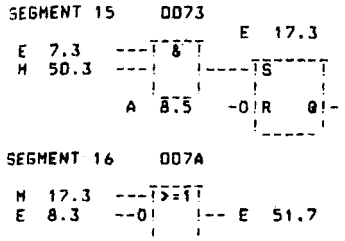


Таблица команд

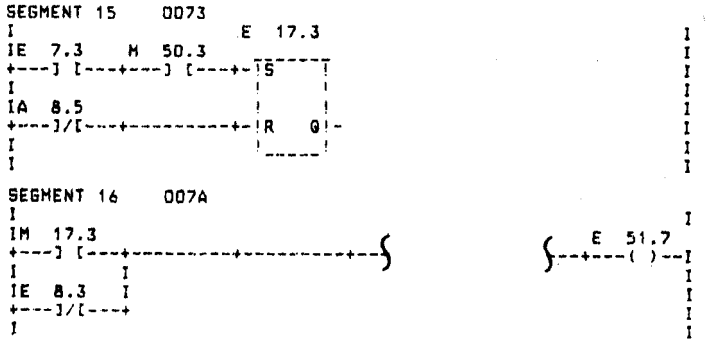
```
SEGMENT 15
0073      :U   E 7.3        Setzen des Eingangs
0074      :U   M 50.3
0075      :S   E 17.3
0076      :UN  A 8.5
0077      :R   E 17.3       Rücksetzen des Eingangs
0078      :NOP 0
0079      :***
SEGMENT 16
007A      :O   M 17.3
007B      :ON  E 8.3
007C      :=   E 51.7       Zuweisung
007D      :***
```

Установка входа

Сброс входа

Присвоение

Контактная схема



К установке входов прибегают при пуско-наладочных работах, когда необходимо создать имитацию входных сигналов. В качестве первого вызова блока в организационном блоке 0В 1 /см.раздел 9/ пишут вызов блока программы.
В этом блоке программы производится установка или сброс входов. Тогда последующая программа работает с этими измененными /имитируемыми/ параметрами входов.

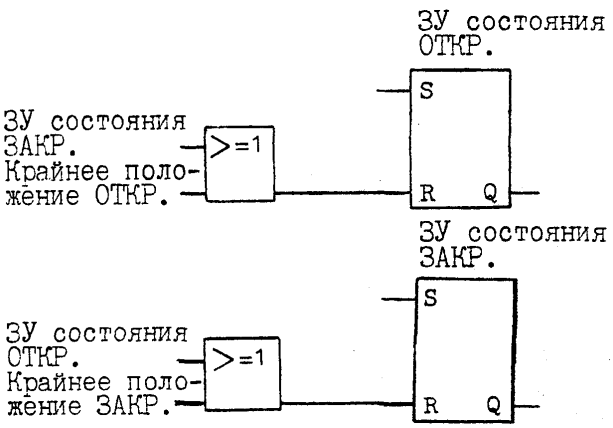
3.9 Пример программирования "Управление задвижкой"

Рассмотрим случай управления задвижкой. Задвижка может управляться по 3 положениям: "ОТКР.", "ЗАКР." и "СТОП". Один выход устройства автоматизации управляет задвижкой в направлении "ОТКР.", второй - в направлении "ЗАКР." Если не один из выходов не имеет сигнала «1», задвижка пребывает в неподвижном состоянии /"СТОП"/. Сигнал «1» всегда может находиться только на одном выходе /единичное звено с управлением по 3 точкам/.
Как только задвижка достигла одного из конечных положений /"ОТКР." или "ЗАКР."/ , происходит отключение соответствующего выхода. Тогда управлять задвижкой можно только через другой выход.
Управление задвижкой может производиться в двух режимах. В ручном режиме рабочее состояние задвижки задается через 3 кнопки /"ОТКР.", "ЗАКР." и "СТОП"/. В автоматическом режиме рабочим состоянием задвижки управляют 3 метки /устанавливаемые, к примеру, в циклических управляющих устройствах/.
Разрешающий вход через состояние «1» деблокирует выходы. Только тогда на них может появиться «1». Если на разрешающем входе «0», то и на выходах будет «0».

Функциональная схема таблица команд

В рассматриваемом примере задвижка управляется через 2 выхода. На обоих выходах одновременно допускается состояние «0» и не допускается «1». Выходы являются взаимно блокируемыми. Если один из выходов имеет состояние «1», то на входе сброса другого выхода тоже появляется «1». При приоритете сброса последний выход будет иметь состояние «0» до тех пор, пока первый сохраняет установленное состояние.

Состояние «0» будет на выходе также и в том случае, если задвижка находится в соответствующем крайнем положении. Это крайнее положение регистрируется датчиком, также связанным с входом сброса.

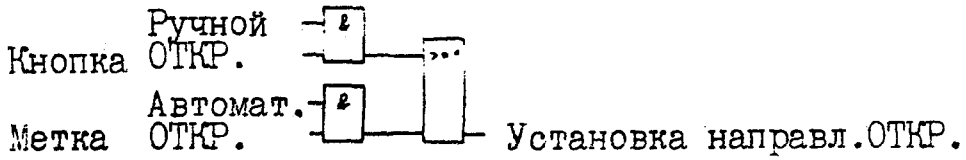


Задвижка должна управляться в двух режимах - ручном и автоматическом. Рассмотрим сначала, как будет протекать управление в направлении ОТКР. Задвижка открывается, если :

- включен ручной режим и нажата кнопка ОТКР.

  или

- включен автоматический режим и метка ОТКР. имеет сигнал «1».

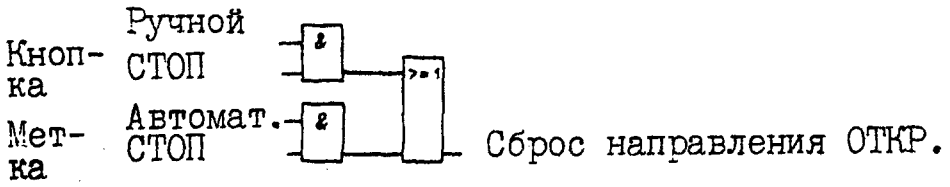


Направление ОТКР. будет сброшено, если:

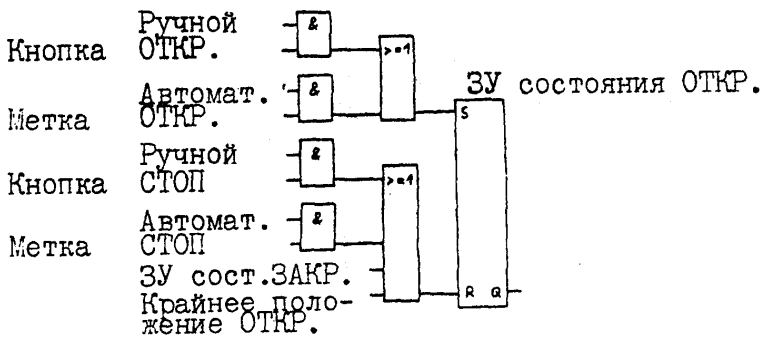
- включен ручной режим и нажата кнопка СТОП

  или

- включен автоматический режим и метка СТОП имеет сигнал «1».



Последние два сопряжения ставятся теперь перед соответствующими входами ЗУ состояния ОТКРЫТО, причем операция ИЛИ сразу же увязывается с входом сброса.



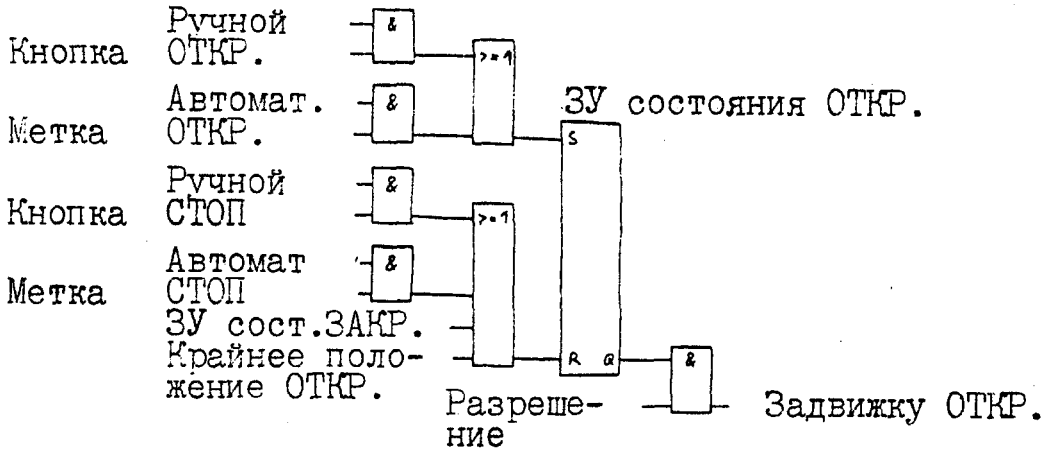
Теперь в качестве последней функции остается учесть еще вход разрешения. Выход ОТКР. должен иметь состояние "1", если:

- ЗУ состояния ОТКР. имеет сигнал «1»

и

- если вход разрешения имеет сигнал «1».

Таким образом, после выхода ЗУ состояния ОТКР. должна быть установлена функция И, на которую выходит второй вход деблокирующего входа. Тогда выход Функции И будет соответствовать выходу на задвижку.



Функция "Задвижку ЗАКРЫТЬ" готовится по образу и подобию функции "Задвижку ОТКРЫТЬ". В сочетании со следующим ниже списком присвоений мы получаем полную функциональную схему этого примера.

Список присвоений

E 48.0	кнопка ОТКР.
E 48.1	кнопка ЗАКР.
E 48.2	кнопка СТОП
E 48.6	вход разрешения
E 51.6	подтверждение о достижении положения ОТКР.
E 52.6	подтверждение о достижении положения ЗАКР.

M 27.3 Метка ОТКР.

M 27.4 Метка ЗАКР.

M 27.5 Метка СТОП

M 30.0 Режим ручной

M 30.2 Режим автоматический

M 43.5 ЗУ состояния ОТКР.

M 43.6 ЗУ состояния ЗАКР.

A 16.0 Задвижку ОТКР.

A 16.1 Задвижку ЗАКР.

Функциональная схема

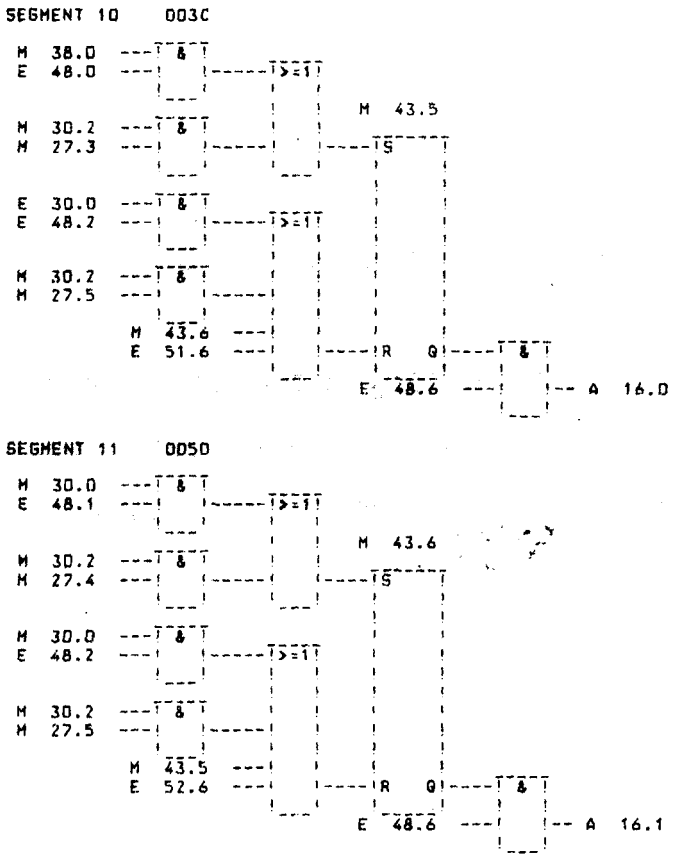


Таблица команд

```
SEGMENT 12
0064      :AWL
0065      :U    M 30.0
0066      :U    E 48.0
0067      :O
0068      :U    M 30.2
0069      :U    M 27.3
006A      :S    M 43.5
006B      :U    M 30.0
006C      :U    E 48.2
006D      :O
006E      :U    M 30.2
006F      :U    M 27.5
0070      :O    M 43.6
0071      :O    E 51.6
0072      :R    M 43.5
0073      :U    M 43.5
0074      :U    E 48.6
0075      :=    A 16.0
0076      :U    M 30.0
0077      :U    E 48.1
0078      :O
0079      :U    M 30.2
007A      :U    M 27.4
007B      :S    M 43.6
007C      :U    M 30.0
007D      :U    E 48.2
007E      :O
007F      :U    M 30.0
0080      :U    M 27.5
0081      :O    M 43.5
0082      :O    E 52.6
0083      :R    M 43.6
0084      :U    M 43.6
0085      :U    E 48.6
0086      :=    A 16.1
0087      :***
```

Контактная схема

Данный пример - в отличие от изображения в виде функциональной схемы - нужно запрограммировать так, как если бы это делалось обычным способом на контакторах.

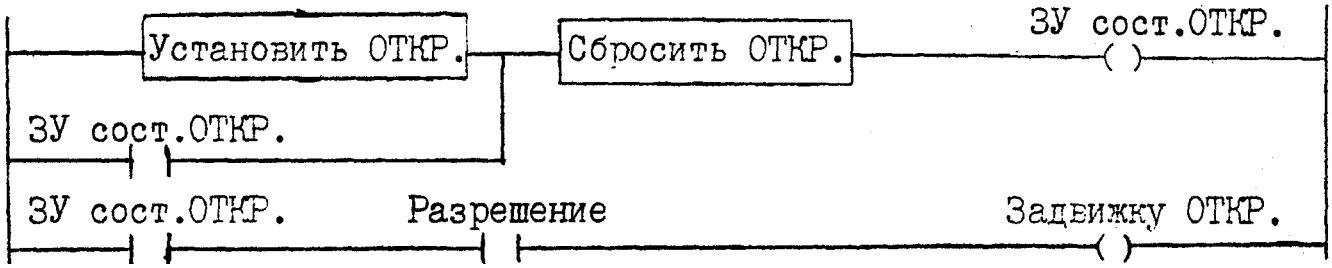
Имеется задвижка, которая управляется через два выхода. Электрические схемы обоих выходов идентичны, благодаря чему достаточно запрограммировать пепь для одного выхода /например, выхода, управляющего направлением ОТКР./

Схема другого выхода /управляющего направлением ЗАКР./ будет соответственно одинаковой.

Выход, управляющий направлением ОТКР., должен иметь память с приоритетом сброса. Кроме того, должна быть возможность отключения

выходного сигнала /"разрешение"/ без сброса памяти в направлении ОТКР. Таким образом, сначала должно программироваться запоминающее устройство. Затем контакт ЗУ включается последовательно с деблокирующим контактом и выводится на выход "задвижку ОТКР."

Схема программирования будет тогда выглядеть следующим образом:



Теперь запрограммируем схему для установки ЗУ состояния ОТКР.

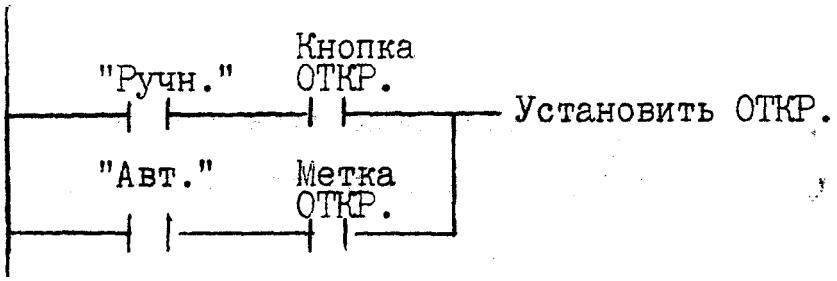
ЗУ будет установлено, если

- в ручном режиме нажата кнопка ОТКР.

или

- в автоматическом режиме метка ОТКР. имеет состояние «1».

Это соображение ведет к следующему программированию:



При сбросе ЗУ направления ЮТКР. следует учитывать, что - в отличие от изображения в виде функциональной схемы - логическое сопряжение должно давать состояние «0», если хотят, чтобы оно было действенным. ЗУ направления ОТКР. не может быть установлено, если установлено ЗУ направления ЗАКР. и если достигнуто крайнее положение ОТКРЫТО. В обоих случаях тогда на соответствующих контактах будет сигнал «1»;

цепь в обоих случаях должна быть прервана, т.е. необходимо последовательно включить два размыкающих /НЗ/ контакта.

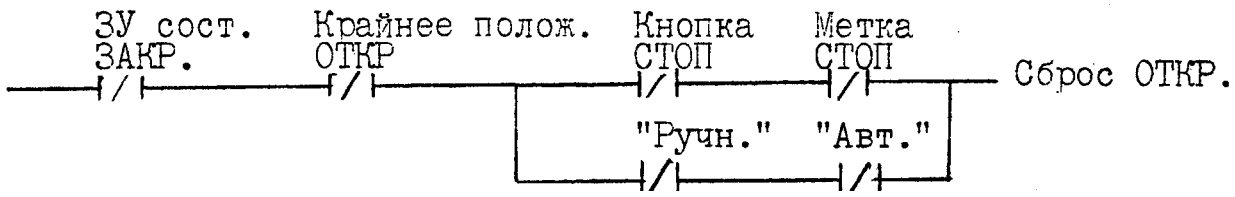
Цепь должна также прерываться, если

- в ручном режиме нажата, кнопка СТОП

или

- в автоматическом режиме метка СТОП имеет состояние «1».

Кнопка. СТОП при нажатии должна прерывать цепь /состояние «1»/. Таким образом, она программируется как НЗ контакт. Однако действовать она должна только при условии, что ручной режим имеет состояние «1». Если ручной режим имеет состояние «0», кнопка СТОП действовать не может. Эта Функция реализуется за счет параллельного включения к кнопке СТОП одного НЗ контакта "ручной режим". Соответственно параллельно к НЗ контакту метки СТОП подключается НЗ контакт автоматического режима. Таким образом, для реализации условия сброса ЗУ состояния ОТКР. мы получаем следующее логическое сопряжение:



Эти оба последних сопряжения можно теперь внести в первоначально составленную схему программы. По такому же образцу составляется цепь для направления ЗАКР. В сочетании с нижеследующим списком присвоений получают полную программу для данного примера.

Список присвоений

E 48.0 Кнопка ОТКР.

E 48.1 Кнопка ЗАКР.

E 48.2 Кнопка СТОП

E 48.6 Вход разрешения

E 51.6 Подтверждение достижения крайнего положения ОТКР.

E 52.6 Подтверждение достижения крайнего положения ЗАКР.

M 27.3 Метка ОТКР.

M 27.4 Метка ЗАКР.

M 27.5 Метка СТОП

M 30.0 Ручной режим

M 30.2 Автоматический режим

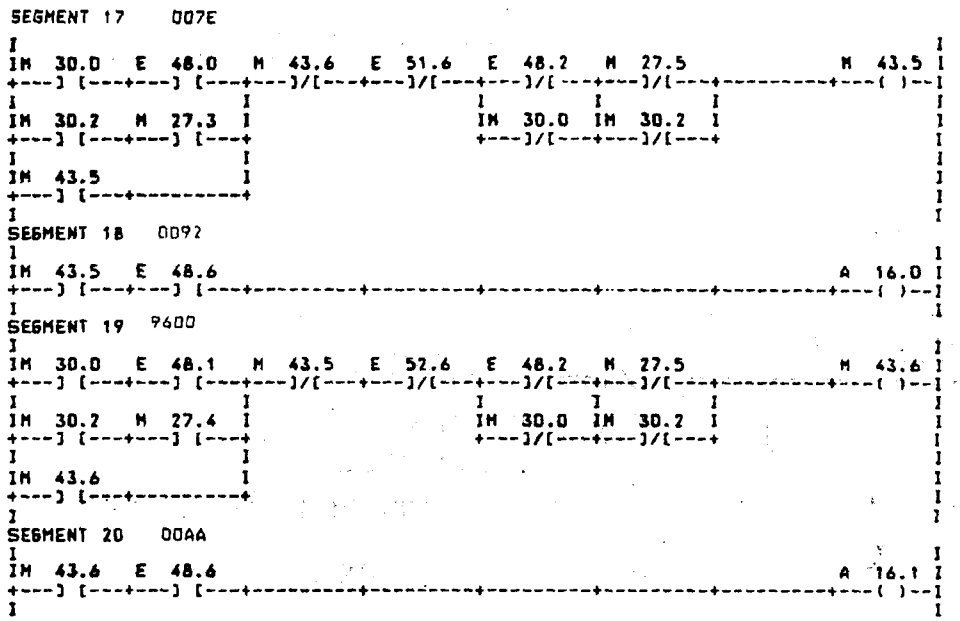
M 43.5 ЗУ направления ОТКР.

M 43.6 ЗУ направления ЗАКР.

A 16.0 Задвижку ОТКР.

A 16.1 Задвижку ЗАКР.

Контактная Схема



4 ФУНКЦИИ ВРЕМЕНИ

Функции времени в системе автоматизации SIMATIC S5 реализуются двумя различными способами. Они или выступают как специальные блоки периферии, или образуют область операндов в памяти центрального устройства.

В устройствах автоматизации S5-110A и S5-130A используют специальные блоки периферии, причем в ПК S5-110A для запуска и спроса Функций времени обращаются к каналам ввода/вывода. Хотя в этих аппаратах запуск времени происходит программным путем, величина периода регулируется на блоке времени, например, с помощью потенциометров. Объем Функций для временных процессов, предлагаемый языком программирования STEP5 может быть полностью использован устройствами автоматизации S5-110A и S5-130A /см. раздел 10 "Программирование устройств автоматизации S5-110A " и раздел II "Программирование устройства автоматизации S5-130A"/.

Функции времени, реализуемые устройствами автоматизации S5-110S, S5-130W, S5-130A, S5-150K и S5-150S, являются областями операндов в памяти центрального блока. Время в этой области операндов представляет собой 16-разрядное слово. Это 16-разрядное слово включает в себя биты состояний, дискретность, а также параметр времени. Биты состояний необходимы процессору для обработки Функций времени. Дискретность указывает на интервалы, с которыми параметр времени уменьшается на одну единицу. Параметр времени указывает на количество истекших единиц. Для запуска отсчета времени, реализуемого математическим путем, помимо операции пуска дополнительно нужно указать продолжительность /см.раздел 4.1/. Поведение функции времени при сбросе описывается в разделе 4.2. В последующих разделах раскрывается опрос временной Функции и различные программируемые характеристики времени. В разделе 4.9 показано программирование контроля частоты. Опрос параметров времени /в цифровом виде/, загрузка параметра времени описываются в разделе 6. В одной логической операции или в одной электрической цепи можно изображать только одну функцию времени. Не

допускается ее программирование в сочетании с функциями RS-памяти, счетчиками и компараторами.

4.1 Запуск таймера

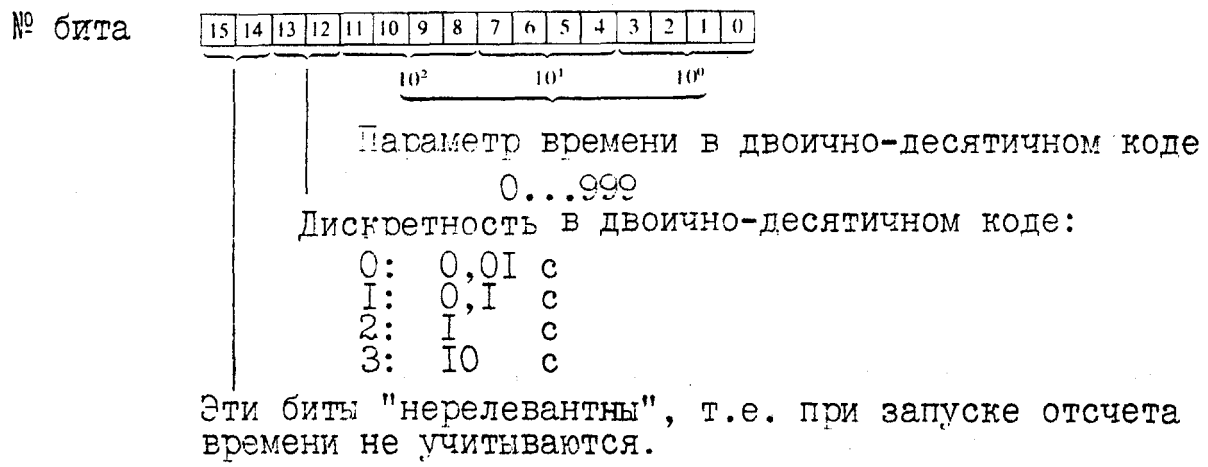
Отсчет времени считается запущенным, если на пусковом входе /при изображении программы в. виде Функциональной или контактной схемы/ или перед операцией пуска /при изображении в виде таблицы команд/ результат логического сопряжения изменится с «0» на «1». Для запуска отсчета времени всегда необходима такая смена состояния сигналов.

Любую уставку времени можно запустить по одной из пяти следующих характеристик:

- в виде короткого импульса
- в виде продленного импульса
- в виде задержки включения
- в виде задержки включения с запоминанием
- в виде задержки отключения.

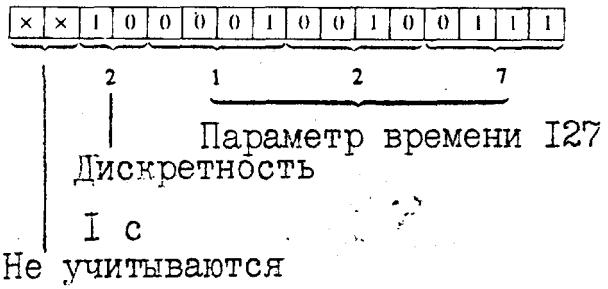
Ввод уставок времени

При запуске таймера в качестве уставки времени принимается значение, стоящее в аккумуляторе. Как и когда оно загружается в аккумулятор, роли не играет /см.также раздел 6.1 - "Загрузка"/. В качестве уставки времени из аккумулятора берутся 16 разрядов, стоящих справа.



Пример:

необходимо задать уставку времени 127 с. Распределение битов:



При задании уставки времени в виде константы вслед за указанием параметра времени через точку дается- дискретность-./см.раздел 6.1 "Загрузка"/.

Заданное в десятичном виде значение времени преобразуется в соответствующее двоичное слово времени и в таком виде обрабатывается. Параметр времени в двоичном коде /биты 0...9/входит в слово времени. Остальные биты являются битами состояний, которые нужны процессору для распознания и обработки временной характеристики, а также битами, указывающими на дискретность бремени.

В соответствии с заданной дискретностью после запуска параметр времени уменьшается на одну единицу пока не достигнет нулевого значения.

4.2 Сброс таймера

Установка времени оказывается сброшенной, 'если на входе сброса /на функциональной или контактной схеме/ или перед операцией сброса /в таблице команд/ появляется логическая «1». До тех пор пока сохраняется этот результат, опросы времени на/состояние «1» будут давать результат «0», а опросы на состояние «0» -результат «1».

При сбросе уставки времени обработка /отсчет/ времени заканчивается. Параметр времени устанавливается на ноль. Если сброс уставки времени должен действовать "статически" и независимо от результата. На входе запуска времени, нужно, чтобы /в таблице команд/ сброс уставки программировался сразу после установки времени и еще до его опроса.

Пример:

Последовательность программирования уставок времени /таймеров/

Запуск отсчета времени

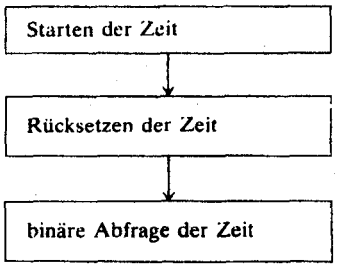
Сброс уставки времени

Двоичный опрос времени

Пример:

Beispiel:

Programmierfolge bei Zeiten



Beispiel:

```
U   E  17.3
L   EW 20
SI  T   5

U   E  18.0
R   T   5

U   T   5
=   A  20.1
```

4.3 Опрос таймера

Установку времени можно опрашивать точно также, как например, опрашивают вход, и использовать результат опроса в последующих логических операциях. В зависимости от характеристики функции времени опрос на состояние «1» дает различные варианты временных процессов /см. последующие разделы/. Опрос на состояние «0» дает, также, как например, опрос входов, прямо противоположный результат, чем при опросе на состояние «1».

4.4 Запуск установки времени /таймера/ в режиме короткого импульса

Функциональная схема

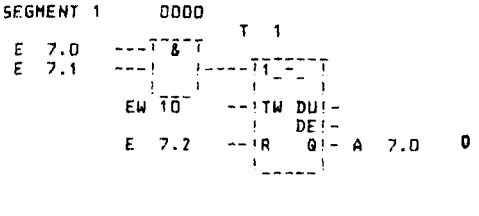
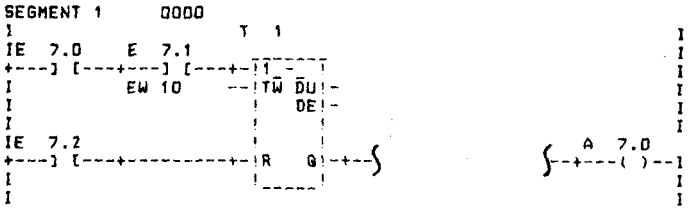


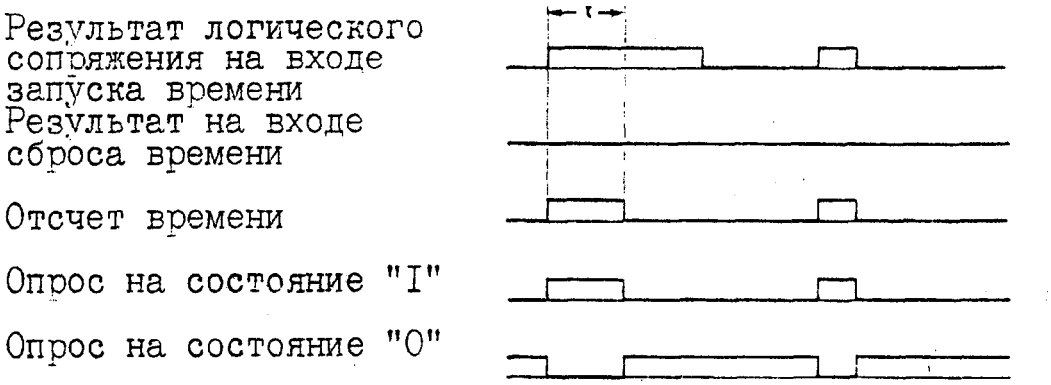
Таблица команд [1]

```
SEGMENT 1
0000      :U   E 7.0
0001      :U   E 7.1
0002      :L   EW10
0003      :SI  T 1          Starten als Impuls
0004      :U   E 7.2
0005      :R   T 1          Rücksetzen
0006      :NOP 0
0007      :NOP 0
0008      :U   T 1          Abfragen
0009      :=   A 7.0
000A      :***
```

Контактная схема



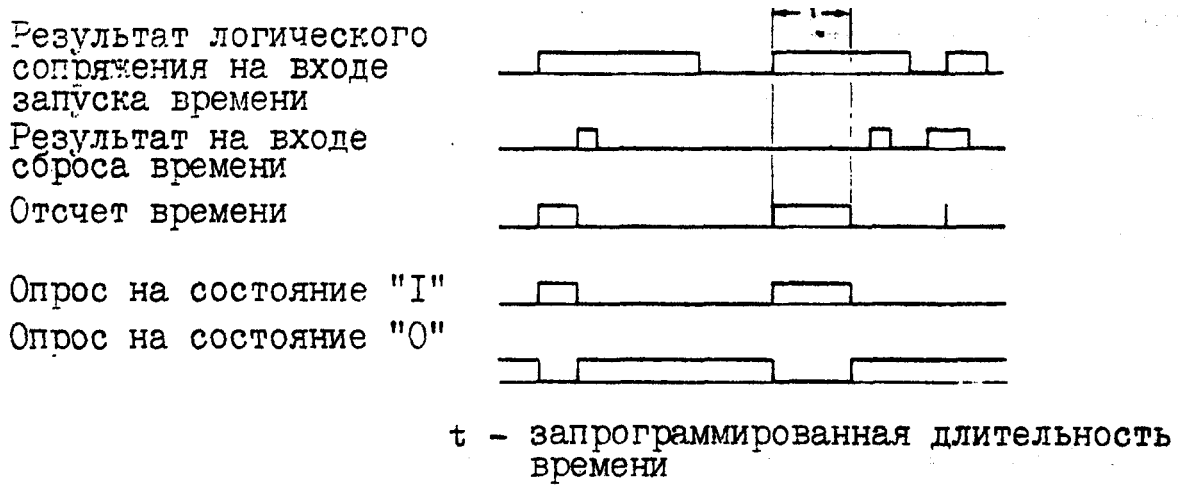
Временная диаграмма короткого импульса



t-запрограммированная длительность времени

Запуск таймера происходит при смене результата логического сопряжения на входе запуска времени с «0» на «1». Отсчет времени заканчивается с истечением запрограммированного периода. При логическом результате «0» происходит сброс таймера. До тех пор, пока длится отсчет запрограммированного времени, опросы на состояние «1» будут давать результат «1».

Временная диаграмма при сбросе таймера



Сброс таймера происходит при появлении на входе сброса времени логической «1». При смене логического результата на входе сброса с «1» на «0», тогда как на входе запуска «1», время воздействию не подвергается.

Если при поступлении сигнала сброса результат логического сопряжения на входе запуска сменится с «0» на «1», таймер хотя и будет запущен , однако благодаря запрограммированному впоследствии сбросу окажется немедленно сброшенным /что обозначено на диаграмме вертикальной черточкой/. Учитывая последовательность программирования, описанную в разделе 4.2 это обстоятельство не скажется на двоичном опросе времени.

4.5 Запуск таймера, запрограммированного в режиме удлиненного импульса

Функциональная схема

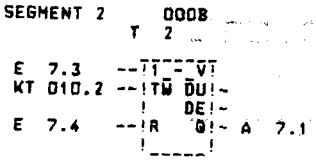
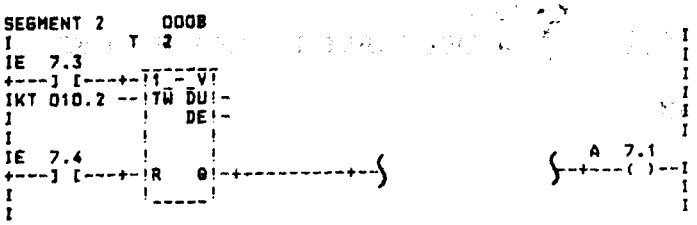


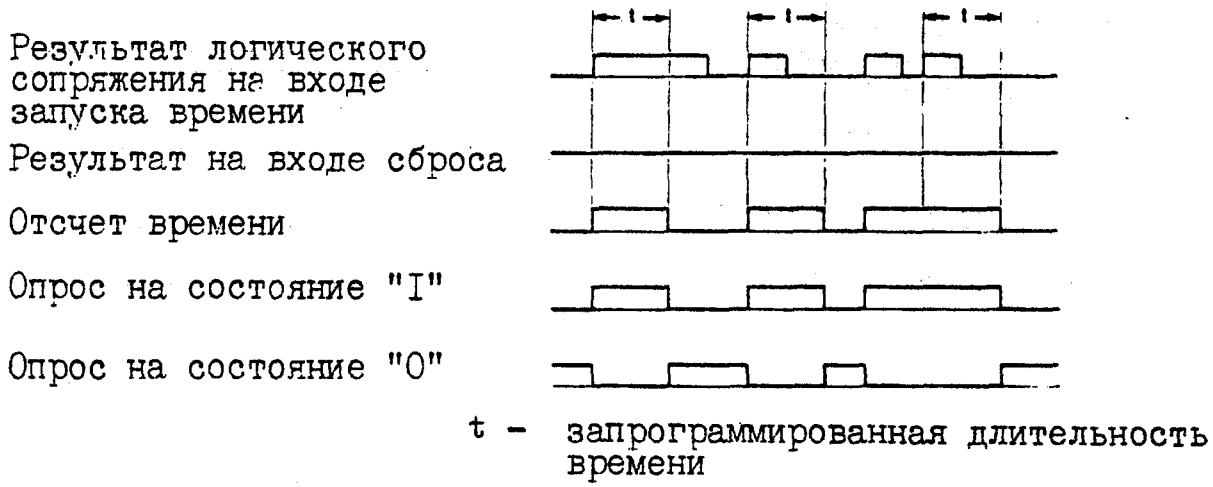
Таблица команд

```
SEGMENT 2
0008      :U   E 7.3
0009      :L   KT010.2
000E      :SV  T 2        Starten als verlängerter Impuls
000F      :U   E 7.4
0010      :R   T 2        Rücksetzen
0011      :NOP 0
0012      :NOP 0
0013      :U   T 2        Abfragen
0014      :=   A 7.1
0015      :***
```

Контактная схема



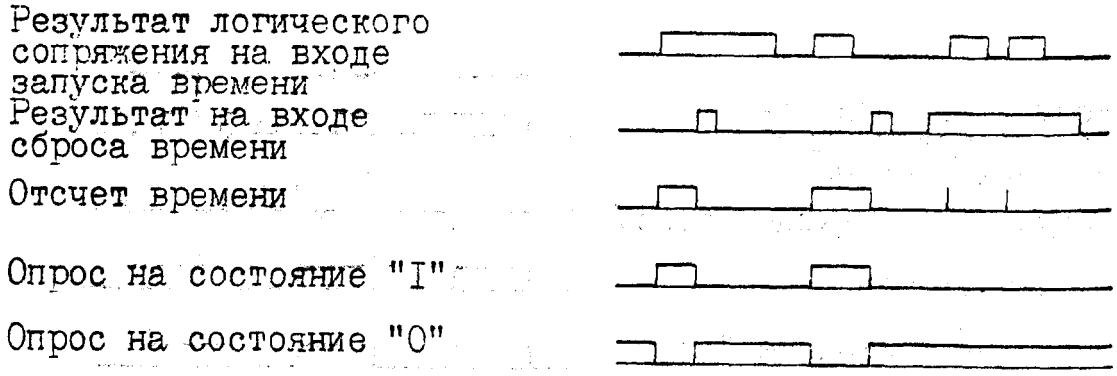
Временная диаграмма удлиненного импульса



Запуск таймера происходит при смене результата логического сопряжения на входе запуска времени с «0» на «1». Отсчет времени заканчивается с истечением запрограммированного периода независимо от последующего логического результата на входе запуска. Если изменение состояния сигнала с «0» на «1» произошло до истечения заданного времени, оно запускается вновь с запрограммированным значением /"пост-триггер"/.

Пока будет длиться отсчет запрограммированного времени, опросы на состояние «1» будут давать результат «1».

Временная диаграмма при сбросе



Сброс таймера происходит при появлении на входе сброса логического результата «1». Если результат логического сопряжения на входе сброса меняется с «1» на «0», тогда как на входе установки сигнал,." I:", воздействия на - время: не происходит.

Если при поступлении сигнала сброса результат логического сопряжения на входе запуска сменится с «0» на «1», таймер хотя и будет запущен , однако благодаря запрограммированному впоследствии сбросу окажется немедленно сброшенным /что обозначено на диаграмме вертикальной черточкой/. Учитывая последовательность программирования, описанную в разделе 4.2, это обстоятельство не скажется на двоичном опросе времени.

4.6 Запуск таймера запрограммированного в режиме задержки включения

Функциональная схема

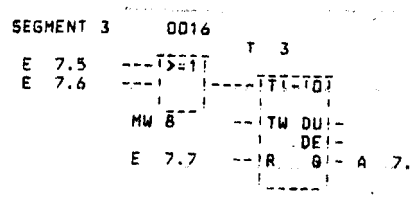
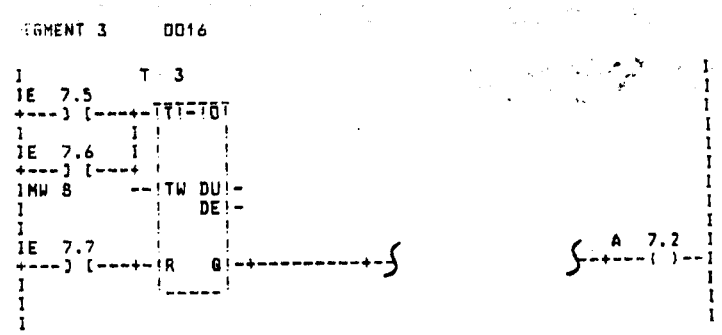


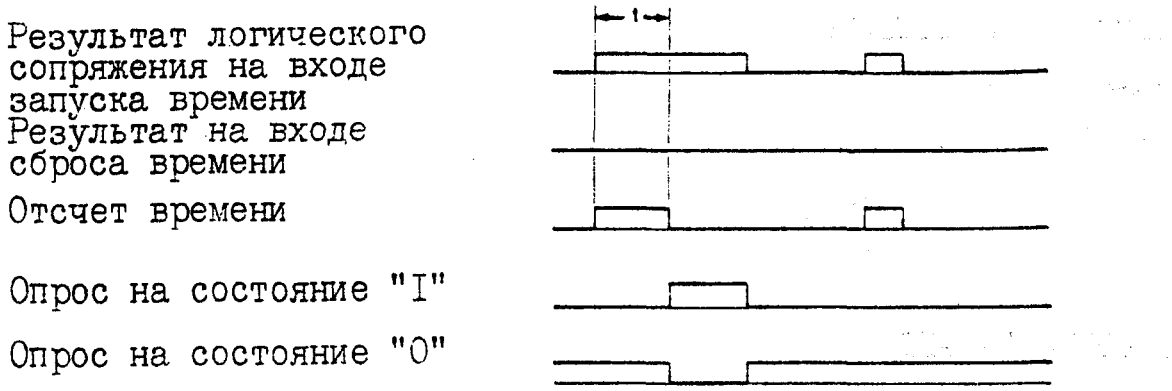
Таблица команд

```
SEGMENT 3
0016      :O   E 7.5
0017      :O   E 7.6
0018      :L   MW8
0019      :SE  T 3          Starten als Einschaltverzögerung
001A      :U   E 7.7
001B      :R   T 3          Rücksetzen
001C      :NOP 0
001D      :NOP 0
001E      :U   T 3          Abfragen
001F      :=   A 7.2
0020      :***
```

Контактная схема



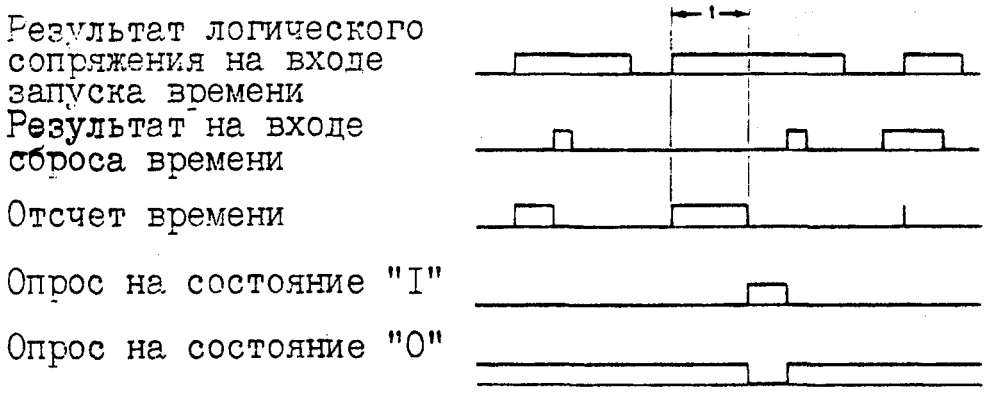
Временная диаграмма задержки включения



t - запрограммированная длительность времени

Запуск таймера происходит при смене сигнала на входе запуска с «0» на «1». Отсчет заканчивается с истечением запрограммированного периода. При логическом результате «0» происходит сброс таймера. Опросы на состояние сигнала «1» дают результат «1» до тех пор, пока время не истекло, а на входе запуска сигнал «1». Однако, если на входе сброса логический результат «1», результат опроса даст "0"см.ниже/.

Временная диаграмма при сбросе



t - запрограммированная длительность времени

Сброс таймера происходит при появлении на входе сброса логического результата «1». Если логический результат на входе сброса меняет свое состояние с"1" на «0», тогда как на входе запуска сигнал «1», воздействия на время не происходит. Если при поступлении сигнала сброса результат логического сопряжения на. входе запуска сменится с «0» на «1», таймер хотя и будет

запущен , однако благодаря запрограммированному впоследствии

сбросу окажется немедленно сброшенным /что обозначено на диаграмм вертикальной черточкой/. Учитывая последовательность программирования, описанную в разделе 4.2, это обстоятельство не скажется на двоичном опросе времени.

4.7 Запуск таймера запрограмированого в режиме задержки включения с запоминанием

Функциональная схема

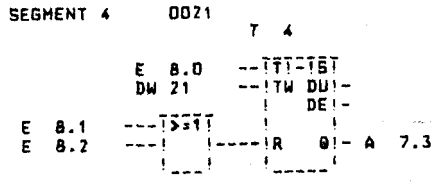
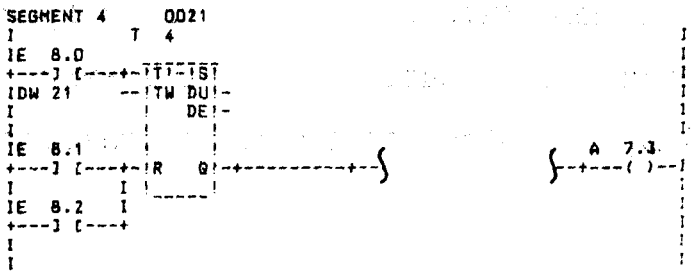


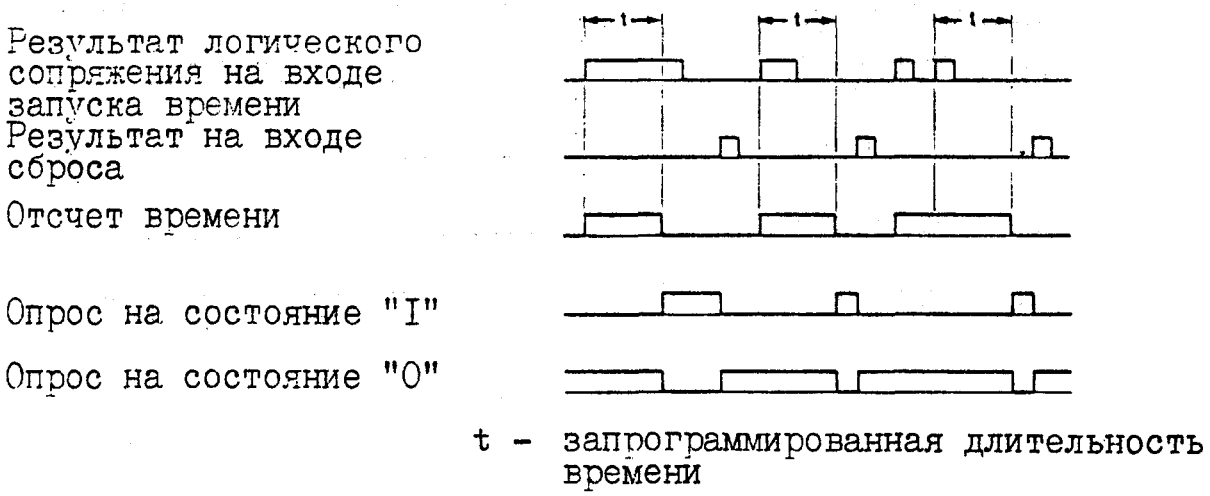
Таблица команд

```
SEGMENT 4
0021      :U   E 8.0
0022      :L   DW21
0023      :SS  T 4          Starten als speichernde Einschaltverzögerung
0024      :O   E 8.1
0025      :O   E 8.2
0026      :R   T 4          Rücksetzen
0027      :NOP 0
0028      :NOP 0
0029      :U   T 4          Abfragen
002A      :=   A 7.3
002B      :***
```

Контактная схема

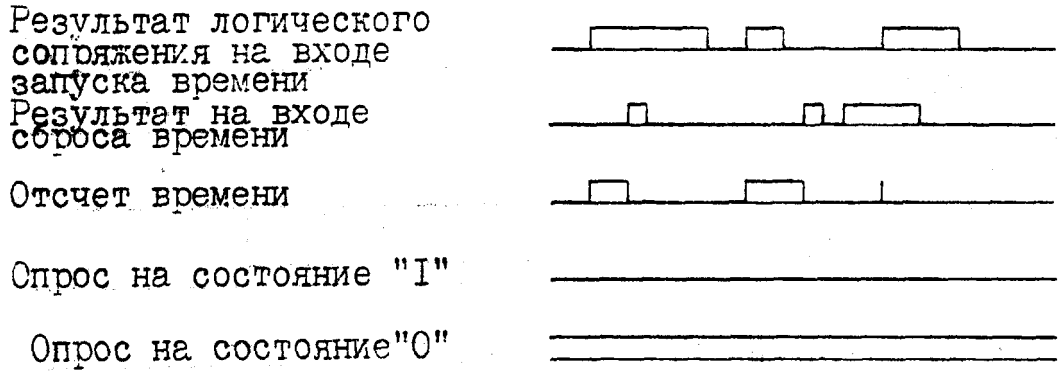


Временная диаграмма задержки включения с запоминанием



Запуск таймера происходит при смене сигнала на входе запуска с «0» на «1». Отсчет заканчивается с истечением запрограммированного периода независимо от последующего результата на входе запуска. Опросы на состояние сигнала дадут результат «1» при условии истечения времени. Эта «1» так и останется. Она сменится на «0» только после появления на входе сброса сигнала «1».

Временная диаграмма при сбросе



Сброс таймера происходит при появлении на входе сброса логического результата «1». Изменение логического результата на входе сброса с «1» на «0», тогда как на входе запуска будет сигнал «1», на времени не сказывается.

Если при поступлений сигнала: сброса результат логического сопряжения на входе запуска сменится с «0» на «1», таймер хотя и будет запущен, однако благодаря запрограммированному впоследствии сбросу окажется немедленно сброшенным /что обозначено на диаграмме вертикальной черточкой/. Учитывая последовательность программирования, описанную в разделе 4.2, это обстоятельство не скажется на двоичном опросе времени.

4.8 Запуск таймера запрограммированного в режиме задержки отключения

Функциональная схема

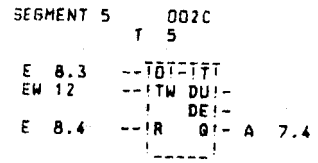
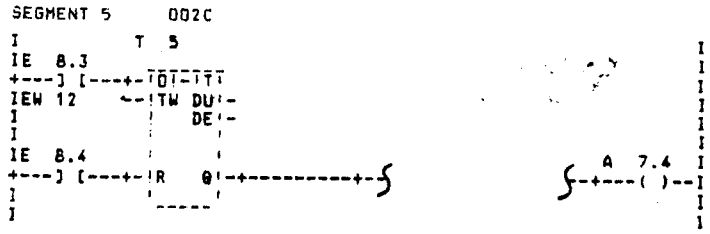


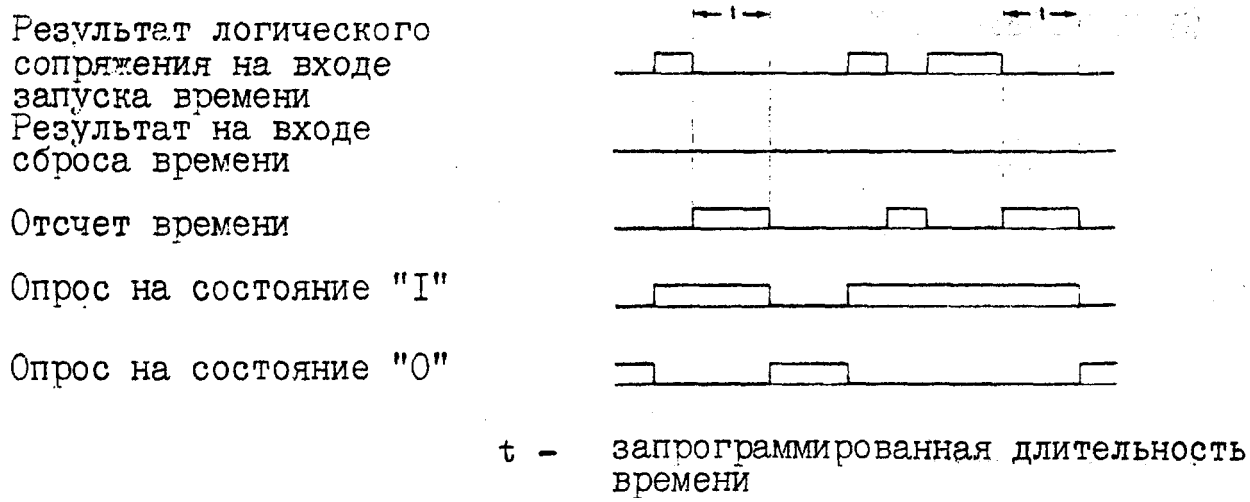
Таблица команд

```
SEGMENT 5
002C      :U   E 8.3          Запуск задержки отключения
002D      :L   EW12
002E      :SA  T 5     Starten als Ausschaltverzögerung
002F      :U   E 8.4
0030      :R   T 5 Rücksetzen Сброс
0031      :NOP 0
0032      :NOP 0
0033      :U   T 5 Abfragen
0034      :=   A 7.4          Опрос
0035      :***
```

Контактная схема

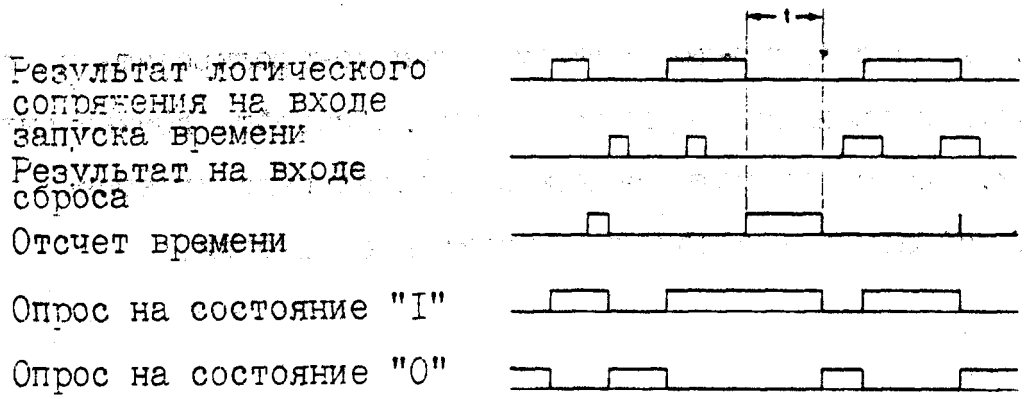


Временная диаграмма задержки отключения



Запуск таймера происходит при смене сигнала на входе запуска с «1» на «0». Отсчет заканчивается с истечением запрограммированного периода. При появлении результата «1» происходит сброс таймера. Опросы на состояние «1» дают результат «1», если на входе запуска В результат "1" или если время еще не истекло.

Временная диаграмма при сбросе



t - запрограммированная длительность времени

Таймер сбрасывается, если до истечения времени на входе сброса будет результат «1». Тогда при опросе на состояние «1» в результате появится «0». Смена результата на входе сброса с «1» на «0» на время не влияет.

Появление на входе сброса «1» без отсчета времени для таймера значения не имеет.

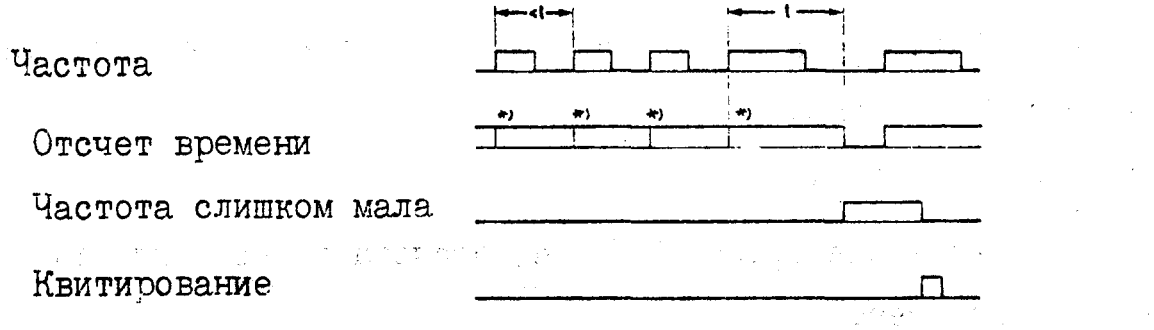
Если при поступлении сигнала сброса - результат логического сопряжения сменится с «1» на. «0», таймер хотя избудет запушен , однако благодаря,. запрограммированному впоследствии сбросу окажется немедленно сброшенным /что обозначено на диаграмме вертикальной черточкой/. Опрос на состояние «1» даст тогда результат «0».

4.9 Пример программирования "Контроль частоты"

Предположим, что какую-то частоту нужно контролировать по определенной нижней границе. Функция контроля частоты должна быть отключаемой. Выход частоты за нижний предел индицируется. Индикацию можно сбрасывать с помощью квитирующей кнопки.

Решение

Любое нарастание фронта сигнала частоты вызывает запуск - таймера. Если время истекает, значит пауза между двумя фронтами сигнала слишком велика, т.е. частота слишком мала.



Для выполнения требуемого условия необходимо произвести запуск тай мера в режиме удлиненного импульса.

Любое нарастание Фронта сигнала вызывает запуск таймера, при условии, что нет сигнала "Контроль ОТКЛ." Поэтому перед входом запуска оба сигнала сопрягаются по функции И, т.е. оба контакта включаются последовательно.

При истечении запрограммированного времени происходит установка памяти. Для этого время опрашивается на состояние «0». Память остается установленной, даже если следующий фронт сигнала вызовет повторный запуск таймера.

Память сбрасывается нажатием на квитирующую кнопку или через функцию "Контроль ОТКЛ.". В обоих случаях индикации не происходит, так как в Функции памяти приоритет отдан сбросу.

Функциональная схема

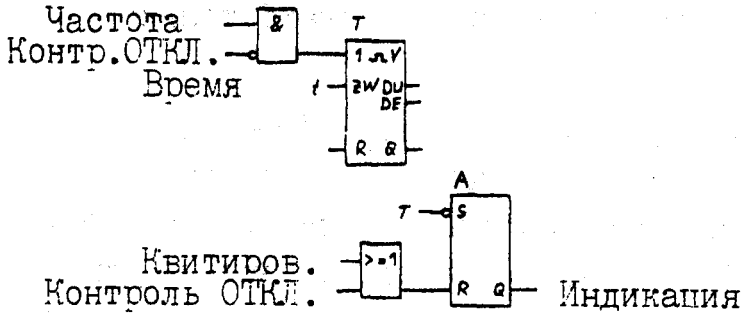
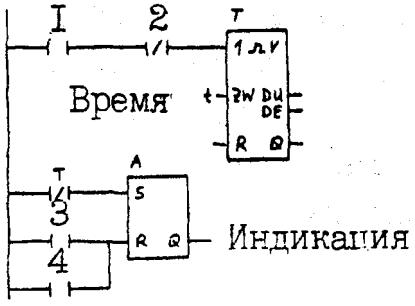


Таблица команд

U E	Частота
UN E	Контроль ОТКЛ.
SV T	Запуск таймера в режиме удлиненного импульса
K	Параметр времени
UN T	Время истекло?
S A	Установка индикации
O E	Квитирование
O E	Контроль ОТКЛ.
R A	Сброс индикации

Контактная схема



1 - частота

2 - контроль ОТКЛ.

3 - квитирование

4 - контроль ОТКЛ.

5 ФУНКЦИИ СЧЕТА

Язык программирования STEP 5 допускает прямую реализацию счетчиков. Эти счетчики располагаются или на специальном блоке, или образуют область операндов в памяти центрального устройства.

Счетчики устройства автоматизации ПК S5-I30A находятся в специальном блоке - блоке времени счетчиков. Параметр счета устанавливается на этом блоке. Такие счетчики допускают только обратный порядок счета.

Счетчики устройств автоматизации S5-110S,S5-130K, S5-150A,S5-150K и S5-150S являются областями операндов в памяти центрального устройства. В этой области операндов счетчик представляет собой 16-разрядное слово. В этом 16-разрядном слове размещены биты состояний и параметр счета. Биты состояний нужны процессору для обработки счетчика. Параметр счета является собственно "содержимым" счетчика и соответствует состоянию счетчика.

Эти счетчики устанавливаются на какую-то числовую величину /см. раздел 5.1/ и сбрасываются в нулевое положение /см.раздел 5.2/. Они используются как счетчики прямого и обратного счета, в зависимости от того, увеличивается ли числовое значение на единицу /см.раздел 5.3/ или уменьшается /см.раздел 5.4/. Кроме того, при опросе счетчика можно получить двоичный результат опроса, имеет ли счетчик величину больше нуля, или равную нулю /раздел 5.5/.

Опрос /цифровой/ параметра счета, загрузка параметра описываются в разделе 6.

На каждое логическое сопряжение соотв. электрическую цепь, можно изобразить только одну функцию счета. Ее программирование в сочетании с функциями RS -памяти, установками времени и компараторами не допускается.

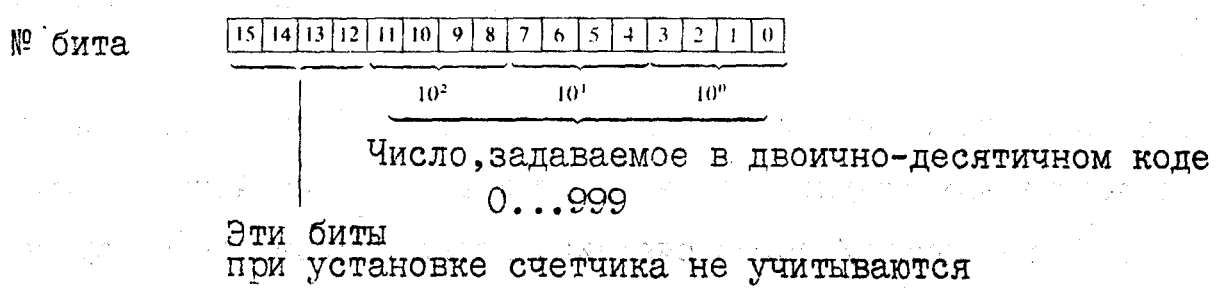
5.1 Установка счетчика

Установка счетчика происходит при смене на входе установки /при изображении в виде функциональной или контактной схемы/ или перед операцией установки /при изображении в виде таблицы команд/ логического результата с «0» на «1».

Такая смена состояния всегда является непременным условием установки счетчика.

Ввод числовых уставок

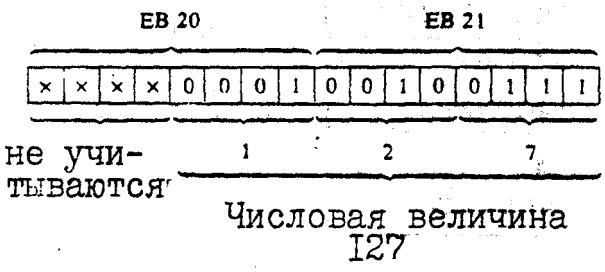
При установке счетчика в качестве уставки берется значение, стоящее в аккумуляторе. Как и когда оно загружается в аккумулятор, значения не имеет /см.также раздел 6.1 - "Загрузка"/. В качестве параметра счета из аккумулятора берутся 16 разрядов, стоящих справа. Они распределяются следующим образом:



Пример:

требуется задать числовую величину 127

Распределение битов:



Числовая величина, задаваемая в десятичном виде, переносится в соответствующее слово в двоичном виде и там обрабатывается. Числовая величина занимает в слове числа разряды с 0 по 9 в двоичном коде. Остальные 6 битов являются разрядами состояний, которые требуются в процессоре для обработки счетчика.

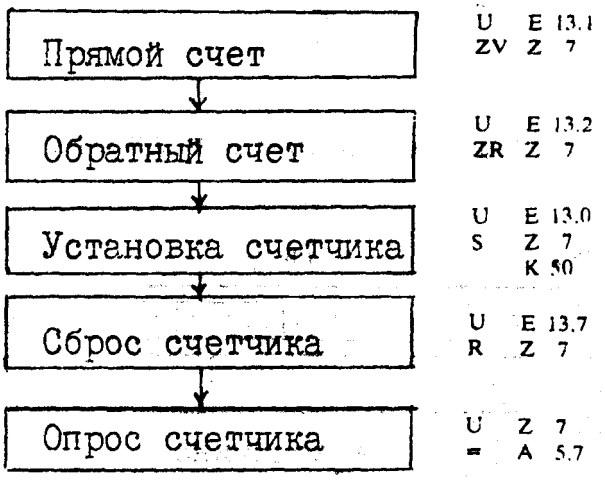
5.2 Сброс счетчика

Счетчик будет сброшен, если на входе сброса /при изображении программы в виде функциональной или контактной схемы/ или перед операцией сброса /в таблице команд/ появляется логическая «1». До тех пор, пока сохраняется этот результат, опросы счетчика на состояние «1» будут давать результат «0», а опросы на состояние «0» - результат «1». При сбросе счетчика числовая величина устанавливается на "ноль" /стирается/.

Для того, чтобы сброс счетчика действовал "статически" и не зависел от логических результатов на других входах счетчика, нужно, чтобы /в таблице команд/ сброс счетчика программировался сразу же за установкой прямого или обратного счета, но еще до опроса.

ПРИМЕР

Последовательность программирования счетчиков



5.3 Прямой счет

Счетчик будет производить прямой отсчет, если на входе прямого счета /при изображении программы в виде функциональной или контактной схемы/ или перед операцией прямого счета /в таблице команд/ произойдет смена состояния сигнала с «0» на «1». Такая смена всегда необходима для ведения счетчиком прямого счета. При каждой смене сигнала на входе прямого счета числовая величина увеличивается на одну единицу. Как только она достигает верхней границы 999, повышение прекращается. После этого смена состояния сигнала на входе прямого счета никакого воздействия не оказывает. Переноса не происходит.

5.4 0братный счет

Счетчик будет считать в обратном порядке, если на входе обратного счета /при изображении программы в виде функциональной или контактной схемы/ или перед операцией обратного счета /в таблице команд/ произойдет смена состояния сигнала с «0» на «1». Такая смена всегда необходима для ведения счетчиком обратного счета.
При каждой смене сигнала на входе обратного счета числовая величина
уменьшается на одну единицу. Как только она достигает нижней границы 0, уменьшение прекращается. После этого смена состояния сигнала на входе обратного счета никакого воздействия не оказывает. Операций с отрицательными числами не производится.

5.5 Опрос счетчика

Счетчик может опрашиваться на своем двоичном выходе /на функциональных и контактных схемах/ или через операции И и ИЛИ, или И-НЕ и ИЛИ-НЕ /в таблице команд/ на значение числа большее или равное нулю. Опрос на состояние «1» дает результат «1», если состояние счетчика больше нуля, и дает результат «0», если состояние счетчика равно нулю.

5.6 Представление счётчика

Функциональная схема

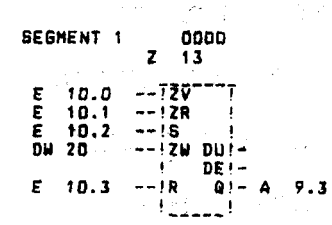
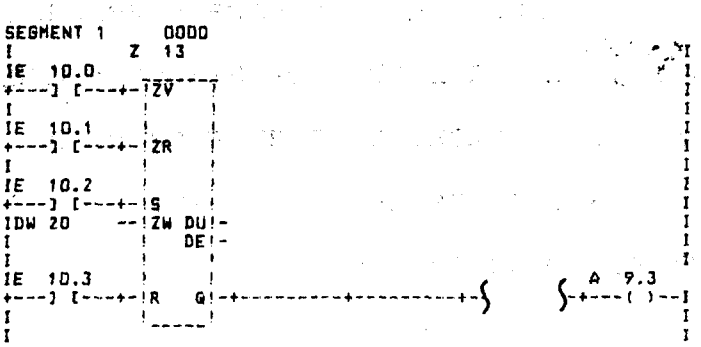


Таблица команд

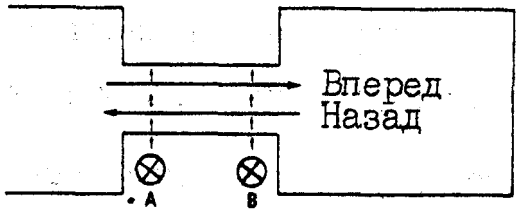
```
SEGMENT 1
0000      :U   E 10.0
0001      :ZV  Z 13        Zählen vorwärts
0002      :U   E 10.1
0003      :ZR  Z 13        Zählen rückwärts
0004      :U   E 10.2
0005      :L   DW20
0006      :S   Z 13        Setzen
0007      :U   E 10.3
0008      :R   Z 13        Rücksetzen
0009      :NOP 0
000A      :NOP 0
000B      :U   Z 13        Abfragen
000C      :=   A 9.3
000D      :***
```

Контактная схема



5.7 Пример программирования "Подсчёт посетителей"

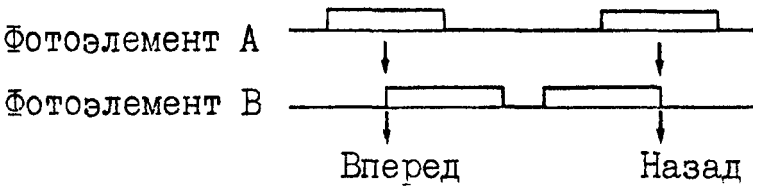
В этом примере нам нужно запрограммировать счетчик прямого и обратного счета. Требуется подсчитать количество людей в помещении. Для этой цели вход оборудуется 2 Фотоэлементами, которые установлены таким образом, что при пересечении этих фотоэлементов прерывается сначала один, а потом оба Фотоэлемента. Из этого образуется сигнал счета. Приводимая ниже схема показывает направление счета.



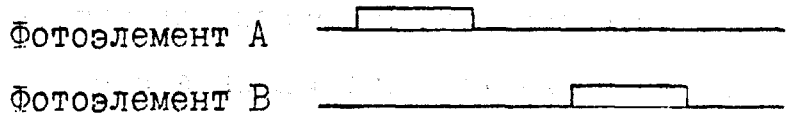
Фотоэлементы выдают сигнал «1», если луч света прерван. Нажатием кнопки можно произвести предварительную установку счетчика. Для этого имеются 2 цифровых задатчика двоично-десятичного кода и кнопка. Кроме того, с помощью цифрового индикатора /2 декады в двоично-десятичном коде/ можно показать количество людей, находящихся в помещении. Световой сигнал будет указывать на то, что помещение занято.

Решение

Направление счета можно определить по временной последовательности перекрытия фотоэлементов. Если первым перекрывается фотоэлемент А, значит счет ведется в прямом направлении. При обратном счете первым будет перекрыт фотоэлемент В.

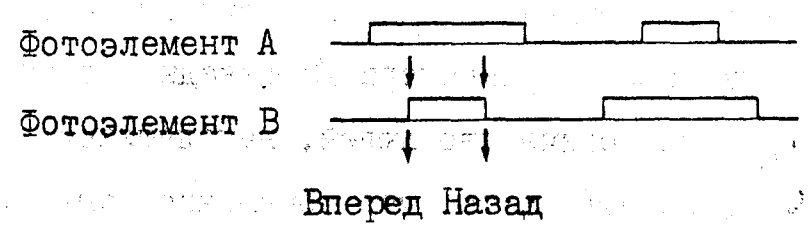


Предпосылкой счета является прерывание минимум одного Фотоэлемента /в нашем примере Фотоэлемента А/, т.е. наличие сигнала «1». При смене сигнала на. фотоэлементе В с «0» на «1» счет пойдет в прямом направлении. При смене состояния сигнала с «1» на «0» счет пойдет в обратном направлении.



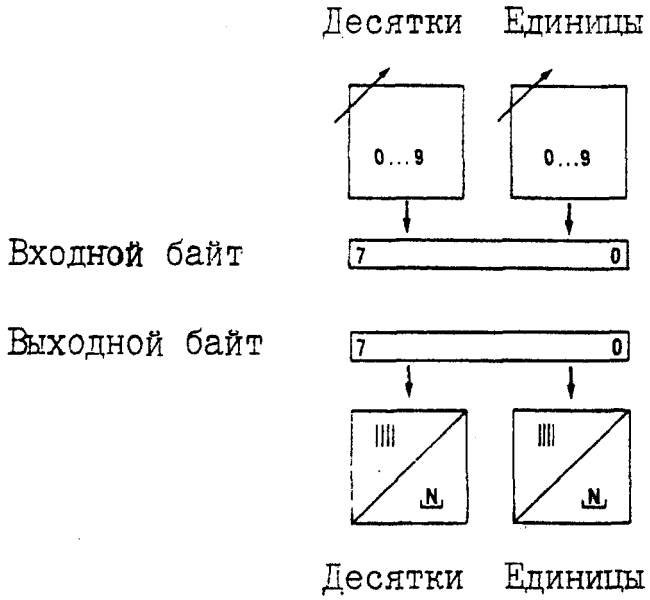
Условие для операции счета /"фотоэлемент А имеет сигнал «1», а "на фотоэлементе В наличие фронта"/ здесь не выполнено.

Показанное также на индикаторе количество посетителей не должно изменяться, если кто-либо приблизится к входу настолько, что и второй фотоэлемент окажется перекрытым, а потом опять двинется в направлении, откуда пришел.



При подходе кого-либо снаружи и при прерывании фотоэлемента В состояние счетчика увеличивается на 1. После этого состояние счетчика снова уменьшается на 1, и таким образом количество людей по-прежнему будет показано правильно. При приближении кого-либо к входу изнутри состояние счетчика не изменяется.

Оба цифровых задатчика подключены к одному входному байту, а цифровая индикация - к одному выходному байту:



Кодирование входной величины для установки счетчика автоматически перенимается от устройства автоматизации /ПК/. Для индикации числовая величина загружается в кодированном виде /см.раздел 6.3/.

Список присвоений

E	15.0	Фотоэлемент A
E	16.0	фотоэлемент B
E	18.3	Предварительная установка счетчика
EB	19	Предварительно установленная величина
A	22.2	Индикация "Помещение занято"
AB	23	Индикация состояния счетчика
M	58.0	Метка фронта нарастания
M	56.1	Метка фронта падения
M	100.0	Метка импульса фронта нарастания
M	100.1	Метка импульса фронта падения
Z	12	Счетчик

Функциональная схема

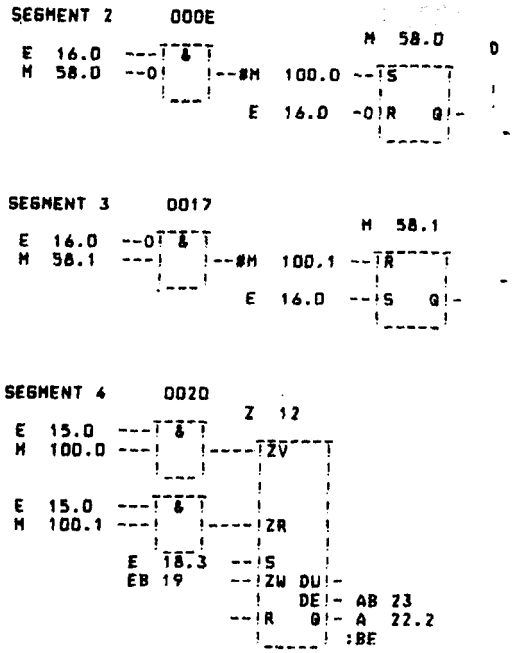
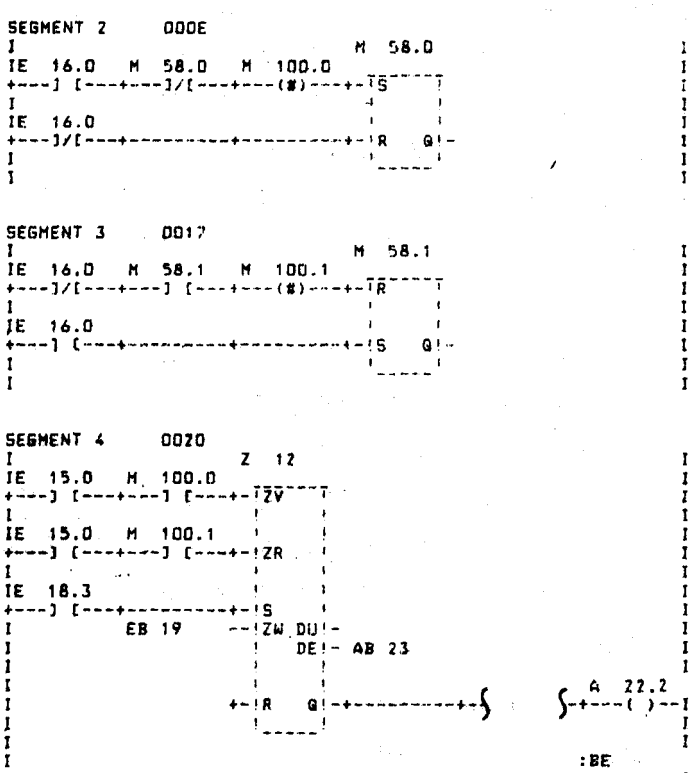


Таблица команд

```
SEGMENT 2
000E      :U   E 16.0    Обработка нарастающего фронта
000F      :UN  M 58.0
0010      :=   M 100.0   Импульсная метка
0011      :U   M 100.0
0012      :S   M 58.0    Установка метки фронта
0013      :UN  E 16.0
0014      :R   M 58.0    Сброс метки фронта
0015      :NOP 0
0016      :***

SEGMENT 3
0017      :UN  E 16.0    Обработка падающего фронта
0018      :U   M 58.1
0019      :=   M 100.1   Импульсная метка
001A      :U   M 100.1
001B      :R   M 58.1    Сброс метки фронта
001C      :U   E 16.0
001D      :S   M 58.1    Установка метки фронта
001E      :NOP 0
001F      :***

SEGMENT 4
0020      :U   E 15.0
0021      :U   M 100.0   Прямой счет
0022      :ZV  Z 12
0023      :U   E 15.0
0024      :U   M 100.1
0025      :ZR  Z 12      Обратный счет
0026      :U   E 18.3
0027      :L   EB19
0028      :S   Z 12      Установка
0029      :NOP 0
002A      :NOP 0         Загрузка числовой величины в кодированном виде
002B      :LC  Z 12      Индикация числовой величины
002C      :T   AB23      Опрос на"состояние счетчика >0"
002D      :U   Z 12      Индикация "Кто-то находится в помещении"
002E      :=   A 22.2
002F      :BE
```

Контактная схема



6. ФУНКЦИИ ЗАГРУЗКИ И ПЕРЕНОСА

Благодаря функциям загрузки и переноса язык программирования STEP 5 позволяет производить обмен информацией между блоками ввода/вывода, отображением процесса на входах и выходах и накопителем меток и данных, а также обрабатывать параметры времени и счета. Этот обмен информацией идет не напрямую, а в "обход" через аккумулятор. Этот аккумулятор представляет собой особый регистр процессора и служит в качестве "промежуточного накопителя".

При обмене информацией указывают направление потока информации. Поток информации от запоминающего устройства в аккумулятор называется загрузкой /аккумулятор "заряжается", см.раздел 6.1/. Поток информации от аккумулятора к запоминающему устройству называется переносом /содержимое аккумулятора" переносится" в ЗУ, см. раздел 6.2/. При загрузке и переносе содержимое областей операндов обрабатывается байтами, словами или двойными словами.

Обработка областей операндов времени и счетчиком отличается от обработки остальных областей операндов. Первые могут комбинироваться только с одной Функцией загрузки, однако путем кодового преобразования допускают также загрузку в аккумулятор /см.разделы 6.3 и 6.4/.

В устройствах автоматизации S5-110S, S5-130W, S5-150A, S5-150K и S5-150S загрузка, и перенос производятся независимо от результата логической операции: Только в устройстве S5-130A эти операции зависят от логического результата. Загрузка или перенос в ПК S5-130A производится только при условии, если результат логической операции равен «1» /см.раздел 11.6/. Операции загрузки и переноса изображаются только в виде таблицы команд. Исключение составляют операции загрузки в сочетании с областями операндов времени и счетчиков /см.раздел 6.3 и 6.4/.

6.1 Загрузка

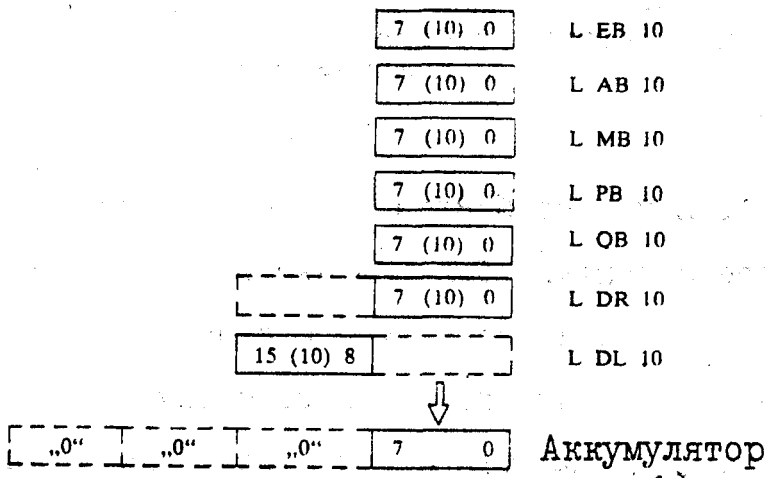
С помощью операции "загрузка" информацию из областей операндов E A, м, т, Z ив, а также константы и информацию из. блоков ввода/ вывода можно непосредственно загружать в аккумулятор. Загрузка из областей операндов T и Z описывается в разделах 6.3 и 6.4.

Информация областей операндов E, A, M и D загружается в аккумулятор в виде байтов, слов или двойных слов, а информация непосредственно с блоков ввода/вывода - как байты или олова.

Ширина аккумулятора в устройстве S5-130A, S5-130K 8 разрядов, в устройствах AG S5-110S, S5-130W, S5-150A и S5-150K - I6 разрядов, а в устройстве S5-150S -32 разряда. Информация с небольшой шириной загружается в аккумулятор " по правую сторону". Остальные разряды заполняются «0».

Загрузка от периферийных блоков касается только блоков ввода. Параметр от 0 до 127 адресует те двоичные вводы, которые служат для отображения процесса. С помощью параметра от 128 до 255 можно обращаться, например, к аналоговым блокам ввода.

Загрузка байтов

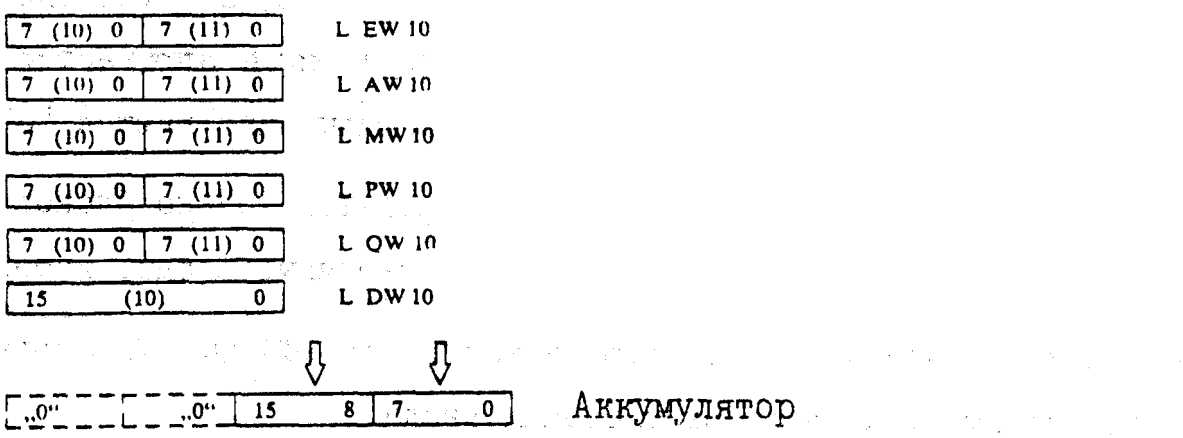


L EB 10 Загрузка I0-го байта входов /отображение процесса/
L AB 10 Загрузка I0-го байта выходов /отображение процесса/
L MB 10 Загрузка I0-го байта меток
L PB 10 Загрузка I0-го байта периферии процесса
L QB 10 Загрузка I0-го байта расширенной периферии процесса
L DR 10 Загрузка правого байта I0-го слова данных
L DL 10 Загрузка левого байта I0-го слова данных

При адресации операндов E, A, M, P и Q по байтам во время загрузки байтов содержимое байтов команд языка STEP-5 загружается в аккумуляторы.

Так как данные адресуются пословно, при загрузке байтами следует указывать "правый байт" или "левый байт". Тогда их содержимое будет загружаться в аккумулятор по правую сторону.

Загрузка слов

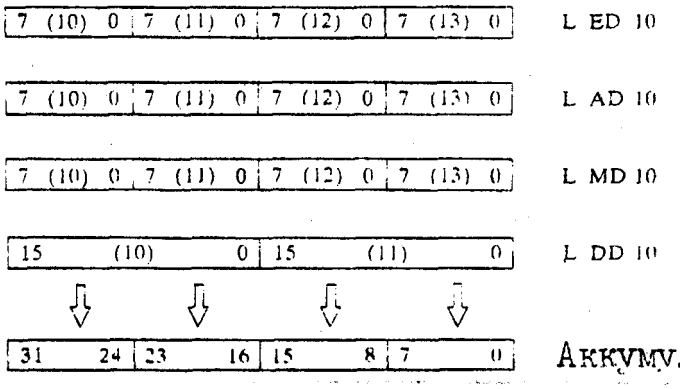


L EW 10 Загрузка IO-го слова входов /отображение процесса/
L AW 10 Загрузка IO-го слова выходов /отображение процесса/
L MW 10 Загрузка IO-го слова меток
L PW 10 Загрузка IO-го слова периферии процесса
L QW 10 Загрузка IO-го слова расширенной периферии процесса
L DW 10 Загрузка IO-го слова данных

При адресации операндов E, A, M и P по байтам во время загрузки слов указывается номер соответствующего более низкого байта команды STEP-5. Содержимое этого байта и содержимое ближайшего более высокого байта загружаются в аккумулятор. Пример: с помощью команды "L EW10" в аккумулятор загружаются содержимое 10-го байта и содержимое 11-го байта входов.

При загрузке данных в аккумулятор загружается содержимое слова, указанного в командеSTEP-5.

Загрузка двойных слов



L ED 10 Загрузка IO-го двойного слова входов /отображение процесса/
L AD 10 Загрузка IO-го двойного слова выходов /отображение процесса/
L MD 10 Загрузка IO-го двойного слова меток
L DD 10 Загрузка IO-го двойного слова данных

При адресации операндов E, A и M по байтам во время загрузки слов двойной длины указывается номер соответствующего более низкого байта команды STEP-5. Содержимое этого байта и содержимое 3 ближайших более высоких байтов загружается в аккумулятор. Пример: с помощью команды "LED To" в аккумулятор загружается содержимое байтов К° 10, II, 12 и 13.

При адресации данных пословно указывается номер соответствующего более низкого слова команды STEP-5. Содержимое этого слова и содержимое ближайшего более высокого слова загружаются в аккумулятор.

Загрузка блоков ввода/вывода словами двойной длины невозможна.

Загрузка констант

При необходимости загрузить в аккумулятор какую-либо постоянную величину /например, при сравнении на предельное значение/, имеется возможность заложить это значение в виде константы в программу. Представление константы зависит от ее назначения. Возможны следующие виды представлений?.

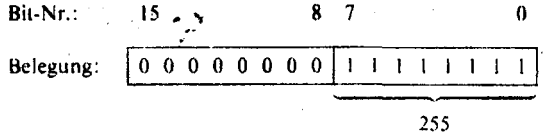
Команда	Задание константы в виде
L KB	8-разр.числа с фикс.запятой
L KF	I6-разр.числа с фикс.запятой и знаком
L KG	числа со скользящей запятой
L KM	комбинации битов /I6 разр./
L KH	I6-ричной комбинации /4 знака/
L KY	в виде двух 8-разр.чисел с фикс.запятой
L KC	в виде двух буквенно-числовых знаков
L KT	параметра времени с базой времени
L KZ	параметра счета

- загрузка константы в виде байта LKB. Константа в виде 8-разрядного числа с фиксированной запятой находится в области от О до +255 и загружается в аккумулятор по правую сторону. Остальные разряды заполняются «0».

Примеру

После выполнения команды LKB255 аккумулятор имеет следующее содержание:

распределение:

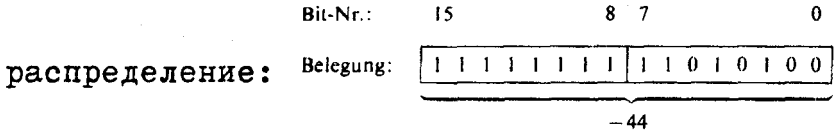


- загрузка константы в виде числа с фиксированной запятой LKB

Константа в виде 16-разрядного числа с фиксированной запятой и со знаком находится в области от -32768 до +32767. Представление отрицательных констант получают путем двойного отрицания /см.раздел 7.2/

Пример:

После выполнения команды LKF-44 аккумулятор показывает следующее содержание:



В устройстве автоматизации S5-150S остальные биты аккумулятора /с 16 по 31/ будут заняты или «0» /при положительной константе/, или «1» /при отрицательной константе/.

- загрузка константы в виде числа со скользящей запятой LKG

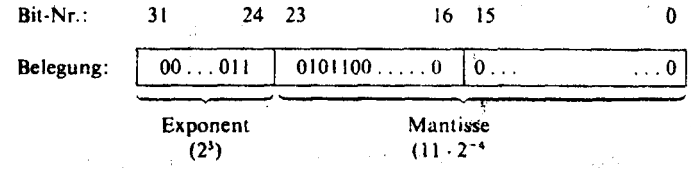
Эта команда допустима только в устройстве S5-150S

Константа в виде 32-разрядного числа со скользящей запятой

находится в области примерно $1{,}7*10^{38}$. Представление дается раздельно по экспоненту и мантиссе /см.раздел 14.5/.

Пример:

После выполнения команды LKG+55+01 аккумулятор показывает следующее содержание:

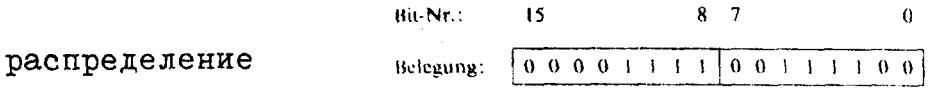


- загрузка константы в виде "комбинации битов LKM

В виде константы есть 16-разрядная комбинация битов. «0» и «1» чередуются произвольно.

Пример:

После выполнения команды *L KM 0000 1111 0011 1100* аккумулятор показывает следующее содержание:



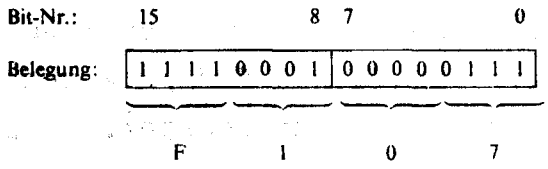
- загрузка константы в виде 16-ричной комбинации LKH

Константа представляет собой 4-значную 16-ричную комбинацию в области от 0000 до FFFF

Пример:

После выполнения команды *L KH F107* аккумулятор показывает следующее содержание:

распределение:



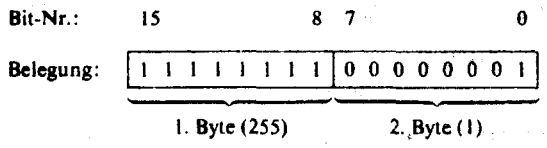
- загрузка константы в виде двух байтов L KY

Константа представляет собой 8-разрядное число с фиксированной запятой, имеющее двойную длину и разделенное запятой. Область каждого числа от 0 до 255.

Пример:

После выполнения команды *L KY 255,1* аккумулятор показывает следующее содержание:

распределение:



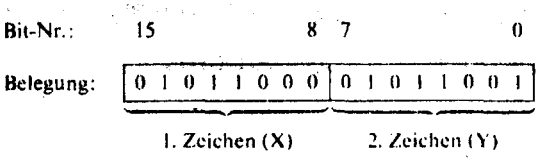
- загрузка константы в виде двух знаков L KC

Константа представляет собой два знака в коде ASCII-
Знаки пишутся без пробела.

Пример:

После выполнения команды *L KC XY* аккумулятор показывает следующее содержание:

распределение:



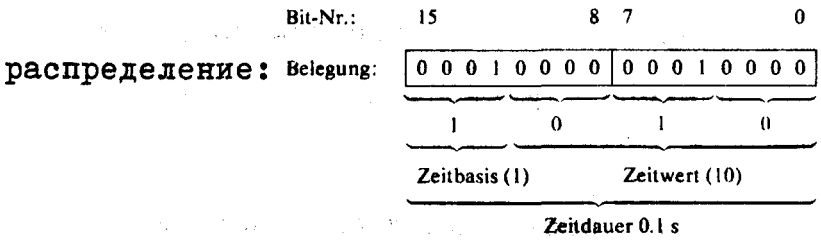
I-й знак 2-й знак

- загрузка константы в виде параметра времени LKT

Константа представляет собой уставку времени, которая требуется при запуске таймера. Она состоит из 3-значйого параметра времени в области от 000 до 999 и базы времени в области от 0 до 3. Оба числа разделяются точкой. Числа загружаются в аккумулятор в двоично-десятичном коде.

Пример:

После выполнения команды L KT10.1 аккумулятор показывает следующее содержание:



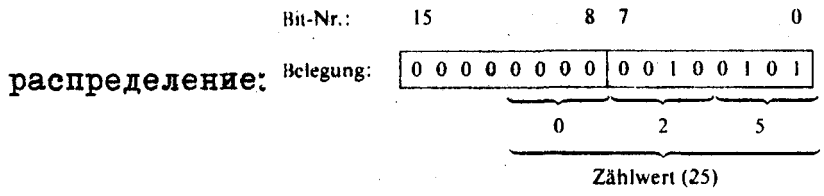
база времени параметр времени
продолжительность

- загрузка константы в виде параметра счета L KZ

Константа представляет собой параметр счета /который необходим для установки счетчика/. Она состоит из 3- значного числа в области от 000 до 999. Число загружается в аккумулятор в двоично-десятичном коде.

Пример:

После выполнения команды *L KZ 25* аккумулятор показывает следующее содержание:



параметр счета

Аккумулятор, вспомогательный аккумулятор

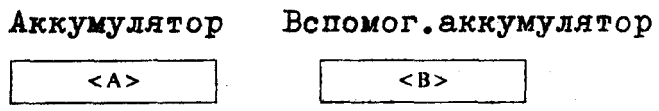
При обработке дискретных значений в качестве промежуточных ЗУ используются регистры блока управления. Эти регистры называются аккумулятор /аккумулятор I/ и вспомогательный аккумулятор /аккумулятор 2/.

Содержание обоих аккумуляторов изменяется за счет операций загрузки. Значение операнда, стоящего при операции загрузки, переносится в аккумулятор. Одновременно свое первоначальное содержание аккумулятор передвигает во вспомогательный аккумулятор.

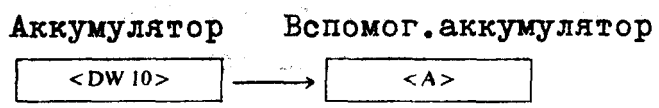
Пример:

В аккумулятор необходимо загрузить слово данных DW10: L DW10

Содержание аккумуляторов перед выполнением загрузки



Содержание аккумуляторов после выполнения загрузки L DW 10:



При этом вспомогательный аккумулятор свое содержание теряет.

6.2 Перенос

С помощью этой операции можно переносить информацию из аккумулятора в области операндов Е, А, М ив, а также непосредственно в блоки ввода/вывода.

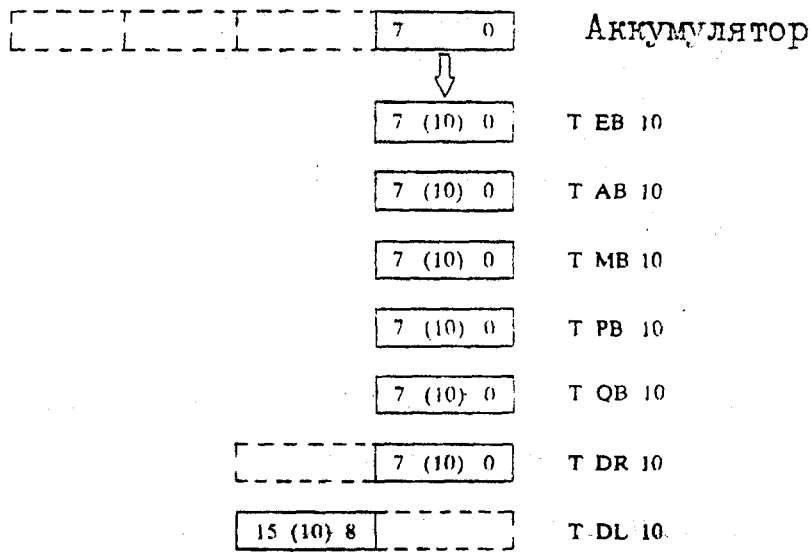
Содержимое аккумулятора в области операндов переносится байтами, словами или словами двойной длины, а в блоки ввода/вывода - байтами или словами. При этом содержимое аккумулятора не изменяется. Оно остается постоянным, что позволяет делать многократные переносы.

Ширина аккумулятора составляет в устройствах автоматизации S5-130A, S5-150K 8 разрядов, в устройствах S5-110S,S5-130W,S5-150A, и S5-150K - I6 разрядов, и в устройствах S5-150S - 32 разряда. Если переносимая информация имеет небольшую ширину, она выбирается по правой части аккумулятора.

Перенос в блоки периферии касается только блоков вывода. Параметром от 0 до 127 адресуются двоичные выводы, относящиеся к отображению процесса. Перенос в эти блоки периферии в устройствах S5-110S,S5-130W,S5-150A, и S5-150K и ведёт одновременно к изменению параметра в отображении процесса.

С помощью параметра от 128 до 255 можно обращаться, например, к аналоговым блокам вывода.

Перенос байтов

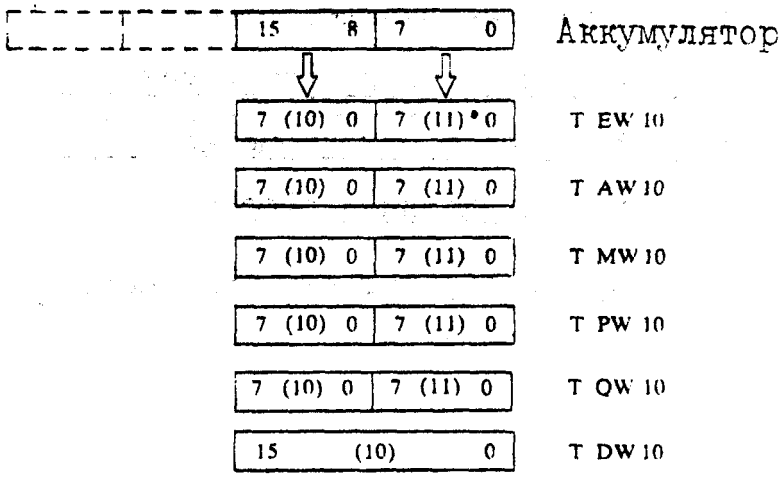


T EB 10	Перенос в IO-й байт входов /отображение процесса/
T AB 10	Перенос в IO-й байт выходов /отображение процесса/
T MB 10	Перенос в IO-й байт меток
T PB 10	Перенос в IO-й байт периферии процесса
T QB 10	Перенос в IO-й байт расширенной периферии процесса
T DR 10	Перенос в правый байт IO-го слова данных
T DL 10	Перенос в левый байт IO-го слова данных

При адресации операндов E, A, M и P по байтам во время переноса содержимое байта, стоящего в аккумуляторе "справа", переносится в байт, указанный в команде STEP-5.

Так как адресация данных производится по словам, при переносе обязательно нужно указывать "правый байт" или "левый байт". Тогда перенос будет произведен в этот байт. Остальной байт данных воздействию не подвергается.

Перенос слов

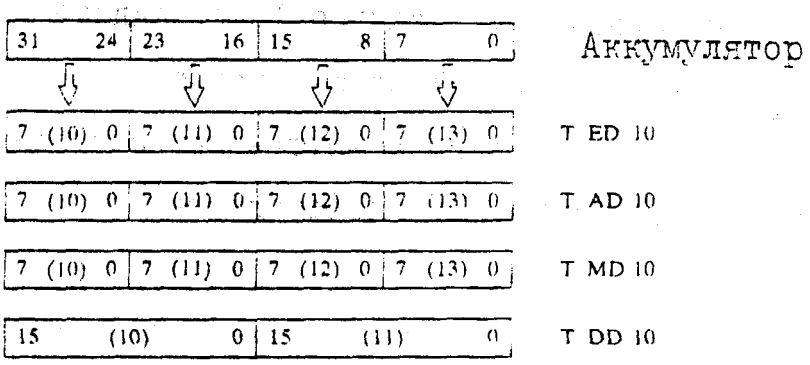


T EW 10 Перенос в IO-е слово входов /отображение процесса/
T AW 10 Перенос в IO-е слово выходов /отображение процесса/
T MW 10 Перенос в IO-е слово меток
T PW 10 Перенос в IO-е слово периферии
T QW 10 Перенос в IO-е слово расширенной периферии
T DW 10 Перенос в IO-е слово данных

При адресации операндов E, A, M и P по байтам во время переноса слов указывается номер соответствующего более низкого байта в команде STEP-5. Содержимое стоящего в аккумуляторе справа слова переносится в этот байт и в ближайший байт более высокого номера. Например: с помощью команды "TAW10" содержимое аккумулятора с 0 по 7-й разряд переносится в 11-й байт, а содержимое аккумулятора с 8 по 15 разряд - переносится в 10-й байт выходов.

При переносе данных содержимое аккумулятора с 0 по 15 разряд переносится в слово данных, указанное в команде STEP-5.

Перенос двойных слов



T ED 10 Перенос в IO-е двойное слово входов /отображ.процесса/

T AD 10 Перенос в IO-е двойное слово выходов /отображ.процесса/

T MD 10 Перенос в IO-е двойное слово меток

T DD 10 Перенос в IO-е двойное слово данных

При адресации операндов E, A, и M по байтам во время переноса двойных слов указывается номер соответствующего более низкого байта в команде STEP-5. Содержимое аккумулятора будет перенесено в этот байт и в 3 ближайших байта с более высокими номерами. Например: с помощью команды " TAD10" содержимое аккумулятора с 0 по 7-й разряд переносится в 13-й байт, содержимое разрядов с 8 по 15 переносится в 12-й байт, содержимое разрядов с 16 по 23 в 11-й байт, и содержимое разрядов с 24 по 31 - в 10-й байт выходов.

При переносе данных содержимое аккумулятора с 0 по 15 разряд переносится в слово данных, указанное в команде STEP-5, а содержимое разрядов с 16 по 31 переносится в ближайшее более высокое по порядку слово данных.

Перенос слов двойной длины в блоки ввода/вывода невозможен.

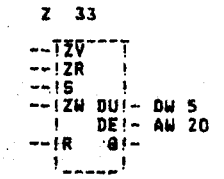
6.3 Загрузка числовых значений

В области операндов, называемых счетчиками, каждое 16-разрядное слово образует один счетчик. Счетчик состоит из 6 разрядов, которые лают процессору необходимую информацию о внутреннем состоянии счетчика, и из 10 разрядов, изображающих непосредственно числовое значение.

Если сброс счетчика должен действовать "статически" и не зависеть от логического результата на входе установки счетчика, нужно, чтобы /в таблице команд/ сброс счетчика программировался сразу же после прямого или обратного счета и установки счетчика, но еще перед загрузкой числового значения /см.также раздел 5.2/. При сбросе числовое значение устанавливается на ноль.

Графическое изображениё

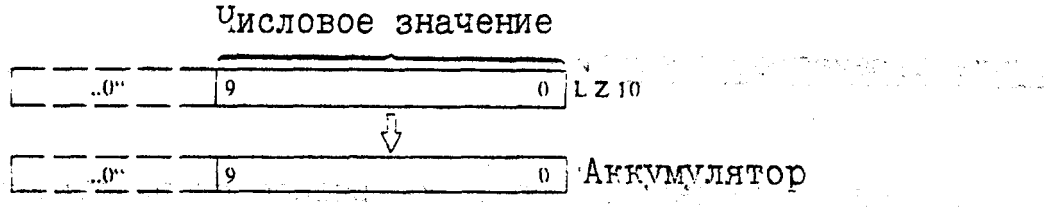
При графическом изображении, как на функциональной, так и на контактной схеме, двоичный выход числового значения обозначается буквами "DU", а десятичный выход - буквами "DE".




```
L   Z   33        Загрузка числового значения
T   DW  5

LC  Z   33        Кодированная загрузка числового значения
T   AW  20
```

Прямая загрузка числового значения

Числовое значение хранится в памяти в двоичном коде. В таком виде его можно загрузить в аккумулятор. Разряды состояния не загружаются; вместо них в аккумуляторе будет стоять «0».

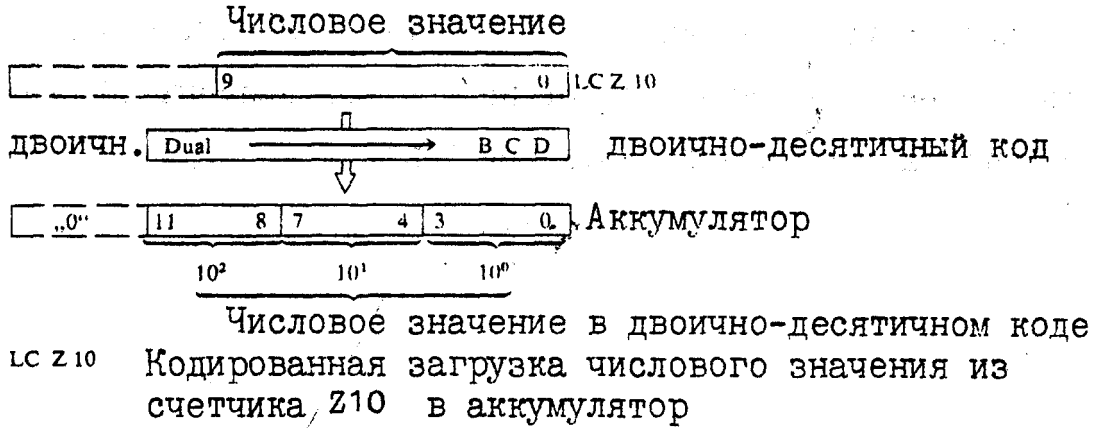


L Z 10 Прямая загрузка числового значения из счетчика Z10 в аккумулятор

Теперь значение, находящееся в аккумуляторе, может быть подвергнуто дальнейшей обработке. Перенос из аккумулятора в счетчик невозможен. Если счетчик требуется установить на какую-либо величину, это делается с помощью операции "Установка счетчика" /см. раздел 5/.

Кодированная загрузка числового значения

Имеющееся в двоичном виде числовое значение также может быть загружено в аккумулятор в "кодированном" виде. В двоично-десятичном коде эта числовая величина хранится в аккумуляторе в ожидании дальнейшей обработки.



LC Z 10 Кодированная загрузка числового значения из счетчика Z10 в аккумулятор

Разряды состояний счетчика также не учитываются при кодированной загрузке. Числовое значение в двоично-десятичном коде может быть подвергнуто дальнейшей обработке, например, перенесено на выходы, к которым подключена цифровая индикация.

6.4 Загрузка параметров времени

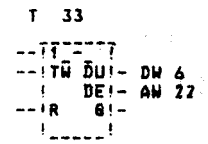
В области операндов времени каждое 16-разрядное слово образует одну (функцию времени). Функция времени состоит из параметра времени /10 разрядов/, дискретности /2 разр./и 4 разрядов состояний, которые показывают процессору состояние функции времени при ее обработке.

Чтобы сброс таймера действовал "статически" и не зависел от логического результата на других входах установок времени, нужно, чтобы /в таблице команд/ сброс таймера программировался сразу же за запуском но еще перед загрузкой параметра времени /см.также раздел 4.2/.

При сбросе таймера параметр времени не изменяется. Его значение остается тем же, что и в момент сброса.

Графическое изображение

При графическом изображении, как на функциональных, так и на контактных схемах, двоичный выход параметров времени обозначается буквами "DU", а десятичный - буквами "DE".



Загрузка параметра времени

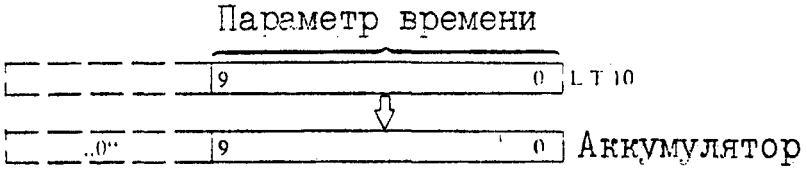
```
L   T   33
T   DW  6
```

Кодированная загрузка параметра времени

```
LC  T   33
T   AW  22
```

Прямая загрузка параметра времени

Параметр времени хранится в памяти в двоичном виде. В таком виде его можно загружать в аккумулятор. Биты состояний и дискретность не загружаются; вместо них в аккумуляторе стоит «0».



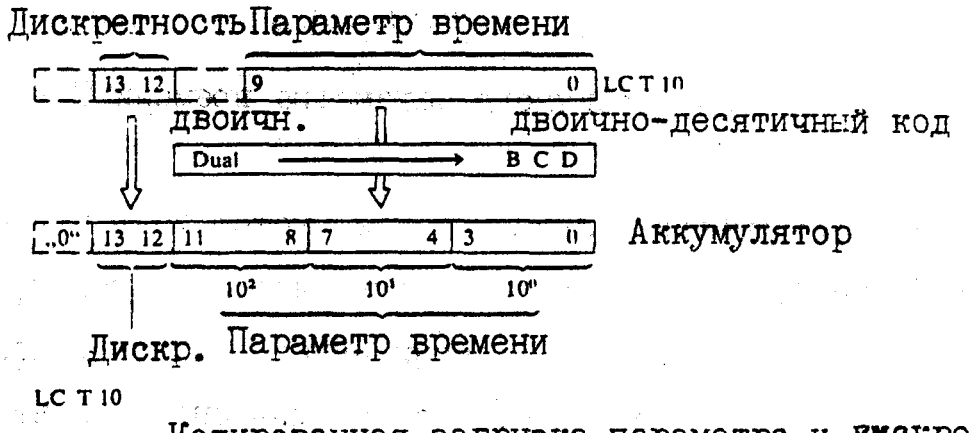
L T 10 Прямая загрузка двоичного параметра времени T IO в аккумулятор

Теперь значение, находящееся в аккумуляторе, может быть подвергнуто дальнейшей обработке. Перенос из аккумулятора в функцию времени невозможен. Если функции времени нужно придать какое-то значение /установить/, то это делается с помощью операции запуска

Функции времени /см.раздел 4/.

Кодированная загрузка параметра времени

Параметр времени, имеющийся в двоичном виде, может также загружаться в аккумулятор в "закодированном" виде. В этом случае в двоично-десятичном коде получают не только параметр, но и дискретность.



Кодированная загрузка параметра и дискретности времени T IO в аккумулятор

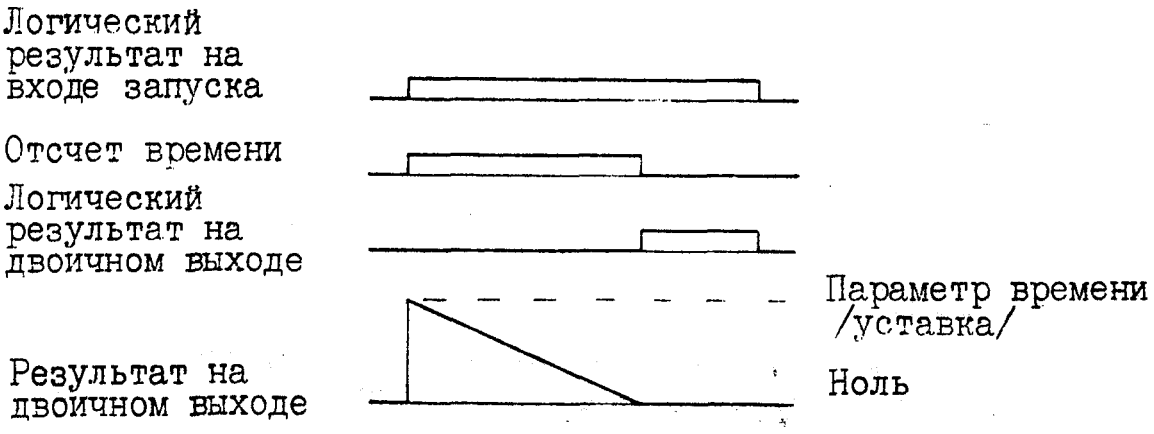
При кодированной загрузке разряды состояний не учитываются;
однако в аккумулятор загружается дискретность. Параметр времени, преобразованный в двоично-десятичный код, может подвергаться дальнейшей обработке, например, переноситься на выходы, к которым подключена цифровая индикация.

Опрос параметра времени

Опрос функции времени /в двоичном виде/ характеризуется исключительно спецификой" временных диаграмм /см.раздел 4/. Выполненное условие опроса на состояние «1» не во всех случаях идентично характеристике "отсчет времени идет". Другими словами: цифровое значение времени только тогда не равно нулю, если ведется отсчет времени, независимо от двоичного результата опроса функции времени.

Пример:

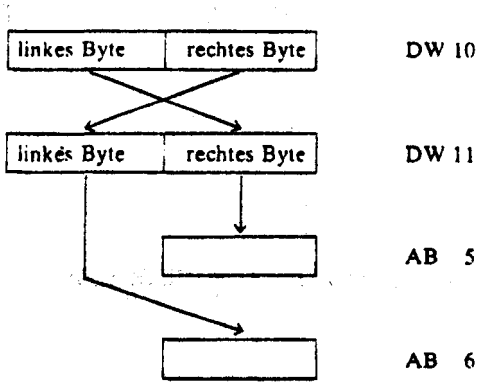
Запуск таймера в режиме задержки включения и индикация времени



6.5 Пример программирования "загрузки и переноса"

Оба байта слова данных DW 10 требуется внести в слово данных DW 11, поменяв их местами. Кроме того их нужно перенести в байты выхода AB 5 и AB 6 по следующей схеме:

левый байт правый байт



Программирование

```
L   DR  10 }
T   DL  11 }   Информация правого байта слова DW 10
T   AB  6  }   переносится в левый байт слова DW 11 и в байт
               выхода AB 6.

L   DL  10 }
T   DR  11 }   Информация левого байта слова DW 10
T   AB  5  }   переносится в правый байт слова DW 11 и в байт
               выхода AB 5.
```

7 ПРЕДСТАВЛЕНИЕ ЧИСЕЛ

Процессоры устройств автоматизации S5-110S,S5-130W,.S5-l50A, S5-150K и S5-150S могут обрабатывать информацию, представленную в числовом виде. Так как процессоры работают на базе двоичного кода, то как правило, непосредственно обрабатываются только величины, закодированные двоичным кодом. Раздел 7.1 знакомит с десятичной и двоичной системами счисления и с представлением чисел в двоичном виде.

В языке программирования STEP5 приняты три различных способа представления чисел:

- в виде 16-разрядных чисел с фиксированной запятой /раздел 7.2/
- в виде 32-разрядных чисел с фиксированной запятой /раздел 7.3/ и
- в виде чисел с плавающей запятой /раздел 7.4/.

Кроме этих представлений чисел в двоичном кодера некоторых случаях могут обрабатываться и величины, закодированные в двоично-десятичном коде. При переносе этих величин в области операндов они, однако, подвергаются преобразованию в двоичный код. После этого дальнейшая обработка ведется в двоичном виде.

Устройства автоматизации S5-110S, S5-130W, S5-150A и 55-150K обрабатывают только 16-разрядные числа с фиксированной запятой; S5-150S работает с числами в любом представлении.

7.1 Системы счисления

Десятичная система счисления, с которой мы сталкиваемся повседневно, для изображения числовых величин /чисел/ пользуется 10 цифрами от 0 до 9. С помощью этих цифр можно образовать любое число десятичной системы счисления. Цифры относятся к числам так же, как буквы к словам.

Однако цифры сами по себе еще не все. При составлении цифр в числа важно также положение /позиция/ цифр. Значение каждой входящей в число цифры зависит и меняется от ее положения /разряда/ в записи числа. Эти разряды в десятичной системе счисления являются степенями 10. Число 10 называют поэтому также "основанием" десятичной системы счисления. Цифры в соответствующих местах числа указывают, сколько раз представлено соответствующее разрядное значение. Любое число можно представить как сумму всех произведений цифр на разрядные значения.

Пример:

Число 1024 можно представить как сумму следующих произведений:

1		0		2		4	
1×10^3	+	0×10^2	+	2×10^1	+	4×10^0	/цифра x разряд/
1000	+	0	+	20	+	4 = 1024	

Такие системы счисления называют "позиционными". Позиционные системы счисления различаются по основаниям: двоичные имеют основание 2, восьмеричные - основание 8, шестнадцатеричные - основание 16 и т.д.

Двоичная система счисления особенно удобна для представления чисел с помощью электронных приборов /например ЭВМ/. Основанием этой системы является число 2. Поэтому она имеет только две цифры 0 и 1. С помощью этих цифр особенно просто изображать, например, состояния "напряжение отсутствует" и "напряжение присутствует".

Если эти состояния рассматривают отдельно, без учета их положения /разрядности/, то говорят о "двоичных величинах", как например, состояние сигнала «0» и состояние сигнала «1».. Логические операции над этими величинами называются "коммутационной алгеброй, а эти же операции с использованием законов, описанных в разделе 2, называются "Булевой алгеброй".

Если обрабатывают цифры от 0 до 1, то говорят о "пирровых величинах". Двоичная система счисления по своей структуре похожа на десятичную. Положение цифр называют разрядом, который представляет собой степень основания системы счисления, т.е. степень 2. Цифры указывают на наличие /цифра I/ и отсутствие /цифра 0/ разряда.

Пример:

Перевод двоичного числа 1011 в десятичную систему производится следующим образом:

```
1        0        1        1
1×2³ +   0×2² +   1×2¹ +   1×2⁰     /цифра x разряд/
8    +   0    +   2    +   1  =  11
```

Двоичное число 1011 имеет то же значение, что и десятичное число II. Запись очень больших двоичных чисел только с помощью 0 и 1 получается довольно громоздкой. Поэтому большие двоичные числа изображаются символами шестнадцатеричной системы счисления. Шестнадцатеричной называется система счисления с основанием 16.

Ниже приводятся некоторые цифры шестнадцатеричной системы счисления с соответствующими десятичными и двоичными значениями /см.стр.165/. Основанием шестнадцатеричной системы счисления служит 4-я степень основания 2 двоичной системы счисления. Отсюда следует, как это видно из приводимой таблицы, что на каждые 4 разряда двоичного числа /тетраду/ при изображении приходится одна цифра шестнадцатеричной системы счисления.

шестнадцатиричные цифры	десятичные цифры	двоичные цифры
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111
10	16	1 0000
11	17	1 0001
...	..	....

Пример:

Изображение 16-разрядного двоичного числа в виде шестнадцатиричного числа:

1001	0000	0110	0010	двоичн.
9	0	6	2	шестнадцатиричн.

Написать и прочитать число 9062 гораздо легче", чем 16-разрядное двоичное число. Чтобы отличать шестнадцатеричное число от десятичного, после шестнадцатеричного пишется буква "H", т.о. 9062H.

"H" - начальная буква латинского слова hex -шесть. Шестнадцатеричные числа также относительно просто преобразуются в двоичные числа.

Пример:

Преобразование шестнадцатиричного числа 1FA3H в двоичное число:

1	F	A	3	шестнадцатиричное
0001	1111	1010	0011	двоичное

Шестнадцатиричные числа являются всего лишь вспомогательной Формой представления двоичных чисел. Чтобы определить значение какого-либо двоичного числа по десятичной системе, приходится прибегать к пересчетным таблицам. Обратный путь построения двоичного числа, чтобы можно было непосредственно читать его десятичное значение, значительно проще. Для этого пользуются так называемым двоично-десятичным кодом (binary coded decimal code (-BCD) = двоично-кодированный десятичный код).

При изображении двоичных чисел в двоично-десятичном коде разряды десятичного числа /степени основания 10/ сохраняются. В двоичном виде изображаются только цифры десятичного числа.

Пример:

Представление десятичного числа 1024 в двоично-десятичном коде:

1	0	2	4	
0001	0000	0010	0100	в коде BCD

десятичное

Здесь применяется тот же самый принцип, что и при преобразовании шестнадцатиричных чисел в двоичные. Так же просто происходит преобразование чисел в коде BCD в десятичные числа.

Пример:

Представление числа, закодированного двоично-десятичным кодом, в виде десятичного числа:

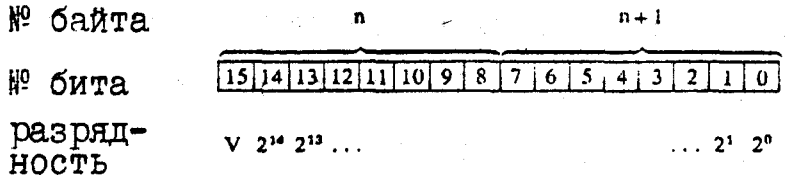
1000	0111	0100	0011	в коде BCD
8	7	4	3	десятичное

В цифрах в двоично-десятичном коде используются не все возможные тетрады двоичного числа. В таком изображении /десятичные/ величины от 10 до 15 шестнадцатиричные цифры от A до не встречаются. Поэтому тетрады, изображающие эти величины, в двоично-десятичном коде называются "псевдотетрадами".

7.2 16-разрядные числа с фиксированной запятой

16-разрядные.числа с Фиксированной запятой представляют собой двоичные числа со знаком. Они могут принимать как положительное, так и отрицательное значение. В языке программирования STEP 5 эти числа отмечаются буквой "F:".

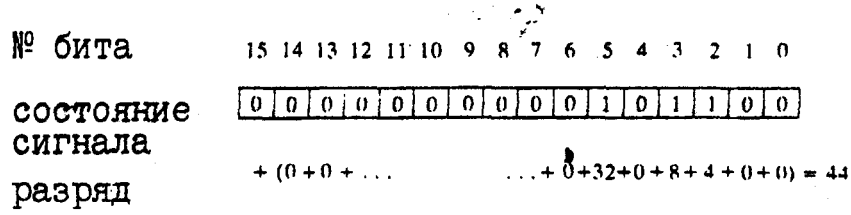
Одно 16-разрядное число с фиксированной запятой занимает одно слово. Распределение информации по битам внутри этого слова выглядит следующим образом:



Состояние сигнала бита №15 представляет знак(V). Сигнал «0» означает, что число положительное. Сигнал «1» обозначает отрицательное
число.

Разрядность отдельных битов положительного числа также указана в изображенном выше графике. При сигнале «0» в бите показатель степени не учитывается. Значение числа представляет собой сумму всех разрядов, биты которых имеют сигнал «1».

Пример:



Таким образом, число 002CH в десятичной системе имеет значение +44. Максимальное значение, которое может иметь положительное 16-газряд-ное число, представляет собой сумму всех разрядов битов от 0 до 14:

F: $Z_{max} = +2^{15} - 1 = 32\,767$

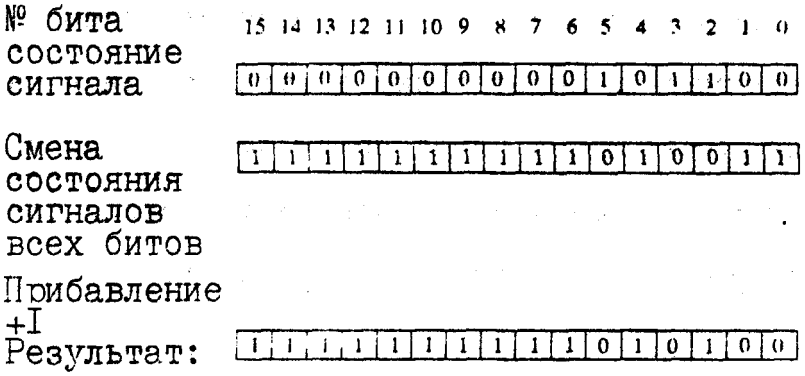
7FFFH является изображением этого числа в шестнадцатеричной системе. Самое малое положительное число имеет значение 0. Оно изображается как 0000H.

Отрицательное число изображается двоичном дополнением. Двоичное дополнение получают путем смены состояния сигналов всех битов и прибавлением затем к этому числу +1.

Пример:

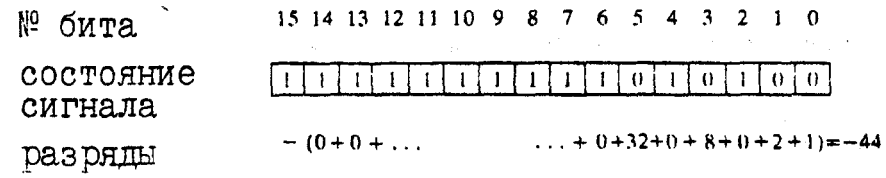
Представление десятичного числа -44:

Комбинация битов числа +44 выглядит следующим образом:



В соответствии с этим число FFD4H имеет десятичное значение -44. Десятичное значение отрицательного числа с фиксированной запятой получают аналогичным образом с той лишь разницей, что суммируются разряды с состоянием сигнала «0» и к результату прибавляется +1.

Пример:



Минимальное значение для числа с фиксированной запятой получается в том случае, когда биты с № 0 по № 14 имеют состояние сигнала «0».

Математическое выражение этого числа,:

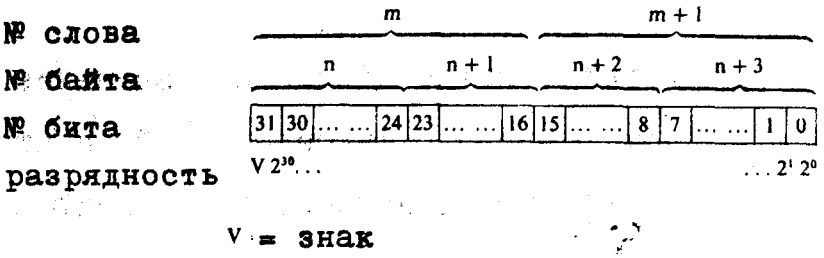
F: $Z_{min} = -(2^{15}-1+1) = -2^{15} = -32\,768$

В шестнадцатеричном выражении это число выглядит как 8000H. Значение -I как самое большое, отрицательное число запишется выражением FFFFH.

7.3 32-разрядные числа с фиксированной запятой

32-разрядные числа с фиксированной запятой являются двоичными числами со знаком. Они могут принимать положительные и отрицательные значения. В языке программирования STEP5 эти числа обозначаются буквой ”D”

32-разрядное число с фиксированной запятой занимает двойное слово. Отдельные биты такого слова двойной длины распределяются следующим образом:



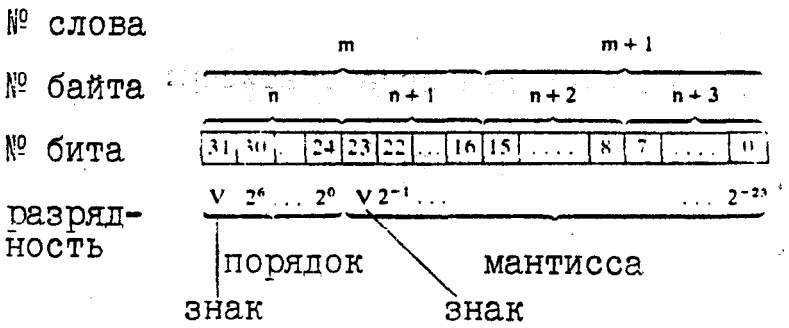
32-разрядные числа с фиксированной запятой обрабатываются так же, как и 16-разрядные с фиксированной запятой /см.раздел 7.2/. Отличие заключается лишь в увеличении области чисел.

D: Z_{max} = +2 147 483 647 (7 FFF FFFF H)
D: Z_{min} = −2 147 483 648 (8 000 0000 H)

7.4 Числа с плавающей запятой

Числа с плавающей запятой являются показательными числами, оба компонента которых - мантисса и показатель /порядок/ представляются как числа с Фиксированной запятой в двоичном виде. Мантисса изображается как 24-разрядное число с фиксированной запятой, а показатель /порядок/- в виде 8-разрядного числа с фиксированной запятой. В языке программирования STEP 5 эти числа отмечаются буквой "G:".

Число с плавающей запятой всегда занимает двойное слово. Распределение информации по битам этого двойного слова выглядит следующим образом:



Значение числа с плавающей запятой определяется выражением

$$Ч_{п.з.} = \text{мантисса} \times 2^{(\text{показатель})}$$

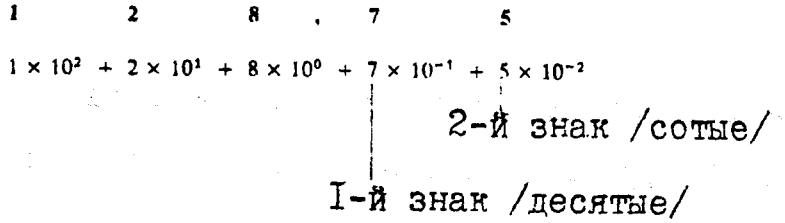
Мантисса представляется битами с 0 по 23. Состоянием сигнала в 23 бите задается знак мантиссы, который является также знаком всего числа. Состояние «0» означает, что число положительное. Состояние «1» обозначает отрицательное число.

Мантисса представляет дробную часть двоичного числа, стоящего после запятой. Это можно пояснить на примере из десятичной системы Число

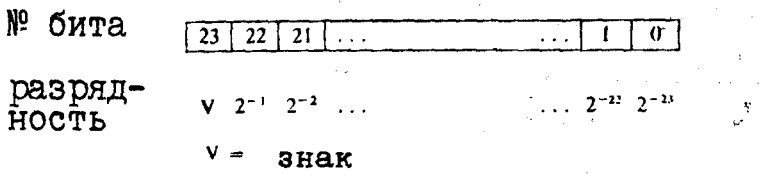
128,75 делится на две части:

128 - целое число перед запятой 75 - дробная часть числа после запятой.

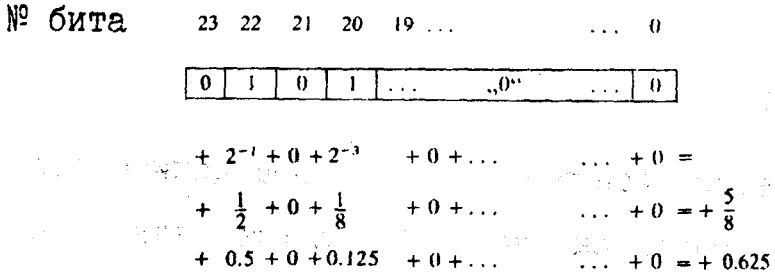
Разрядность числа, стоящего после запятой, образуется отрицательными значениями степени основания:



Аналогичным образом определяется значение мантиссы:



Пример:



Крайние значения положительной мантиссы составляют:

полож.мантисса$_{макс.}$ G: pos. Mantisse$_{max}$ = $2^0 - 2^{-23} \approx 1$ (7F FFFFH)

полож.мантисса$_{мин.}$ G: pos. Mantisse$_{min}$ = 0 (00 0000H)

Крайние значения отрицательной мантиссы составляют:

отриц.мантисса$_{макс.}$ G: neg. Mantisse$_{max}$ = $-2^{-23} \approx 0$ (FF FFFFH)

отриц.мантисса$_{мин.}$ G: neg. Mantisse$_{min}$ = $-(2^0 - 2^{-23}) \approx -1$ (80 0000H)

Порядок числа, представлен битами 24 - 31. Состоянием сигнала бита 31 определяется знак порядка. Сигнал «0» означает положительный порядок. Сигнал «1» обозначает отрицательный порядок.

Представление положительного или отрицательного порядка соответствует изображению числа с фиксированной запятой в разделе 7.2. Крайние значения будут выглядеть следующим образом:

порядок$_{макс.}$ G: Exponent$_{max}$ = $+(2^7-1)$ = +127 (7 FH)

порядок$_{мин.}$ G: Exponent$_{min}$ = $-(2^7-1+1)$ = −128 (80 H)

Число со скользящей запятой представляет собой значение, заданное следующими границами:

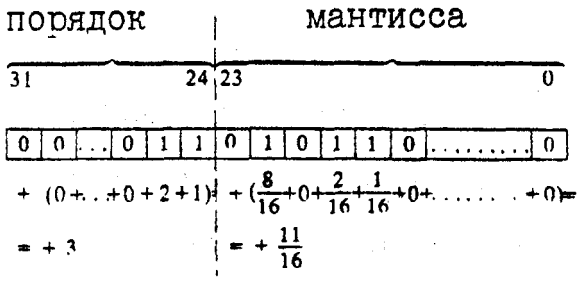
Ч$_{с.з.}$

$Z_G \approx -1{,}7 \cdot 10^{38} \ldots -1{,}4 \cdot 10^{-37}$

0

$+1{,}4 \cdot 10^{-37} \ldots +1{,}7 \cdot 10^{38}$

Пример:



I$_{с.з.}$

$Z_G = +\frac{11}{16} \cdot 2^{+3} = +\frac{11}{16} \cdot 8 = 5.5$

8. ФУНКЦИИ СРАВНЕНИЯ

Язык программирования STEP 5 дает возможность прямого сравнения содержимого /комбинаций битов/ двух цифровых операндов. Длина операндов /байт, слово, двойное слово/ указывается вместе с операндами. Есть несколько типов сравнений:

- сравнение на "равно" /раздел 8.1/
- сравнение на "не равно /раздел 8.2/
- сравнение на "больше" /раздел 8.3/
- сравнение на "больше - равно" /раздел 8.4/
- сравнение на "меньше" /раздел 8.5/ и
- сравнение на "меньше -равно" /раздел 8.6/.

Результат сравнения имеет двоичное выражение. Состояние «1» означает выполнение сравнения. Состояние «0» появляется при невыполненном сравнении. Полученный логический результат может
быть подвергнут дальнейшей обработке /см.раздел 8.7/.

Функция сравнения заключается в сравнении содержаний аккумулятора и вспомогательного аккумулятора. Поэтому до ее выполнения сравниваемые друг с другом операнды необходимо загрузить в аккумуляторы /см.раздел 6.2 "Загрузка"/.

В общем виде функция сравнения может быть представлена следующим образом:

Вспомог.акк. функция сравнения Акк. = Результат

После выполнения операции сравнения содержания обоих аккумуляторов сохраняют прежние значения.

При сравнении с оценкой содержимого операндов /например, при сравнении на "меньше"/ необходимо давать их числовое представление. Содержимое обоих операндов оценивается и сравнивается по приводимому представлению чисел. По соображениям единообразия представления чисел даются также при функциях сравнения на равенство и неравенство. Допускается использование всех представлений чисел (F,D и G).

В следующих разделах приводится обзор принятых в языке STEP5 функций сравнения, графический символ сравнений дается только один раз, так как на функциональных и контактных схемах он выглядит одинаково. Рядом пишутся команды в той, последовательности, как они были записаны в таблице команд.

В одной логической операции или в одной цепи изображается только одна функция сравнения. Ее нельзя программировать в сочетании с функциями установки и сброса памяти /RS/, таймерами и счетчиками.

8.1 Сравнение на "равно"

Функциональная и контактные схемы

Комбинация битов операндов на входе Z1 сравнивается с комбинацией битов на входе Z2. При равенстве на выходе будет состояние «1». Символ F в представлении чисел показывает, что они сравниваются по 16 разрядам, символы с и о - по 32 разрядам.

Таблица команд

Первая команда служит для загрузки содержания операнда (Z1) в аккумулятор. Вторая команда содержит второй операнд (Z2). Далее следует операция сравнения (!=) с указанием представления числа (F). В последующей команде можно производить дальнейшую обработку /двоичного/ результата сравнения.

8.2 Сравнение на "не равно"

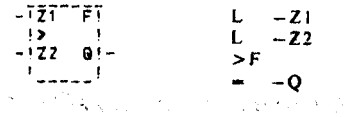
Функциональная и контактная схемы

Комбинация битов операндов, находящихся на входе Z1 , сравнивается с комбинацией битов операндов на входе Z2. При неравенстве на выходе будет состояние «1». При наличии символа F сравниваются 16 разрядов, при D и G - 32 разряда.

Таблица команд

По первой команде в аккумулятор загружается содержание одного операнда (z1). Вторая команда содержит второй операнд (Z2).. Далее следует операция сравнения (><) с указанием представления числа (F) . В последующей команде можно производить дальнейшую обработку /двоичного/ результата сравнения.

8.3 Сравнение на "больше"



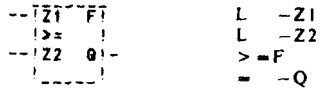
Функциональная и контактная схемы

Содержимое обоих операндов (Z1 и Z2) интерпретируется и сравнивается друг с .другом в, соответствии с представлением числа - F,D или G. Если содержимое операнда z1 больше содержимого операнда Z2, на выходе Q будет Состояние «1».

Таблица команд

По первой команде в аккумулятор загружается содержание одного операнда(z1). Вторая команда содержит второй операнд (Z2). Далее следует операция сравнения (>) с указанием представления числа (F). Сравнение выполнено, если Z1 больше Z2. Тогда в "следующей команде /при выполненном сравнении/ будет результат «1», который можно использовать в последующих операциях.

8.4 Сравнение на "больше - равно"



Функциональная и контактные схемы

Содержимое обоих операндов (Z1 и Z2) интерпретируется и сравнивается друг с другом в соответствии с представлением числа – F, D или G.. Если содержимое операнда Z1 больше или равно содержимому операнда Z2, на выходе будет состояние «1».

Таблица команд

По первой команде в аккумулятор загружается содержание первого операнда (Z1). Вторая команда содержит второй операнд (Z2). Далее следует операция сравнения (>=1)с указанием представления числа(F). Сравнение выполнено, если Z1 больше или равно Z2 . Тогда в последующей команде /при выполненном сравнении/ будет результат «1», который можно использовать в дальнейших операциях.

8.5 Сравнение на "меньше"

Функциональная и контактные схемы

Содержимое обоих операндов (Z1 и Z2) интерпретируются и сравниваются друг с другом в соответствии с представлением числа F,D или G.

Если содержимое операнда Z1 меньше содержимого операнда Z2. на выходе Q будет состояние «1».

Таблица команд

По первой команде в аккумулятор загружается содержание первого операнда (z1) . Вторая команда содержит второй операнд (Z2) . Далее следует операция сравнения (<) с указанием представления числа (F).

Сравнение выполнено, если Z1 меньше, чем Z2. Тогда в следующей команде /при выполненном сравнении/ появляется результат «1», который может быть использован в дальнейших операциях.

8.6 Сравнение на "меньше - равно"

Функциональная и контактные схемы

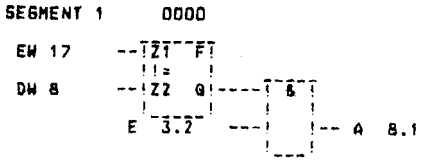
Содержимое обоих операндов (z1 и z2) интерпретируются и сравниваются друг с другом в соответствии с указанием числа F, D или G . Если содержимое операнда Z1 меньше или равно содержимому операнда Z2, на выходе Q будет состояние «1».

Таблица команд

По первой команде в аккумулятор загружается содержание первого операнда (Z)). Вторая команда содержит второй операнд (Z2). Далее следует операция сравнения (<=) с указанием представления числа (F).

Сравнение выполнено, если Z1 меньше или равно Z2. Тогда в следующей команде /при выполненном сравнении/ появляется результат "1", который может быть использован в дальнейших операциях.

8.7 Функции сравнения в логических операциях



Функциональная схема

Выход схемы сравнения может соединяться или с выходом А и соответственно с регистром меток М, или с последующими логическими операциями. К выходу схемы сравнения можно, например, подключить символы функций И или ИЛИ. Результат сравнения в этом случае будет обрабатываться на входе символа следующей функции как результат опроса.

Пример:

Если комбинация битов входа EW17 равна комбинации битов данных DW8, а на входе Е3.2 состояние «1», произойдет установка выхода А8.1.

Таблица команд [1)]

```
SEGMENT 1
0000      :U(
0001      :L    EW17
0002      :L    DW8
0003      :!=F
0004      :)
0005      :U    E 3.2
0006      :=    A 8.1
0007      :***
```

Стоящие в скобках команды представляют собой сравнение. В команде, следующей за выражением "Скобку закрыть", содержится в виде результата опроса результат сравнения, над которым можно производить дальнейшие действия по И или ИЛИ.

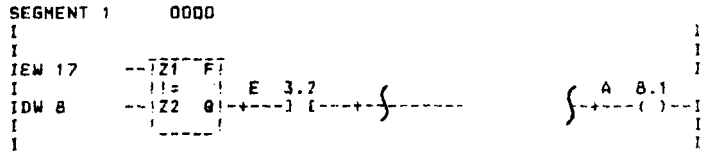
Пример:

Если комбинации битов входа EW17 и данных DW8 равны, результат опроса равен «1». Этот результат по функции И связывается с состоянием сигнала на входе Е3.2. Результат этой операции присваивается

выходу А8.1.

"Приведенные в примере операции в скобках И при программировании в виде таблицы команд опускаются. Они нужны только в случае необходимости вывести этот сегмент на индикацию или распечатку в виде функционального или контактного плана.

Контактная_схема



Символ функции сравнения может рассматриваться как "контакт", так как он подключен к левой "токоведущей шине". Если условие сравнения выполнено, контакт "замкнут" и в цепи появляется ток.

Пример:

Если комбинация битов входа EW17 и данных DW8 равны, на выходе схемы сравнения появится "ток". Если дополнительно на входе E3.2 будет состояние «1», то и этот контакт будет "замкнут" и реле /выход A3.1/ "притянет".

8.8 Пример программы "Контроль диапазона"

Значение данных DW15 должно находиться в пределах какого-то диапазона. На выходе A 16.7 появляется мигающий сигнал, как только это значение выходит за пределы диапазона. Мигание будет частым /шина M 99.0/, если значение данных DW15 больше 105. Мигание будет редким /шина M 99.1/, если значение данных DW15 меньше 77. Если значение находится в диапазоне от 77 до 105, на выходе A 16.7 будет наблюдаться постоянное свечение лампочки.

Функциональная схема

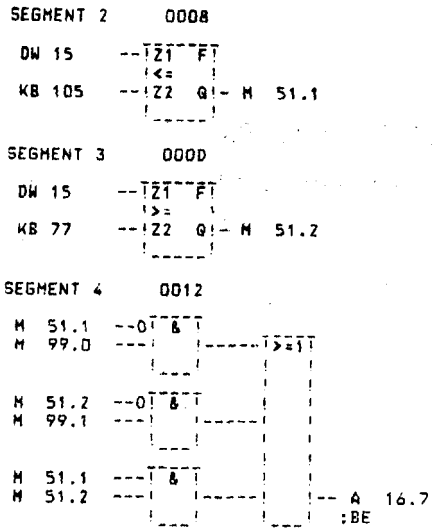


Таблица команд

```
SEGMENT 2
0008      :L    DW15
0009      :L    KB105
000A      :<=F
000B      :=    M 51.1
000C      :***
SEGMENT 3
000D      :L    DW15
000E      :L    KB77
000F      :>=F
0010      :=    M 51.2
0011      :***
SEGMENT 4
0012      :UN   M 51.1
0013      :U    M 99.0
0014      :O
0015      :UN   M 51.2
0016      :U    M 99.1
0017      :O
0018      :U    M 51.1
0019      :U    M 51.2
001A      :=    A 16.7
001B      :BE
```

DW15 меньше или равно I05

DW15 больше или равно 77

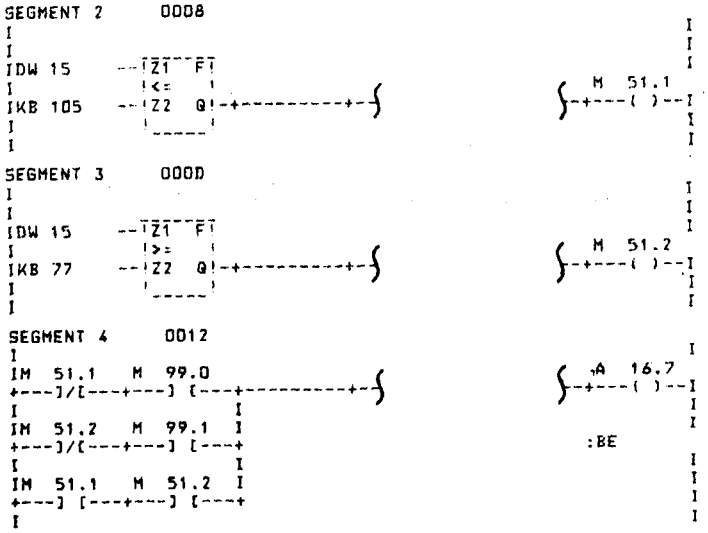
DW15 больше I05

DW15 меньше 77

DW15 в диапазоне от 77 до I05

сигнальная лампа

Контактная схема



9. ВЫЧИСЛИТЕЛЬНЫЕ ФУНКЦИИ

Язык программирования STEP 5 позволяет производить арифметические действия над содержанием /значением/ двух числовых операндов. Длина операндов /байт, слово, двойное слово/ указывается вместе с операндом. Выполняются следующие вычислительные функции:

- сложение /раздел 9.1/
- вычитание /раздел 9.2/
- умножение /раздел 9.3/ и
- деление /раздел 9.4/.

Результат вычислительной операции хранится в аккумуляторе до дальнейшей обработки. При выполнении вычислительных операций учитывается указание по представлению чисел /F или G/. Вычисления выполняются независимо от логического результата и не влияют на него. При вычислениях действия производятся над содержанием основного и вспомогательного аккумуляторов. Поэтому аккумуляторы предварительно должны быть загружены значениями тех операндов, над которыми собираются производить арифметические операций /см.раздел 6.1 "Загрузка"/, В общем виде вычислительную функцию можно представить следующим образом:

Вычислительные функции предварительно могут быть представлены только в виде таблицы команд. Поэтому при .программировании в виде функционального или контактного плана предварительно необходимо переключиться на табличную запись.

9.1 Сложение

Значения операндов слова входов EW40 и слова меток Mw52 необходимо представить и сложить как числа с фиксированной запятой. Результат присвоить слову выходов AW14.

```
L EW 40      Значение слова входов EW 40  загружается в аккумулятор.
```

L MW 52	Значение слова меток MW 52 загружается в аккумулятор; одновременно происходит перенос ранее хранившегося в нем содержания во вспомогательный аккумулятор.
+F	Содержание обоих аккумуляторов интерпретируется как I6-разрядные числа с фиксированной запятой и суммируется. Результат этого сложения хранится в основном аккумуляторе.
T AW 14	Содержание основного аккумулятора /результат/ переносится в слово выходов AW 14 .

9.2 Вычитание

Значение слова меток MW 54 требуется вычесть из значения слова данных DW 117 /действия с фиксированной запятой/. Результат записать в слове данных DW 118.

L DW 117	Значение слова данных DW 117 загружается в аккумулятор.
L MW 54	Значение слова меток загружается в аккумулятор; одновременно хранившееся ранее в аккумуляторе содержание переносится во вспомогательный аккумулятор.
-F	Содержание аккумулятора (MW 54) вычитается из содержания (DW 117) вспомогательного аккумулятора; вычитание происходит как с числами с фиксированной запятой. Результат этого вычитания хранится в аккумуляторе.
T DW 118	Содержание аккумулятора /результат/ переносится в слово данных DW 118.

9.5 Умножение

Записанное в слове данных DW 67 число с фиксированной запятой требуется умножить на I20 и перенести в слово выходов AW 102.

L DW 67	Значение слова данных DW 67 загружается в аккумулятор.
L KF +120	Величина I20 загружается в аккумулятор; одновременно хранившееся ранее в нем содержание переносится во вспомогательный аккумулятор.
XF	Содержание обоих аккумулятор интерпретируется и перемножается как I6-разрядные числа с фиксированной запятой. Результат этого умножения хранится в аккумуляторе.
T AW 102	Содержание аккумулятора /результат/ переносится в слово выходов.

9.4 Деление

Стоящее в слове "данных DD50 число с плавающей запятой требуется разделить на число с плавающей запятой двойного слова меток MD196

Результат записать в двойное слово данных 0052.

L DD 50	Значение двойного слова DD 50 загружается в аккумулятор.
L MD 196	Значение двойного слова меток MD 196 загружается в аккумулятор ; одновременно предыдущее содержание аккумулятора переносится во вспомогательный аккумулятор.
G	Содержание вспомогательного аккумулятора (DD 50) делится на содержание аккумулятора (MD 196) ; деление производится как операция над числами с плавающей запятой. Результат этого деления хранится в аккумуляторе.
T DD 52	Содержание аккумулятора /результат/ переносится в двойное слово данных DD 52 .

9.5 Комбинированные вычислительные функции

Вычислительные функции можно комбинировать между собой. Для этого необходимо предварительно загрузить новое значение, над которым будет выполняться арифметическая операция, а затем выполнить следующую вычислительную функцию.

Пример:

Сумму значений слов данных DW 150, DW 151 и DW 152 требуется разделить на 3. В слове выходов AW 6 показать результат.

L DW 150	Значение слова данных DW 150 загружается в аккумулятор.
L DW 151	Значение слова данных DW 151 загружается в аккумулятор ; одновременно предыдущее содержание аккумулятора переносится во вспомогательный аккумулятор.
+F	Содержание обоих аккумуляторов суммируется. Сумма затем сохраняется в основном аккумуляторе.
L DW 152	Значение слова данных DW 152 загружается в аккумулятор ; одновременно предыдущее содержание аккумулятора переносится во вспомогательный аккумулятор.

+F	Содержание обоих аккумуляторов суммируется. Сумма всех значений находится теперь в основном аккумуляторе.
L KF +3	Значение 3 загружается в аккумулятор ; одновременно полученная ранее сумма переносится во вспомогательный аккумулятор.
:F	Находящаяся во вспомогательном аккумуляторе сумма делится на записанное в основном аккумуляторе значение 3. Результат затем остается в основном аккумуляторе.
T AW 6	В слове выходов AW 6 записывается вычисленное значение.

Во время этого вычислительного процесса аккумуляторы загружаются следующим образом:

	Аккумулятор /основной/ Akkumulator	Вспомогат. аккумулятор Hilfsakkumulator
L DW 150	<DW 150>	
L DW 151	<DW 151>	<DW 150>
+F	<DW 150+DW 151>	<DW 150>
L DW 152	<DW 152>	<DW 150+DW 151>
+F	<DW 150+DW 151+DW 152>	<DW 150+DW 151>
L KF +3	3	<DW 150+DW 151+DW 152>
:F	$<\frac{DW\ 150+DW\ 151+DW\ 152}{3}>$	<DW 150+DW 151+DW 152>
T AW 6	$<\frac{DW\ 150+DW\ 151+DW\ 152}{3}>$	<DW 150+DW 151+DW 152>

10 СТРУКТУРА ПРИКЛАДНОЙ ПРОГРАММЫ

Прикладные программы должны иметь определенную структуру, т.е. подразделяться на законченные фрагменты. Если каждому фрагменту присвоить определенную /технологическую/ функцию, то при программировании получатся наглядные части программы с простыми сопряжениями с другими частями программы. Эти преимущества программирования скажутся также во время пусконаладочных работ. Проверка программы по фрагментам особых трудностей не представляет, упрощается и обнаружение сбоев.

В устройствах автоматизации S5-110S, S5-130W, S5-l50A, S5-150K и S5-150S разбивка программы на структуры опирается на. процессор. Отдельные фрагменты программы называются "блоками". В зависимости от их назначения различаются следующие виды блоков:

- организационные блоки /раздел 10.2/,
- программные блоки /раздел 10.3/,
- функциональные блоки /раздел 10.4/
- шаговые блоки /раздел 10.5/ и
- блоки данных /раздел 10.6/

Чтобы обработать эти блоки, их нужно "вызвать". Это делается с помощью абсолютных или обусловленных логическим результатом операций вызова, предусмотренных программой. При выполнении операции вызова линейная обработка программы прекращается и происходит обработка программы в вызванном блоке.

Окончание блока указывается операцией "конец блока" /см.раздел 10.7/. Эта операция может программироваться абсолютно или в зависимости от логического результата. При выполнении операции окончания блока обработка программы продолжается в том блоке, где находился вызов только что обработанного блока. Обработка продолжается с команд, стоящих после этого вызова /см.раздел 10.1/.

10.1 Обработка программ

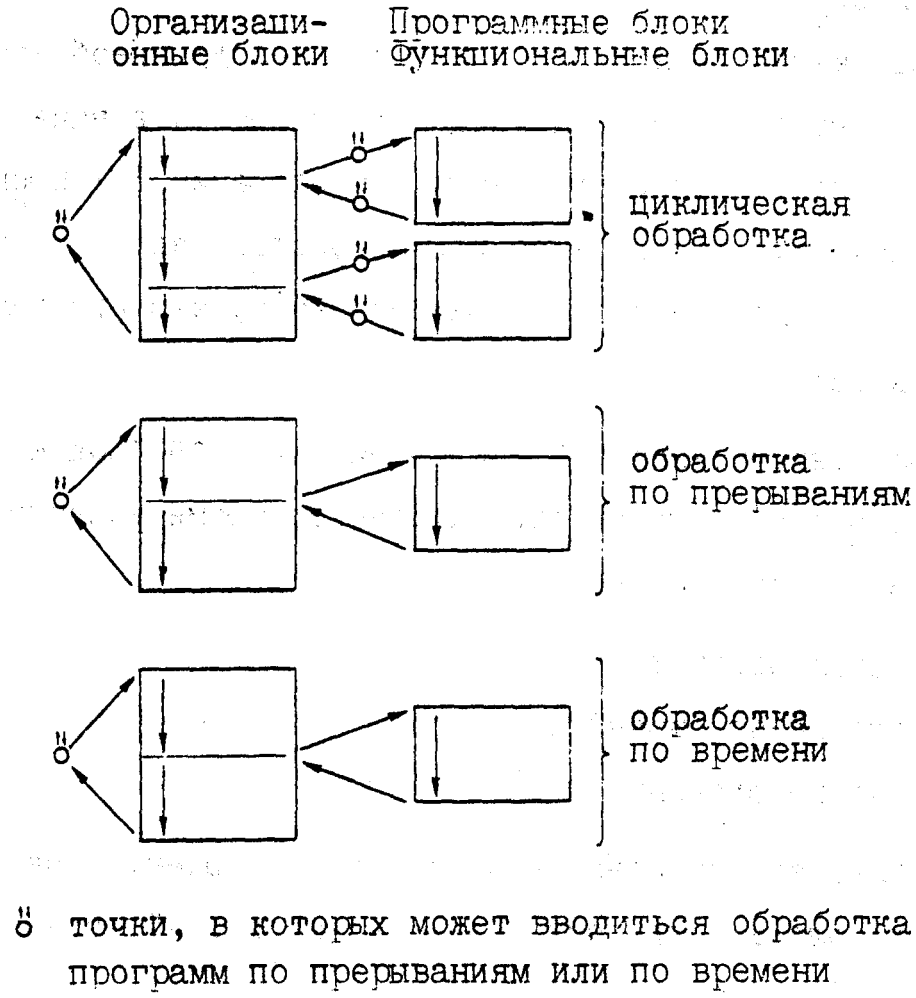
Прикладные программы могут обрабатываться различными способами. В устройствах автоматизации S5-110A, S5-130A и S5-130K преобладает циклическая обработка программ. Прикладная программа в начале обрабатывается линейно. В конце программы /на операции BE/ происходит возврат на начало. Однако в обоих устройствах автоматизации имеется возможность досрочного возвращения на начало программы /см. разделы 11 и 12/.

В устройствах автоматизации S5110S, S5-130W, S5-150A, S5-150K и S5-150S прикладная программа может обрабатываться тремя различными способами /см. рис. на стр. /:

- циклическая обработка,
- обработка по прерываниям и
- обработка по времени.

Обычно преобладает циклическая обработка программ. В местах перехода. с одного блока к другому может "вставляться" обработка по прерыванию или по времени.

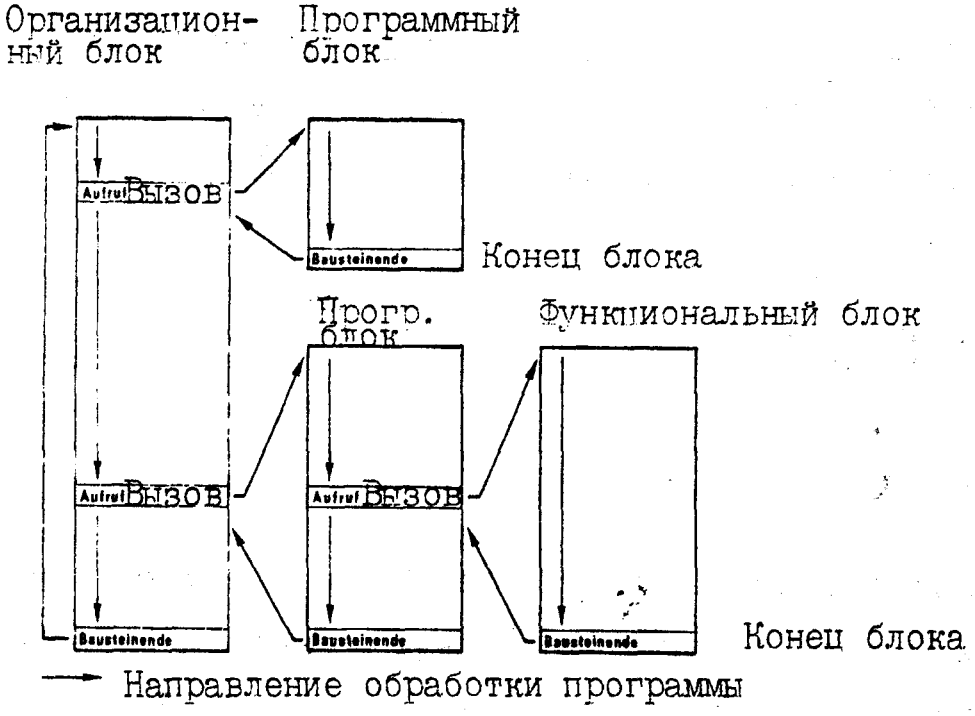
Обработка прикладных программ



Суперпозиционирование / вложение/ блоков

Обработкой программ различными способами управляет системная программа. Местом сопряжения с прикладной программой для каждого способа обработки являются организационные блоки. В эти организационных блоках, которые вызываются системной программой независимо от прикладной программы, запрограммированы операции вызова блоков на обработку. Блоки обрабатываются в порядке вызовов.

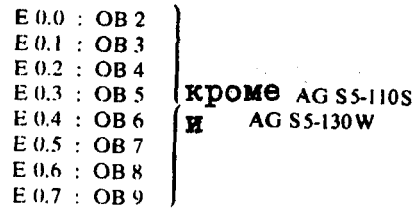
Из этих блоков в свою очередь можно вызывать другие блоки. Блоки могут "вкладываться" один в другой. Глубина вложения /суперпозиционирования/ составляет в устройствах S5-110S, S5-130W, S5-150A и S5-150K -7 блоков, а в устройстве S5-150S - 15 блоков. Это касается как программных, так и функциональных блоков /см.рисунок/. Блоки данных, в отличие от блоков программ, например, не обрабатываются. Они представляют собой зоны данных, обращение к каждой из которых происходит через операцию вызова. Операции с областью операндов "дачные" тот час же обращаются обратно в только что открытую /"свою"/ зону данных.



10.2 Организационные блоки

В организационных блоках 0В находится организация прикладной программы в норме перечня вызовов блоков. Различают организационные блоки для циклической обработки программ, для обработки по прерываниями, для обработки с управлением по времени. Организация *циклически обрабатываемой програ*ммы находится в организационном блоке ОВ1. ОВ1 вновь вызывается и обрабатывается системной программой сразу же после прохождения прикладной программы. Таким образом, блоки, вызываемые в организационном блоке ОВ1, обрабатываются по замкнутому циклу.

Организацией программы с *управлением по прерываниям* занимаются организационные блоки ОВ 2 - ОВ 9. Каждый из этих организационных блоков присвоен определенному биту входного байта ЕВ 0:



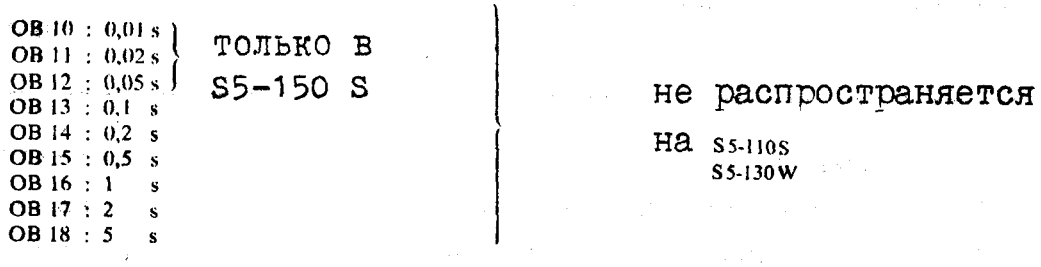
Обращение к отдельным битам входного байта ЕВ 0 происходит через специальные блоки "цифрового ввода со сборным сигналом". Системная программа при каждой смене блоков этого входного байта ЕВ 0 опрашивает о происшедших изменениях /через LPB0 непосредственно на периферии процесса/. При смене состояния бита с «0» на «1» соответствующий организационный блок прогоняется один единственный раз.

Эта обработка по прерываниям "вставляется" в циклическую обработку. Т.е. циклическая обработка прикладной программы по тревоге прерывается в месте вызова или окончания блока. После этого обрабатывается организационный блок обработки по прерываниям. После обработки

этого блока, а также блоков, вызванных внутри этого блока, процессор продолжает работу по циклической программа с того места, где произошло прерывание.

Вызовы для обработка по прерываниям являются приоритетными. Наивысший приоритет имеют вход E 0.0 и тем самым организационный блок 0B 2, а вход E 0.7 и организационный блок OB 9 - наименьший.

Организацией программы с управлением по времени занимаются организационные блоки OB 10 - OB 18. Каждому организационному блоку присвоена определенная цена времени:



Организационные блоки вызываются системной программой с соответствующими им интервалами времени. Т.е. находящаяся в этих организационных блоках программа обрабатывается с определенными интервалами. Например, вызов программного блока, находящийся в организационном блоке OB 13, выполняется один раз в 100 миллисекунд.

Эта обработка по времени "вставляется" в циклическую обработку. Циклическая обработка прерывается и обрабатывается соответствующий организационный блок. После обработки этого блока, а также блоков, вызванных внутри этого блока, процессор продолжает работу по циклической программе с того места, где произошло прерывание.

При "одновременном" появлении обработки по прерываниям /тревогам/ и обработки по времени сначала при вызове блока иди окончании блока вводится обработка по времени. Если во время этой обработки возникают границы блоков /вызовы блоков или окончания блоков/, то в этих местах вставляется обработка по прерываниям /тревогам/. Последняя доводится до конца, затем продолжается обработка программы, управляемая по времени. После окончания обработки по времени с места прерывания доводится до конца циклическая обработка программы.

Организационные блоки с 0В 19 по 0В 31 предусмотрены для сигналов от программного обеспечения, касающихся сбоев аппаратуры и пусковых режимов.

Организац. блок	Причина вызова	Реакция при незапрограммированном блоке
ОБ I9	Вызван незагруженный блок	никакой
ОВ 20	Новый запуск вручную	никакой
ОВ 2I	Вторичный запуск вручную	никакой
ОВ 22	Авт.повторный запуск после перерыва питания	никакой
ОВ 23	Задержка в квитировании при единичном обращении к блокам периферии	никакой
ОВ 24	Задержка в квитировании при актуализации отображения процесса	никакой
ОВ 25	Ошибка в адресе	стоп
ОВ 26	Превышено время цикла	стоп
ОВ 27	Ошибка в замещении	стоп
ОВ 28	Задержка в квитировании при байте входов ЕВ О	стоп
ОВ 29	свободны	
ОВ 30		
ОВ 3I		

Организационные блоки могут располагать индивидуальной программой. Если они не запрограммированы, то программное обеспечение или никак не реагирует, или устройство автоматизации /контроллер/ переключается на “СТОП" /см.таблицу выше/. Если не хотят, чтобы при одном из вышеназванных сбоев контроллер останавливался, то достаточно занять соответствующий блок командой ВЕ - конец программы.

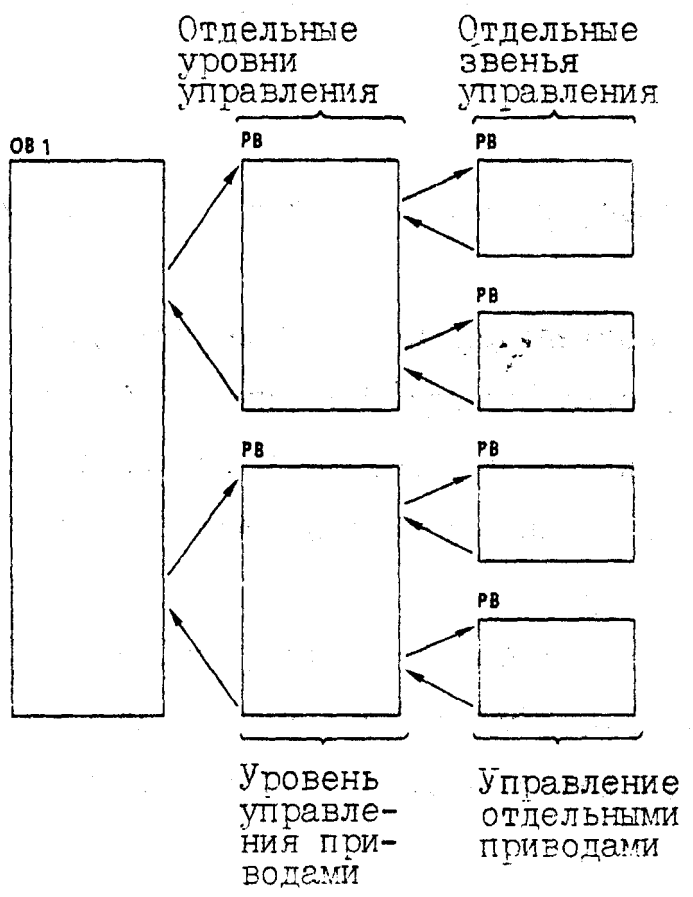
Программируемый в организационных блоках объем функций охватывает все описываемые в этом томе основное Функции языка программирования STEP 5.

10.3 Программные блоки

Блоки, которые получаются в результате членения прикладной программы, называют программными блоками РВ. Распределение старших блоков программы по целевому назначению дает обзорную картину прикладной программы. А в соответствующих подчиненных программных блоках обобщаются все технологически взаимосвязанные функции, например, Функции исполнительного механизма.

Программируемый в программных блоках объем функций охватывает все описываемые в этом томе основные функции языка программирования STEP 5.

Пример разбивки программы на блоки



Абсолютный вызов программного блока
SPA PB 7

Абсолютный вызов программного блока выполняется независимо от каких-либо условий. Циклическая обработка программы прерывается и затем продолжается в начале вызванного программного блока /в данном случае PB 7/. Логический результат сохраняется.

Условный вызов программного блока
SPB PB 7

Условный вызов программного блока выполняется в зависимости от логического результата. При результате «1» условный вызов рассматривается как абсолютный. При результате «0» вызов не учитывается и циклическая обработка программы продолжается. Логический результат, если он прежде равнялся «0», устанавливается в состояние «1».

10.4 Функциональные блоки

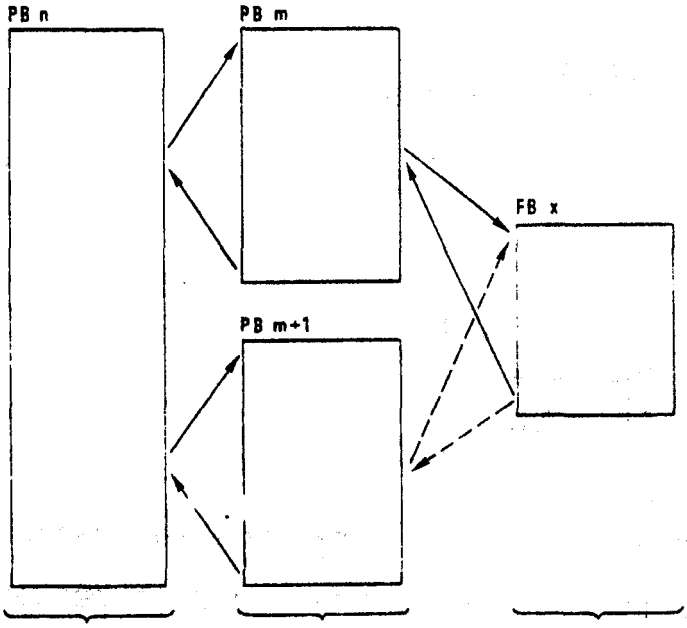
С помощью *функциональных блоков* реализуются часто повторяющиеся или очень сложные Функции.

функциональный блок (FB) представляет собой последовательность операций, которая описывает ограниченную Функцию. В памяти Функциональный блок записывается только один раз и вызывается старшими блоками /программными или функциональными/ один или несколько раз.

Вызов функционального блока может быть "параметрированным", за исключением устройств S5-110S, S5-130W. Ему могут быть приданы операнды, с которыми должен обрабатываться функциональный блок. Благодаря этому Функциональные блоки можно использовать произвольно. Внутренняя обработка Функциональных блоков и внутреннее программирование описываются в 3 томе.

Пользователь может или сам программировать функциональные блоки, или получать их от Фирмы СИМЕНС. Эти предоставляемые фирмой функциональные блоки называются "стандартными функциональными блоками". Они рассчитаны на общие случаи применения и описываются во втором томе.

Применение функциональных блоков



Отдельный уровень управления	Отдельное звено управления	Отдельный элемент управления
Собрание нескольких звеньев управления. Программа для старших функций, как например, контроль и управление сигнализацией.	Снабжение функциональных блоков параметрами /"параметрирование"/.	Функция отдельного звена управления, используемая многократно.

Абсолютный вызов функционального блока

Функциональная схема

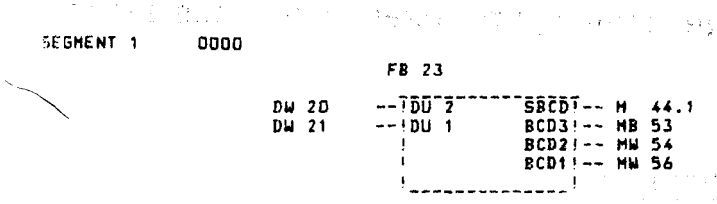
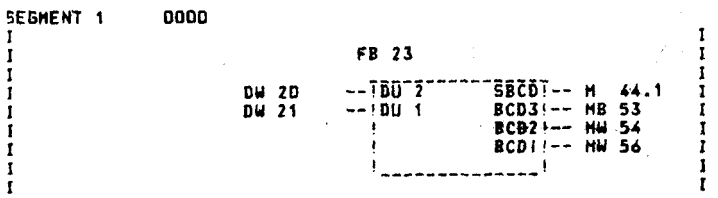


Таблица команд

```
SEGMENT 1
0000       :SPA FB23
0001 NAME  :COD:32
0002 DU 2  :    DW20
0003 DU 1  :    DW21
0004 SBCD  :    M 44.1
0005 BCD3  :    MB53
0006 BCD2  :    MW54
0007 BCD1  :    MW56
0008       :***
```

Абсолютный вызов функционального блока

Контактная схема



Абсолютный вызов функционального блока выполняется независимо от каких-либо условий. Циклическая обработка программы прерывается и продолжается в начале вызванного функционального блока /в данном случае блока FB 23 /. Логический результат" сохраняется.

После вызова функционального блока указываются операнды, с которыми должен обрабатываться функциональный блок /параметры блока/. Задание обозначений параметров блока, производится через программирующее устройство.

Разрешается также программирование функциональных блоков без параметров блоков.

Условный вызов функционального блока

```
SEGMENT 2     0009

                          FB 23
                            !
                  EW 36   --!DU 2       SBCD!-- A  25.0
                  EW 38   --!DU 1       BCD3!-- AB 26
                            !           BCD2!-- AW 27
                            !           BCD1!-- AW 29
                            !--------------!
```

Функциональная схема

```
SEGMENT 2
0009      :SPB FB23
000A NAME :COD:32
000B DU 2 :    EW36
000C DU 1 :    EW38
000D SBCD :    A 25.0
000E BCD3 :    AB26
000F BCD2 :    AW27
0010 BCD1 :    AW29
0011      :***
```

Условный вызов
функционального блока

Контактная схема

```
SEGMENT 2     0009
I                                                    I
I                          FB 23                     I
I                            !                       I
I                  EW 36   --!DU 2       SBCD!-- A  25.0 I
I                  EW 38   --!DU 1       BCD3!-- AB 26   I
I                            !           BCD2!-- AW 27   I
I                            !           BCD1!-- AW 29   I
I                            !--------------!        I
I                                                    I
```

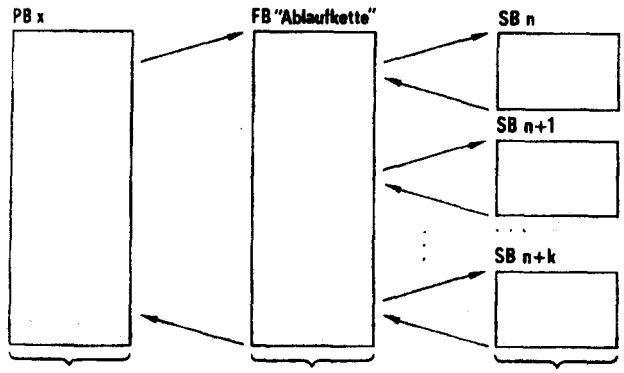
Условный вызов Функционального блока выполняется в зависимости от результата логической операции. При результате «1» условный вызов рассматривается как абсолютный. При результате «0» вызов игнорируется и продолжается циклическая обработка программы. Логический результат, если он раньше равнялся «0», устанавливается в состояние «1».

После вызова функционального блока указываются операнды, с которыми должен работать функциональный блок /параметры блока/. Задание обозначений параметров блока производится через программирующее устройство.

Разрешается также программирование функциональных блоков без параметров.

10.5 Шаговые блоки

Шаговые блоки служат для представления отдельных шагов в сблокированных системах управления. Один шаговый блок соответствует одному шагу в сблокированной системе. Он содержит команды для выходов и условия для выполнения следующего шага. Шаговые блоки, как правило, используются только в связи с каким-либо функциональным блоком "сблокированной цепочки", напр., со стандартными функциональными блоками FB70 ABL:MAST или FB71 ABL:VERF. в случае необходимости шаговые блоки вызываются этими функциональными блоками. Однако их можно вызывать и отдельно и тогда они будут вести себя как программные блоки.



Абсолютный вызов шагового блока

SPA SB 19

Абсолютный вызов шагового блока выполняется независимо от каких-либо условий. Циклическая обработка программы прерывается и продолжается в начале вызванного шагового блока /в данном случае – SB19 /. Логический результат сохраняется.

Условный вызов шагового блока

SPB SB 19

Условный вызов шагового блока выполняется в зависимости от логического результата. При результате «1» условный вызов обрабатывается как абсолютный, результат «0» не учитывается и продолжается циклическая обработка программы. Логический результат, если он раньше равнялся «0», становится «1».

10.6 Блоки данных

В блоках дачных DB находятся данные, с которыми работает прикладная программа. Один блок данных включает 256 слов данных. Если объем этого слова недостаточен, блок данных заменяется. Происходит вызов нового блока данных. Все операции с признаками операндов D будут возвращаться в этот блок данных.

Вызов блока данных

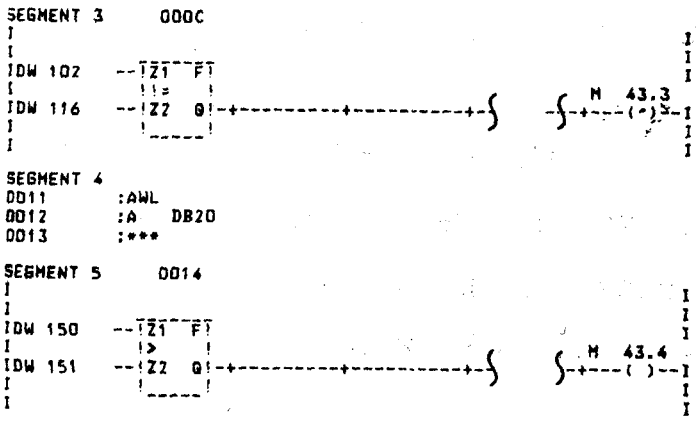
Функциональная схема

```
SEGMENT 3      0000
 DW 102    --!Z1  F!
             !!=   !
 DW 116    --!Z2  Q!- M  43.3
             !_____!

SEGMENT 4
0011       :AWL
0012       :A   DB20
0013       :***

SEGMENT 5      0014
 DW 150    --!Z1  F!
             !>    !
 DW 151    --!Z2  Q!- M  43.4
             !_____!
```

Таблица команд

```
SEGMENT 3
0012       :L    DW102
0013       :L    DW116
0014       :!=F
0015       :=    M 43.3
0016       :***

SEGMENT 4
0017       :AWL
0018       :A    DB20
0019       :***

SEGMENT 5
001A       :L    DW150
001B       :L    DW151
001C       :>F
001D       :=    M 43.4
001E       :***
```

Контактная схема



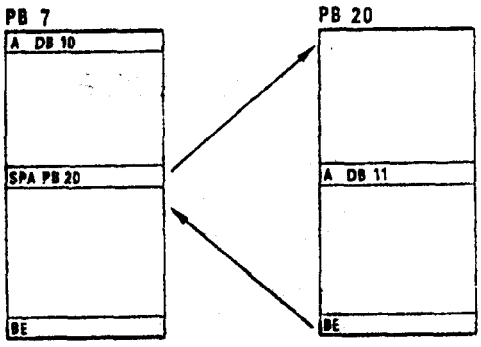
Абсолютный вызов блока данных выполняется независимо от каких-либо условий. Все адресуемые далее данные относятся к этому блоку /в рассматриваемом случае DB 20/. Циклическая обработка программы не прерывается. Логический результат и содержание аккумулятора сохраняются.

При сравнении между собой двух слов данных, стоящих в разных блоках данных, содержание одного слова данных можно загрузить, например, в слово меток. Прямое программирование смены блока данных возможно только в табличном виде.

Например:

```
:AWL
:L    DW 34
:A    DB 23
:L    DW 103
:!=F
:=    M  43.0
:***
```

Обработка программы с блоком данных



В программном блоке PB 7 происходит вызов блока данных DB10. Затем следует обработка данных этого блока. При вызове программного блока PB 20 в стеке записывается не только адрес, откуда происходит переход, но и действительная для этого адреса область данных /в данном случае DB10/.

Теперь будет обрабатываться программный блок PB 20. Однако блок данных DB10 по-прежнему действителен. Смена области данных произойдет только лишь с вызовом блока данных DB II, который будет действовать до конца программного блока PB 20.

При смене блока обратно на программный блок PB 7 из стека будет вызван не только адрес, откуда производился переход, но и дополнительно записанный блок данных DB10. Теперь в дальнейшей обработке блока PB 7 будет действовать блок данных DB10. Блок данных DB11 имел в программном блоке PB 20,так сказать, "локальное" значение.

10.7 Операции "Конец блока"

Оперении "Конец блока" завершают обработку блока независимо или в зависимости от результата логической операции. После этого в старшем блоке продолжается обработка программы. После завершения организационного блока процессор продолжает работу в системной программе. Конец блока BE.

Обрабатываемой в настоящее время блок заканчивается. Происходит обратный скачок в предыдущий блок, где находился вызов. Обработка программы продолжается в первой ячейке памяти после вызова блока /в функциональных блоках к вызову блока относятся и параметры блока/. Логический результат сохраняется. В устройствах автоматизации S5-110A, S5-130A, S5-130K происходит скачок на начало программы.

Условный конец блока

BEB

При логическом результате «1» операция BEB выполняется как BE. При результате «0» эта операция не выполняется. Логический результат, если он раньше равнялся «0», устанавливается в состояние «1». На содержимое аккумулятора эта операция не влияет. В устройствах автоматизации S5-110A, S5-130A, происходит /условный/ скачок на начало программы.

10.8 Стоп

STP

Команда STP вызывает: остановку процессора. После остановки процессора вызванной таким образом, повторный запуск можно произвести только вручную. Команда - STP применяется, например, чтобы как можно скорее вывести устройство управления в пассивное, безопасное состояние в случае программного обнаружения аппаратного сбоя. STP может программироваться везде. Исполнение этой команды не зависит от результата логической операции.

11.ПРОГРАММИРОВАНИЕ УСТРОЙСТВА АВТОМАТИЗАЦИИ S5-110A

В данной главе рассматривается запас операций, которым располагает устройство автоматизации S5-110A, а также некоторые особенности программирования. В разделе 11.1 комментируется перечень операций' для этого аппарата. Непосредственно программируемые с помощью устройства S5-110A логические операции приводятся в разделе 11.2.

При программировании Функций памяти необходимо учитывать, что устройство автоматизации S5-110A не обладает возможностью отображать процесс. Значение этого факта раскрывается в разделе 11.3.

Раздел 11.4 содержит руководство по запуску и опросу функций времени.

В разделе 11.5 показано программирование двоичных чисел, построенное на пересчетной схеме из раздела 3.6, а в разделе 11.6 - программирование двоично-десятичного счетчика.

В разделе 11.7 показана возможность сокращения времени реакции некоторых входов контроллера S5-110A /входы прерываний по сигналам тревоги/. Особенности обработки метки M 0.0 - влияние на выполнение операции - описываются в разделе 11.8. В разделе 11.9 поясняется процесс завершения программы на языке STEP5 или досрочного перехода на начало программы.

Реализация сблокированных систем управления /связанных процессов/ описывается в разделе 12.8 вместе с устройствами 130-A и 130-K. Такое программирование можно использовать также и в контроллерах 110-A.

Общие замечания

Устройство автоматизации 110-A в полном виде располагает 32 местами с разъемами, куда можно вставлять блоки ввода/ вывода и таймеров. Одно место с разъемом можно занимать по выбору 8 входами, 8 выходами или 4 таймерами, или 8 таймерами/счетчиками. Обращение к каждому месту производится через адрес байта, а к отдельному входу или выходу - через адрес бита. Адреса блоков в устройстве 110-A кодируются по местам с разъемами.

II.I Перечень операций устройства S5-110 A

Логические операции	
U -	операция И, опрос на состояние "I"
O -	операция ИЛИ, опрос на состояние "I"
UN -	операция И, опрос на состояние "0"
ON -	операция ИЛИ, опрос на состояние "0"
- E 0.0-15.7	с входом
- A 0.0-15.7	с выходом
- A 16.0-63.7	с памятью выхода
- M 0.0-63,7	с меткой

Функции памяти	
S -	установка
R -	сброс
= -	присвоение результата
- A 0.0-15.7	выхода
- A 16.0-63.7	меток выхода
- M 0.0-63.7	меток

Организационные функции	
BE	конец блока /конец программы/
BEB	конец блока /конец программы/ условный
NOP 0	нулевая операция
NOP 1	нулевая операция

UM 0.0 служит как опрос "прерываний" для скачка на начало программы.

SM 0.0 и RM 0.0 влияют на выполнение операции.

Управление и опрос таймеров производится через входы и выходы.

Примечание:

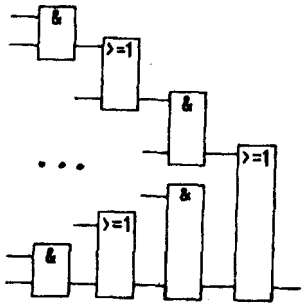
Начиная с блока центрального процессора 6ES5 900-7AD21 , дополнительно появились следующие операции:

U(	**функция И перед выражением в скобках**
)	**закрыть скобку**
O	**функция ИЛИ над операциями И**

11.2 Логические операции

Логические операции, которые можно программировать непосредственно, начиная с блока центрального процессора 6ES5900-7AD21 , выглядят следующим образом:

Функциональная схема



Начав с функции И, можно в последовательности ИЛИ-И-ИЛИ подсоединить еще 3 следующих уровня. Начало с функции ИЛИ допускает подсоединение только 2 уровней в последовательности И-ИЛИ. Внутри этих уровней допускается произвольная комбинация функций И и ИЛИ без необходимости устанавливать промежуточные метки.

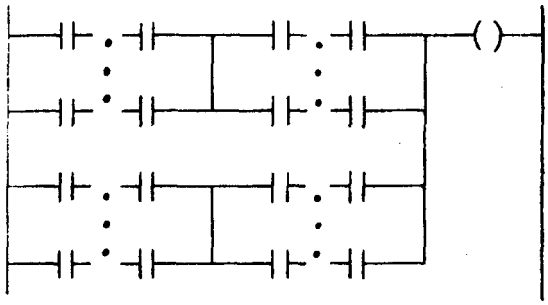
Таблица команд

```
U(      ┐
U   E   │
U   E   │  выражение в скобках
O   E   │
)       ┘
U   E
O
U(      ┐
O   E   │  выражение в скобках
O   E   │
)       ┘
U   E
=   A
```

В табличном виде каждое выражение в скобках можно программировать без суперпозиционирования скобок. Количество выражений в скобках, следующих одно за другим в одной логической операции, может быть произвольным.

Контактная схема

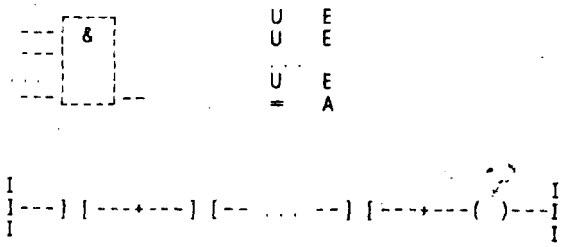
Контактная схема допускает непосредственное программирование следующих структур:



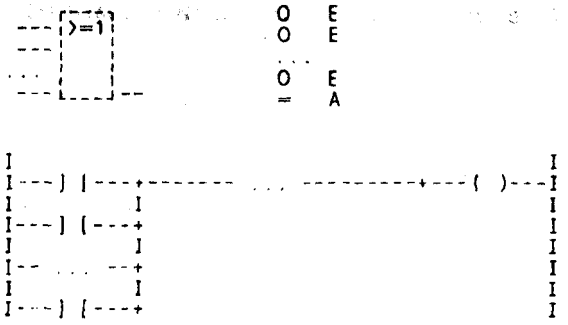
При всех остальных логических операциях должны устанавливаться метки для запоминания и последующего опроса промежуточных результатов. Эти метки промежуточных результатов могут использоваться многократно /см.раздел 3.3/.

Ранее в устройствах автоматизации 110-А непосредственное программирование допускали следующие операции:

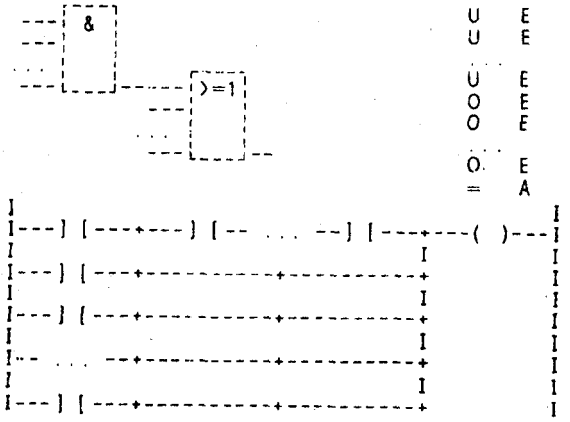
- функция И с произвольно многими входами в соответствии с последовательной схемой произвольно многих контактов



- функция ИЛИ с произвольно многими входами в соответствии с параллельной схемой включения произвольно многих контактов



- функция И-перед-ИЛИ, если выход функции И ведет на первый вход функции ИЛИ в соответствии с параллельной схемой включения контактов, если в главной ветви схемы контакты включены последовательно.

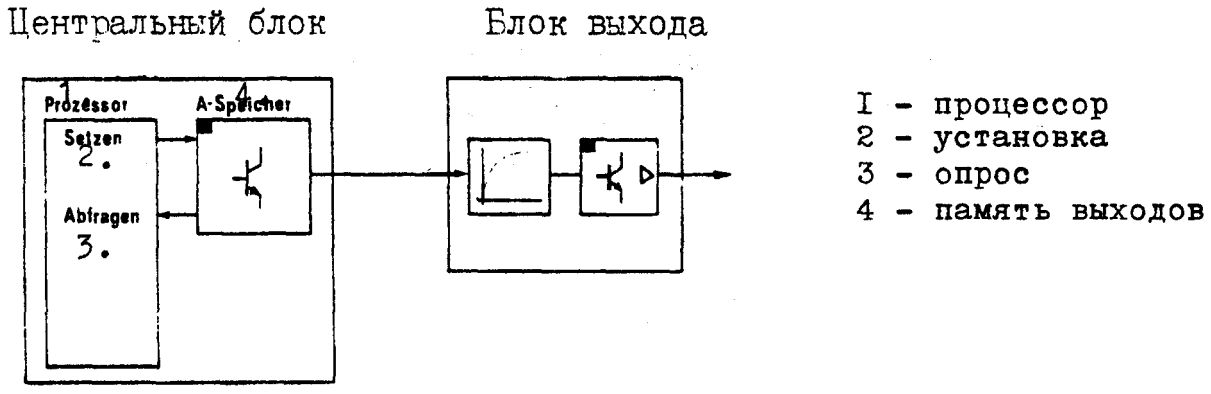


Во всех остальных логических операциях для запоминания и последующего опроса промежуточных результатов должны устанавливаться метки. Эти метки промежуточных результатов могут использоваться многократно /см.раздел 3.3/.

11.3 Функции памяти

Функции памяти в устройстве автоматизации S5-110A используются так же. как и в других устройствах.

При управлении выходами, следует помнить, что S5-110A не обладает возможностью отображать процесс. Структурная схема аппаратного обеспечения выглядит следующим образом:



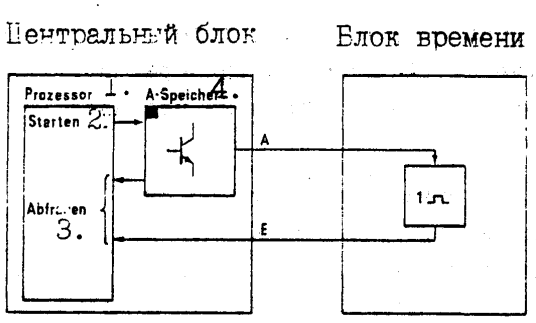
Если в программе имеется обращение к выходам, процессор работает только С памятью выходов, которая находится в центральном блоке. Память выходов производит последующую установку или сброс выходов с задержкой. Если. например, в команде предусматривается установка выхода, то в следующей команде сразу же можно производить его опрос. Однако состояние сигнала на выходных клеммах блоке выхода изменится примерно через 0,3 мс.

Этот период соответствует времени обработки примерно 20 команд. Если внутри этих 20 команд вновь произойдет изменение состояния сигнала, то это изменение не окажет никакого влияния на сигнал на выходных клеммах /выход не "дребезжит"/. Таким образом, через последовательность команд можно программировать, как и при отображении процесса, установку или сброс по приоритетам, при условии что расстояние между командами установки или сброса не превышает 20 команд.

Если команды на установку или сброс для одного выхода удалены друг от друга на большее расстояние, нужно предупредить одновременное исполнение обеих команд.

11.4 Функции времени

С помощью устройства автоматизации S5-110A можно также реализовывать Функции времени. Так как - область операндов "время T" в объем операций не входит, установка звеньев времени в блоках времени производится с помощью операционного кода для выходов, а опрос производится с помощью операционного кода для входов. Структурно это выглядит следующим образом:



1. Процессор
2. Запуск
3. Опрос
4. ЗУ выходов

Блок таймеров 380

Блок содержит 4 таймера, которые запускаются и опрашиваются битами с № 0 по № 3. Остальные биты этого разъема использовать не разрешается.

Если, к примеру, блок таймеров занимает 7-й разъем, то обращение к находящимся в этом блоке таймерам будет происходить следующим образом:

Функция	1-я уставка	2-я уст.	3-я уст.	4-я уст.
Присвоение	= A 7.0	= A 7.1	= A 7.2	= A 7.3
Установка	S A 7.0	S A 7.1	S A 7.2	S A 7.3
Сброс	R A 7.0	R A 7.1	R A 7.2	R A 7.3
Опрос	U E 7.0	U E 7.1	U E 7.2	U E 7.3
	UN E 7.0	UN E 7.1	UN E 7.2	UN E 7.3
	O E 7.0	O E 7.1	O E 7.2	O E 7.3
	ON E 7.0	ON E 7.1	ON E 7.2	ON E 7.3

Блок таймеров/счетчиков 381

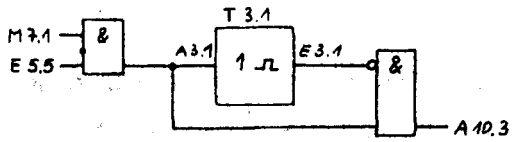
Блок таймеров/счетчиков содержит 8 таймеров, которые можно переключать на выполнение функции счета. Обращение к таймерам происходит также как и в блоке 380, с единственной разницей, что здесь под таймеры отведены все 8 битов.

Счетчики этого блока являются счетчиками обратного счета, подсчет в которых производится от приходящего извне сигнала. Внешне установленный параметр счета принимается счетчиком как задание как только происходит сброс выхода, соответствующего этому счетчику /состояние «1» при RA и «0» - при = A/. Счет в счетчике "разрешается", если соответствующий выход имеет состояние «1» /состояние «1» при SA или * A/. Опрос входа по функции И ив дает сигнал «1», если содержание счетчика больше нуля. При состоянии счетчика «0» опрос UNE даст «1».

Пример программирования задержки включения

Выход A 10.3 должен включаться с выдержкой 5 секунд, если на входе E 5.5 и метке M7.1 одновременно будет состояние «1». Нужно использовать 2-й таймер 3 разъема.

Программа:



Второй таймер на разъеме 3 /T3.1/ запускается через выход A 3.1. На блоке устанавливается задание времени. После истечения этого времени (UN E3.1) и при условии, что на входе таймера все еще состояние "1"(U A3.1), произойдет установка выхода A 10.3.

Таблица команд:

U	E	5.5		
U	M	7.1		
-	A	3.1	Starten der Zeit	Запуск таймера
UN	E	3.1	Ist die Zeit abgelaufen?	Время истекло?
U	A	3.1	Steht Signalzustand „1" noch an?	Сохраняется ли состояние "I"?
-	A	10.3		

11.5 Двоичные счетчики

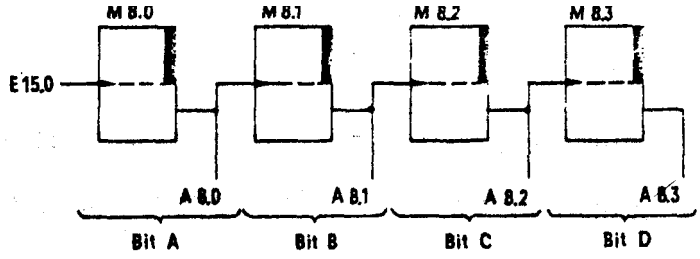
Счетное Функции строятся на основе триггеров со счетными входами. Использоваться могут как триггеры со вспомогательными метками /раздел 3.6, "Функциональнее схемы"/, так и триггеры с обработкой Фронтов /раздел 3.6, "Таблицы команд"/.

Двоичные счетчики со вспомогательными метками

При использовании триггеров со вспомогательными метками триггеры со счетными входами программируются последовательно, причем выход одного выводится на вход следующего триггера. Тогда счетчик при каждом падающем /заднем/ Фронте на входе отсчитывает одну единицу в прямом направлении. Если с входом следующего триггера связать вспомогательную метку, то получают счетчик обратного счета, который считает в обратном направлении при каждом заднем фронте на входе.

Структурное изображение 4-разрядного двоичного счетчика

построенного на триггерах со счетными входами со вспомогательными метками:



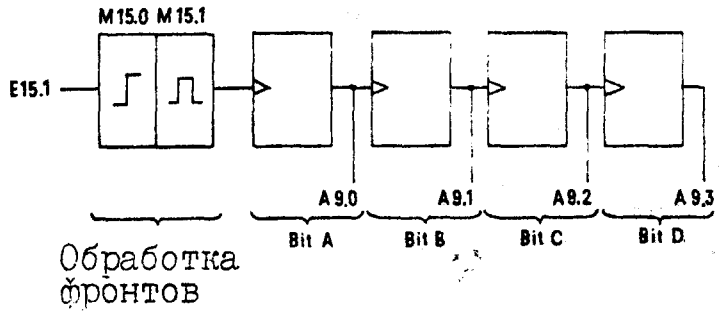
Программирование

```
U   E  15.0   HILFSMERKER   BIT A
UN  A   8.0   вспомог.
S   M   8.0   метка
U   E  15.0
U   A   8.0
R   M   8.0

UN  E  15.0   AUSGANG       BIT A
U   M   8.0   Выход
S   A   8.0
UN  E  15.0
UN  M   8.0
R   A   8.0

U   A   8.0   HILFSMERKER   BIT B
UN  A   8.1   вспомог.
S   M   8.1   метка
U   A   8.0
U   A   8.1
R   A   8.1

UN  A   8.0   AUSGANG       BIT B
U   M   8.1   выход
S   A   8.1
UN  A   8.0
UN  M   8.1
R   A   8.1

U   A   8.1   HILFSMERKER   BIT C
UN  A   8.2   вспомог.
S   M   8.2   метка
U   A   8.1
U   A   8.2
R   M   8.2

UN  A   8.1   AUSGANG       BIT C
U   M   8.2   выход
S   A   8.2
UN  A   8.1
UN  M   8.2
R   A   8.2

U   A   8.2   HILFSMERKER   BIT D
UN  A   8.3   вспомог.
S   M   8.3   метка
U   A   8.2
U   A   8.3
R   M   8.3

UN  A   8.2   AUSGANG       BIT D
U   M   8.3   выход
S   A   8.3
UN  A   8.2
UN  M   8.3
R   A   8.3
```

Двоичные счетчики с обработкой фронтов

При использовании триггеров с обработкой Фронтов обработку достаточно запрограммировать один единственный раз. Затем следует программирование отдельных битов счетчика, как это описано в разделе 3.6. Счетчик прямого счета будет вести отсчет при переднем фронте на счетном входе. Если при программировании поменять последовательность установок и сбросов выхода /сброс импульсной метки будет происходить синхронно со сбросом выхода/, то получится счетчик обратного счета. Если отсчет должен производиться по заднему фронту на входе, следует использовать обработку по заднему фронту сигнала.

Структурное изображение 4-разрядного двоичного счетчика
построенного на триггерах со счетными входами с обработкой фронтов



Программирование

```
U   E  15.1     FLANKENAUSWERTUNG
UN  M  15.0     обработка фронта
=   M  15.7

U   M  15.7
S   M  15.0
UN  E  15.1
R   M  15.0

U   M  15.7     BIT A
UN  A  9.0
S   A  9.0
R   M  15.7

                RUECKSETZEN DES IMPULSMERKERS
                сброс импульсной метки
U   M  15.7
U   A  9.0
R   A  9.0

U   M  15.7     BIT B
UN  A  9.1
S   A  9.1
R   M  15.7

                RUECKSETZEN DES IMPULSMERKERS
                сброс импульсной метки
U   M  15.7
U   A  9.1
R   A  9.1

U   M  15.7     BIT C
UN  A  9.2
S   A  9.2
R   M  15.7

                RUECKSETZEN DES IMPULSMERKERS
                сброс импульсной метки
U   M  15.7
U   A  9.2
R   A  9.2

U   M  15.7
UN  A  9.3      BIT D
S   A  9.3
R   M  15.7

                RUECKSETZEN DES IMPULSMERKERS
U   M  15.7      сброс импульсной метки
U   A  9.3
R   A  9.3
```

В счетчике обратного счета программирование для бита. А выглядит так:

```
U   M  15.7
U   A  9.0
R   A  9.0      RUECKSETZEN DES AUSGANGS        сброс выхода

R   M  15.7     RUECKSETZEN DES IMPULSMERKERS   сброс импульсной метки

U   M  15.7
UN  A  9.0
S   A  9.0      SETZEN DES AUSGANGS             установка выхода
```

Соответствующим образом программируются следующие биты.

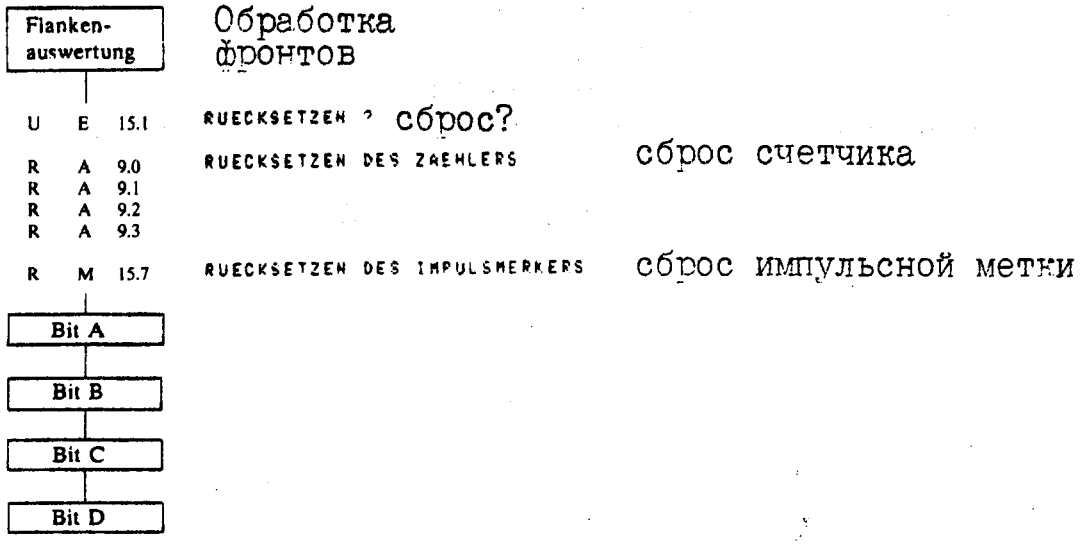
Счетчик можно построить также с помощью команд S M0.0 и R M0.0 /смотри раздел 11.8/. Программа тогда будет значительно короче.

```
U   E  15.1
UN  M  15.0
=   M  15.7
U   M  15.7   Flankenauswertung          Обработка фронтов
S   M  15.0
UN  E  15.1
R   M  15.0

UN  M  15.7   Nur bei einer Flanke       Счет ведется только при одном фронте
S   M  0.0    wird gezählt

UN  A  9.0
=   A  9.0   Bit A                      Бит A
S   M  0.0

UN  A  9.1
=   A  9.1   Bit B
S   M  0.0

UN  A  9.2
=   A  9.2   Bit C
S   M  0.0

UN  A  9.3   Bit D (letztes Bit)        Последний бит
=   A  9.3
R   M  0.0   Ende der Bearbeitungssperre Конец запрета на обработку
```

Сброс счетчика

Через вход E15.1 нужно сбросить только что запрограммированной счетчик. Для этого после обработки фронта опрашивается вход E15.1 на наличие сигнала «1». При сигнале «1» произойдет сброс выходов счетчика. Одновременно также произойдет сброс импульсной метки, что сделает невозможным ведение отсчета при появлении сигнала сброса. Таким образом, сброс статичен.

Программирование



Установка счетчика

Установка счетчика должна реагировать только на изменение Фронта сигнала. Через вход E15.2 на выходы A9.0 - A9.3 должны передаваться состояния сигналов на входах E9.0, E9.1, E9.2 и E9.3. Для этого на каждый выход программируется в соответствии с разделом 3.5 один триггер, причем обработка фронта потребуется только один единственный раз. Последовательность в программе для установки счетчика может программироваться перед программированием самого счетчика.

Программирование

```
U   E  15.2     Flankenauswertung
UN  M  15.2
=   M  15.7

U   M  15.7
S   M  15.2

UN  E  15.2
R   M  15.2

U   M  15.7     Bit A
U   E  9.0
S   M  9.0      Setzen

U   M  15.7
UN  E  9.0
R   M  9.0      Rücksetzen

U   M  15.7     Bit B
U   E  9.1
S   A  9.1      Setzen

U   M  15.7
UN  E  9.1
R   A  9.1      Rücksetzen

U   M  15.7     Bit C
U   E  9.2
S   A  9.2      Setzen

U   M  15.7
UN  E  9.2
R   A  9.2      Rücksetzen

U   M  15.7     Bit D
U   E  9.3
S   A  9.3      Setzen

U   M  15.7
UN  E  9.3
R   A  9.3      Rücksetzen
```

обработка фронтов

установка

сброс

установка

сброс

установка

сброс

установка

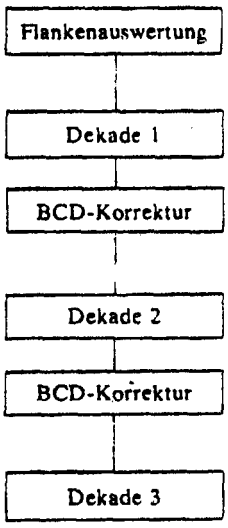
сброс

11.6 Двоично-десятичный счетчик

Счетчик с двоично-десятичным кодированием имеет такую же структуру как и двоичный счетчик. Дополнительно здесь после каждых 4 битов вводится двоично-десятичная коррекция.

Структура двоично-десятичного счетчика с 3 декадами,

построенного на триггерах со счетными входами с обработкой фронтов



обработка фронтов

I-я декада состоит из 4 битов, программируется как описано в разделе I0.5

двоично-десятичная коррекция

2-я декада состоит из 4 битов, программируется как описано в разделе I0.5

двоично-десятичная коррекция

3-я декада состоит из 4 битов, программируется как описано в разделе I0.5

С помощью двоично-десятичной коррекции при десятом импульсе каждой декады, т.е. если значение "10" стоит в соответствующей предыдущей декаде, происходит коррекция значения предыдущей декады на «0». Одновременно заново устанавливается импульсная метка. Она будет служить в качестве "переноса" в следующую декаду.

Программирование двоично-десятичной коррекции

BCD-Korrektur

```
U   A  9.1     опрос параметра I0
U   A  9.3
R   A  9.1     установка значения 0
R   A  9.3
-   M  15.7    установка импульсной метки /перенос/
```

11.7 Вход прерываний

С помощью блока " Цифровой ввод со сборным сигналом" можно реализовать "входы прерываний". Смена состояния сигнала с «0» на «1» на каком-либо входе этого блока сбрасывает в месте прерывания счетчик адресов в центральное устройстве. Циклическая обработка прерывается и продолжается в начале программы. Поэтому критичные по времени части программы следует писать в начале программа.

Место прерывания в программе, на которое должен действовать сигнал тревоги/ обозначается командой UM 0.0.При обработке управляющим устройством этой команды происходит опрос наличия сигнала прерывывания, а в случае необходимости - переход на начало программы.

Эта команда U M0.0не сопрягается с другими опросами. Она выполняется процессором, если сравнивать с логическими операциями, как функция NOP.

В отличие от операции BEB /см.следующий раздел/ команда U M0.0 выполняется только при смене состояния сигнала с "0" на "1" на сборной шине, т.е. один единственный раз. только лишь следующее изменение состояния сигнала вызовет вновь исполнение команды U M0.0

11.8 Запрет исполнения операции

С помощью команд SMO.0 /установить метку M 0.0/ и R M0.0 /сбросит метку M 0.0/ в устройстве автоматизации S5-110A. можно заблокировать исполнение операций:

- = присвоение
- S установка
- R Сброс

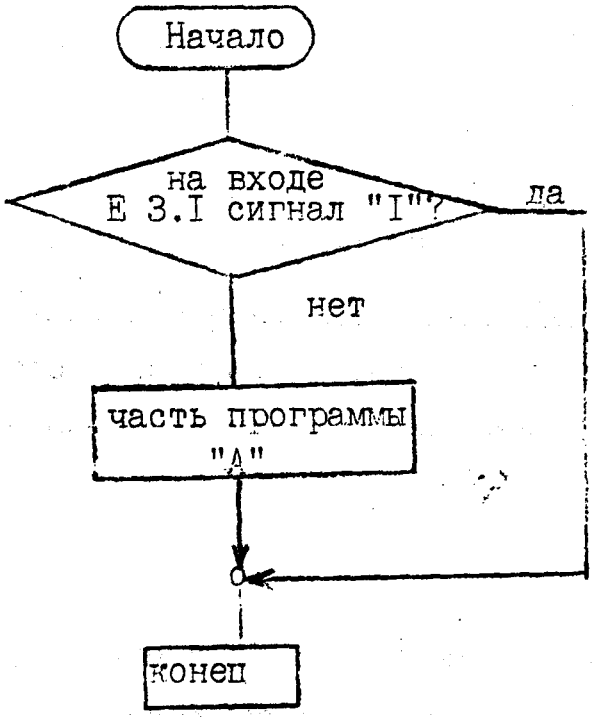
в зависимости от выходов A и меток M. Запрет действителен, если метка M 0.0 имеет состояние «1», т.е. если она установлена.

Метка. M0.0 устанавливается командой S M0.0 и логическим результатом «1». Все последующие команды обрабатываются, за исключением команд, содержащих операции присвоения /=/, установки /S/ и сброса /R/. Время обработки не изменяется. Командой R M0.0 можно снять запрет на исполнение операций. Если команда S M0.0 обрабатывается с логическим результатом «0», блокировки исполнения операций не происходит. Команды обрабатываются как обычно.

Пример:

При наличии на входе E 3.I сигнала "I" часть программы "A" обрабатываться не может.

Алгоритм



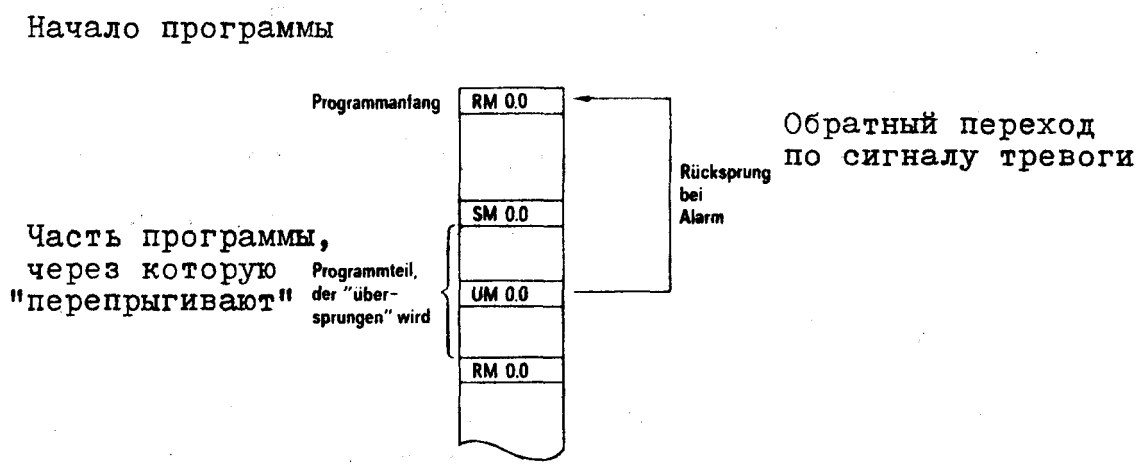
Выполнение программы

```
      U E 3.1
      S M 0.0
Часть программы "A"
      R M 0.0
```

Хотя эта часть программы все время обрабатывается /поэтому время цикла не изменяется/, выполняется она только при условии, если на входе E3.I состояние "0"

Примечание

Опрос сигналов прерываний /тревог/ UM 0.0 продолжает действовать также при наличии запрета на выполнение операции. В этом случае "запрещенная" часть программы покидается /без отмена запрета на исполнение операции, введенного R M0.0 / и происходит возврат на начало программы. Здесь тогда должна быть команда U M0.0 , которая обеспечит выполнение последующих операций.



Внутри "запрещенного" участка программы, т.е. между командами S M0.0 и R M0.0 не разрешается программировать команду BEB. Так как при запрете на выполнение операции /обработка команды S M0.0 по результату «1»/ состояние «1» сохраняется, то операция BEB выполнялась бы все время, что привело бы к "зацикливанию" программы.

Команды U M0.0, S M0.0 и R M0.0 не разрешается программировать непосредственно друг за другом. Как минимум, между этими командами должна стоять одна /другая/ команда STEP-5 .

11.9 Операции окончания блока

Программа в устройствах автоматизации S5-110A завершается операцией "конец блока".

Окончание блока BE

Операцией BE заканчивается обработка программы на языке STEP-5. Процессов устанавливает счетчик адресов на «0» и начинает обработку программы с ее начала.

Устройство программирования 630 пишет операцию BE в конце программы автоматически. Поэтому в этом устройстве ее вводить не нужно.

Условное окончание блока BEB

С помощью операции BEB можно преждевременно закончить обработку программы. Эта операция выполняется только при условии, что логический результат при обработке операции равен «1». Выполнение операции тогда будет таким же, как и при абсолютном окончании блока BE. При логическом «0» BEB не выполняется и продолжается обработка программы по циклу. Однако логический результат становится равным"1"

После операции BEB можно, например, непосредственно устанавливать или сбрасывать выход или метку. Если после BEB следует опрос, например, UE, то этот опрос является первичном; отсюда начинается новая логическая операция /см.раздел 2.3/.

В отличие от команды UM 0.0, которая при выполнении операции также вызывает переход на начало программы, операция .BEB статична, т.е. до тех пор, пока сохраняется результат «1», будет выполняться переход на начало программы. BEB может стоять в программе STEP-5 в любом месте и любое число раз.

Внимание!

Операцию BEB не разрешается программировать между командами SM 0.0 и RM 0.0 При запрете на выполнение операции результат после SM0.0 становится «1», в результате чего операция выполняется постоянно и программа зацикливается.

12 ПРОГРАММИРОВАНИЕ УСТРОЙСТВ АВТОМАТИЗАЦИИ S 5-130 A и S 5-130 K

Устройства автоматизации 130 А и 150 К отличаются друг от друга только конструктивным исполнением. Контроллер 130А имеет защищенное исполнение /платы помещены в металлические, капсулы/; контроллер 130 К имеет компактное исполнение /его платы размещены в стойке ES902. Оба устройства используют один и тот же микропроцессор, благодаря чему оба аппарата обладают одинаковым объемом операций. В этом разделе рассматривается запас операций этих устройств автоматизации, а также некоторые особенности программирования. В разделе 12.1 даны пояснения к операциям, выполняемым этими аппаратами. Непосредственно программируемые с помощью устройств 150 А и 150 К логические операции приведены в разделе 12.2. Функции памяти и отображение процесса на выходах описываются в разделе 12.3. Вместе с раскрытием функций времени в разделе 12.4 приводится программирование тактового генератора. Далее следует описание вычислительных функций /раздел 12.5/ и функций загрузки и переноса /раздел 12.6/. В разделе 12/7 описан блок таймеров/счетчиков и приведен пример индикации сбоя с помощью блока таймеров/счетчиков 390.

В разделе 12.8 раскрывается программирование сблокированных цепочек управления процессами. В последнем разделе описаны операции окончания блоков. Программирование пересчетных схем на триггерах со счетным входом и счетчиков описано в разделах 3.6, 11.5 и 11.6.

Общие замечания

Устройства автоматизации 130 А и К при полном развитии могут работать с 256 входами, 256 выходами, 64 таймерами и 16 счетчиками. Для выходов существует регистр отображения процесса, находящийся в центральном блоке. Опрос входов производится напрямую. Отображение процесса с помощью операции BE переносится в блоки выводов, т.е. обращение к блоку вывода происходит только в конце программы. С помощью операции TPB управление выходами можно сделать независимым от операции BE.

Адресование блоков ввода/вывода производится независимо от места блоков в стойке. Адресование выполняется в блоках.

12.1 Перечень операций устройства

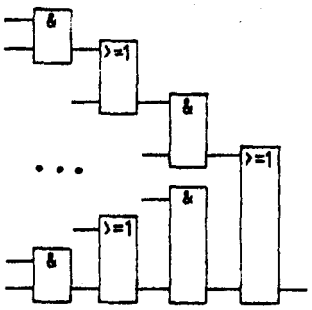
Логические операции			
U	-		И, опрос на состояние "I"
O	-		ИЛИ, опрос на состояние "I"
UN	-		И, опрос на состояние "0"
ON	-		ИЛИ, опрос на состояние "0"
	E	0.0-31.7	входа
	A	0.0-31.7	выхода
	A	32.0-47.7	ЗУ выхода
	M	0.0-63.7	метки
	T	0 -63	таймера
	Z	0 -15	счетчика
U(			И перед выражением в скобках
)			скобку закрыть
O			ИЛИ перед И
Функции памяти			
S	-		Установить
R	-		Сбросить
=	-		Присвоить
	A	0.0-31.7	выход
	A	32.0-47.7	ЗУ выхода
	M	0.0-63.7	метку
Функции времени			
SI	T	0 -63	Запуск таймера в режиме короткого импульса
SV	T	0 -63	Запуск таймера в режиме удлиненного импульса
R	T	0 -63	Сброс таймера
Функции счета			
ZR	Z	0 -15	Обратный счет
S	Z	0 -15	Установка счетчика
R	Z	0 -15	Сброс счетчика

Функции загрузки и переноса				
L	-			Загрузить
T	-			Перенести
	AB	0	-31	байт выхода
	MB	0	-31	байт мѐтки
	PB	0	-47	байт периферии
Организационные функции				
BL				Конец блока /конец программы/
BEB				Конец блока /конец программы/ условный

12.2 Логические операции

Программируемые непосредственно в устройстве автоматизации S5-130A логические операции можно описать следующим образом:

Функциональная_схема



Начав с функции И, можно добавить еще три следующих уровня в последовательности ИЛИ-И-ИЛИ. Если начать с функции ИЛИ, то добавить можно только два следующих уровня в последовательности И-ИЛИ. В пределах этих уровней допускается произвольное комбинирование функций И и ИЛИ без установки промежуточной метки.

Таблица команд

```
U(
U  E
U  E
O  E
)
U  E
O
U(
O  E
O  E
)
U  E
=  A
```

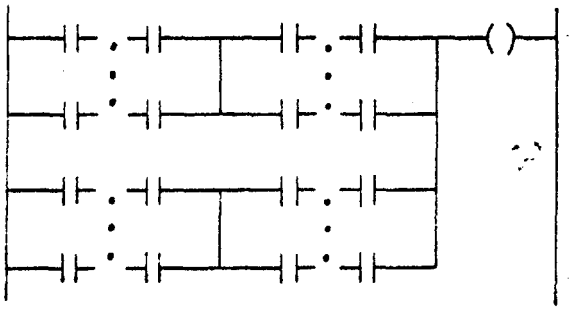
Выражение в скобках

Выражение в скобках

Программирование в виде таблицы команд позволяет программировать выражение в скобках без суперпозиционирования /наложения/ скобок. Количество выражений в скобках внутри логической операции может быть произвольным.

Контактная схема

Исходя из контактной схемы, можно напрямую запрограммировать следующую структуру:

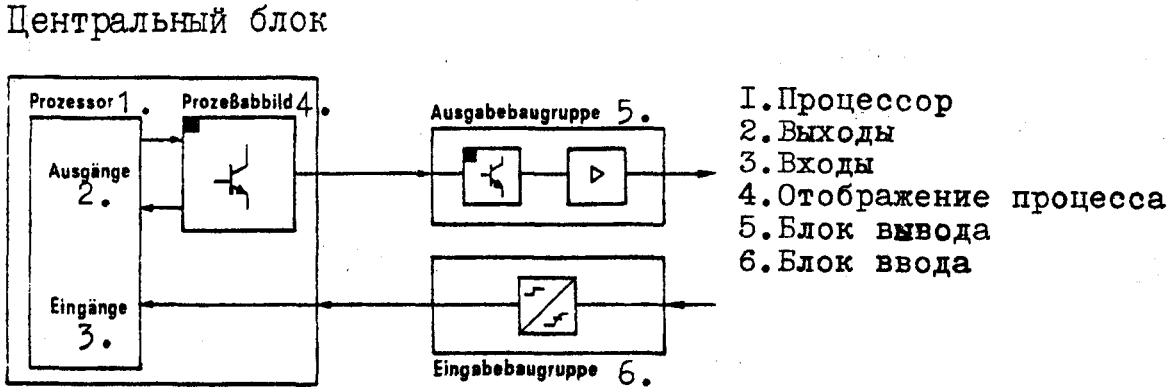


Во всех других логических операциях необходимо устанавливать метки, чтобы можно было запомнить промежуточные результаты и затем их опросить. Метками промежуточных результатов можно пользоваться многократно /см.раздел 3.3/.

12.3 Функции памяти

Функциями памяти в устройствах -автоматизации 130 А и К можно пользоваться также, как и в других устройствах.

Устройства автоматизации ІЗО А и К имеют регистр отображения процесса на выходах. Структурное изображение аппаратной части выглядит следующим образом:



Обработка прикладной программы ведется только на основе отображения процесса на выходах. В конце программы /при BE/ производится перенос отображения процесса на выходах в блоки вывода, т.е. полученные во время обработки программы состояния сигналов присваиваются блокам вывода. Смена состояния сигнала на клемме выхода может произойти только при обработке операции BE /исключение составляет случай прямого переноса в блоки вывода в обход отображения процесса, см.раздел 12.6/.

Наряду с отображением процесса на выходах, обращение к которому производится через операнды A 0.0 - A 31.7, имеется также область памяти с не сохраняющимися метками, к которой адресуются через операнды A 32.0 - A 63.7. Эта вышеназванная область обрабатывается только в двоичном виде.

В центральных блоках с объемом памяти 4 К не сохраняющиеся метки A 48.0 - A 63.7 являются метками фронтов для операции "Запуск таймера". Поэтому их нельзя использовать, если запрограммированы соответствующие таймеры.

12.4 Функции времени

В устройствах автоматизации 130 A и K функции времени реализуются в специальных блоках периферии /например, в блоке таймеров/счетчиков 390. Установки времени задаются в таймерах, запуск или сброс которых происходит в соответствии с программой. Опрашивать /числовое/ значение уставки времени с помощью программы STEP-5 нельзя. Однако его можно считывать через инженерную панель 391. В функциях времени, реализуемых в аналоговом виде, следует учитывать время восстановления готовности.

Пример программирования тактового генератора

Нужно, чтобы метка M49.0 изменяла свое состояние с определенной частотой. В качестве таймера берется время T17. Установка времени задается на блоке.

Чтобы таймер мог работать в качестве генератора такта, его нужно периодически запускать. Новый запуск происходит сразу же после истечения заданного времени. Вместе с истечением времени должно меняться состояние сигнала метки /принцип триггера со счетным входом/.

Решение:

```
UN  T  17
=   M  63.7   Impulsmerker                импульсная метка
SV  T  17     Starten der Zeit            запуск таймера

U   M  63.7   Binäruntersetzer            триггер со счетным входом
UN  M  49.0
S   M  49.0     Setzen                    установка
R   M  63.7     Rücksetzen des Impulsgebers  сброс импульсной метки

U   M  63.7
U   M  49.0
R   M  49.0     Rücksetzen                сброс
```

Если время T17 истекло, метка M 63.7 устанавливается и таймер запускается вновь. Во время следующего цикла обработки опрос UNT 17, поскольку время не истекло, дает результат «0»..Метка M63.7 снова сбрасывается. Таким образом, как и импульсная метка обработки фронта, она имеет состояние «1» только в течение времени одного цикла.

Благодаря режиму "удлиненный импульс" время, несмотря на результат опроса «0» истечет только после истечения установленного периода. Как только время истечет, метка M63.7 установится снова и т.д.

Последующее программирование триггера со счетным входом /пересчетной схемы/ описано в разделе 3.6. В качестве импульсной метки служит метка M63.7. Пересчетная схема реализуется с помощью метки

M 49.0.

12.5 Счетные Функции

Реализуемые с помощью устройств автоматизации 130 A, K счетные функции занимают специальный блок периферии /блок таймера/счетчиков 390/. Параметры счета устанавливаются на блоке таймера/счета через инженерную панель 391. После этого установка, сброс или обратный счет в счетчиках происходят по программе. Опрашивать счетчики по /цифровому/ параметру нельзя.

Если в этих устройствах хотят реализовать функции счета без блока таймера/счетчика, то это делают программным путем, как и в устройстве S5-110A /см.раздел 11.6 и 11.6/.

12.6 Функции загрузки и переноса

Операции загрузки и переноса в устройствах автоматизации 130 A и 150 K ограничены шириной в 8 битов. В отличие от других устройств автоматизации здесь загрузка и перенос зависят от результата логической операции. Они возможны только при логическом результате «1».

При переносе в блоки периферии отображение процесса на выходах не синхронизируется. Поэтому рекомендуется выполнять следующее программирование:

Пример:

```
L    MB 10    , загрузка одного байта меток
T    PB 5       перенос в блок вывода
T    AB 5       синхронизация отображения процесса
```

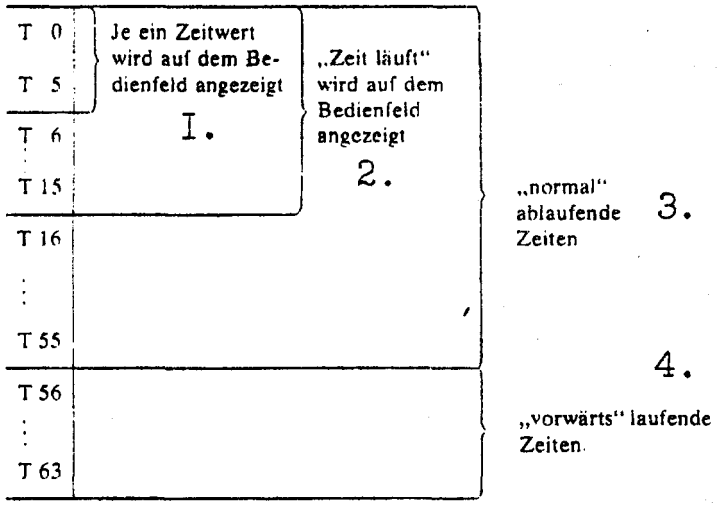
Операции загрузки и переноса используются также для индикации сбоев с помощью блока таймер /счетчиков /см.следующий раздел/.

12.7 Блок таймеров/счетчиков 390 с инженерной панелью 391

Блок таймеров/счетчиков 390 выполняет следующие функции:
обрабатывает функции времени обрабатывает функции счета сигнализирует о сбоях
В блоке таймер /счетчиков размещаются все адресуемые от процессора таймеры (T0...T63) и счетчики (Z0...Z15) Через байты периферии PB32 - PB47 на инженерной панели в качестве сигналов о сбоях вызывают индикацию цифр от 64 до 95.

Функции времени

Таймеры от T0 до T63 разделены по 4 зонам:



I. На инженерной панели индицируется по I параметру времени

2. "Течение времени" индицируется на инженерной панели

3. "нормально" истекающее время /к нулю/

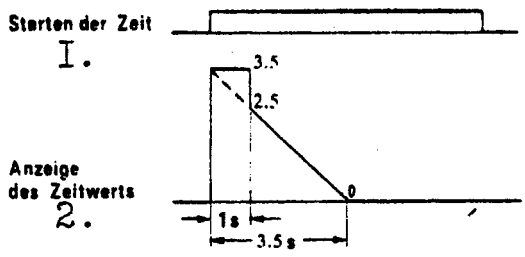
4. время, отсчитываемое "вперед" /от нуля/

Индикация параметров времени

Если время специально не задано, то показания таймеров Т0 - Т5 можно видеть на инженерной панели. Так как индицироваться может только одно значение, то предпочтение отдается параметру таймера с самым высоким адресом. Таким образом, приоритет будет за таймером Т5. При запуске таймера в - течение одной секунды индицируется установленное значение времени. По истечении этой секунды на индикаторе появляется текущее фактическое значение времени.

Пример:

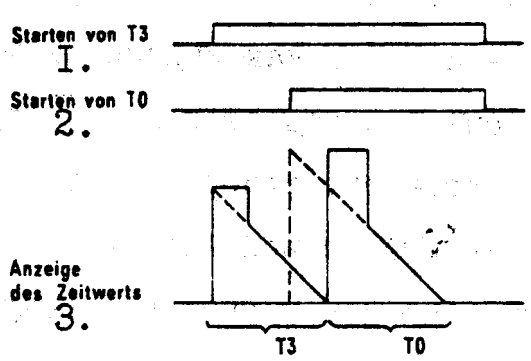
Если на таймере Т3 задана установка 3,5 с, то после запуска таймера в течение одной секунды на индикаторе будет величина 3,5, а затем - 2,5 /фактическое значение на данный момент/. Отсчет ведется в направлении к нулю.



I. Запуск таймера

2. Индикация параметра времени

Если во время индикации параметра Т3 будет запущен таймер Т0 или Т1 /меньший приоритет/, то установка этого таймера появится на индикаторе только после того, как истечет время Т3. Заданное значение в течение одной секунды стоит неподвижно, а затем появляется текущее значение.

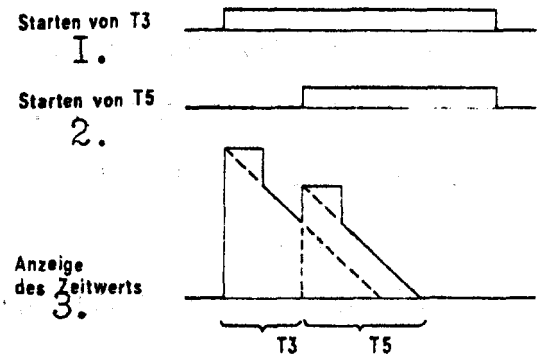


1. Запуск таймера Т3

2. Запуск таймера Т0

3. Индикация параметра времени

Если во время индикации времени T3 будут запущены таймеры T4 или T5 /с более высоким приоритетом/, то их параметры появятся немедленно. Показание в течение одной секунды стоит неподвижно, а затем появляется текущий параметр времени.



1. Запуск таймера T3

2. Запуск таймера T5

3. Индикация параметра времени

Индикация ведущегося отсчета времени

16 светодиодов на инженерной панели показывают, какой таймер от T0 до T15 находится в работе. До истечения заданного времени, т.е. пока опрос на состояние "Гадает результат «1», будет светиться соответствующий светодиод.

"Нормально" считающие таймеры

Время на таймерах TO - T55 отсчитывается так, как описано в разделе 4. При запуске таймера через операции SIT или SVT происходит установка ячейки времени на заданное на блоке и записанное в памяти значение. После этого находящийся в ячейки времени параметр будет понижаться на одну единицу в соответствии с дискретностью. Значение "ноль" означает, что время "истекло".

"Вперед" считающие таймеры

Таймеры T56 - T63 отсчитывают время "вперед". При запуске одного из этих таймеров значение времени равно нулю. Оно повышается в соответствии с дискретностью на одну единицу. Остановив таймер,

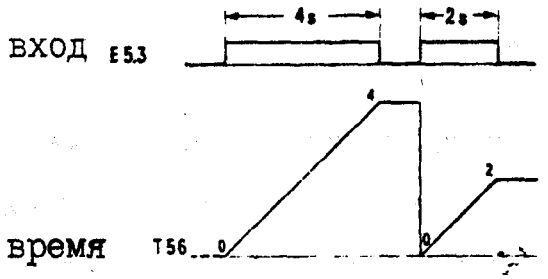
можно узнать истекшее время.

При запуске таймера, операцией SIT отсчет времени будет вестись до тех пор, пока стартовая операция имеет состояние "1". При состоянии «0» величина времени больше не изменяется и теперь можно узнать, сколько времени сохранялось состояние «1». При смене логического результата с «0» на-"1" параметр времени сбрасывается на ноль и начинается новый отсчет времени.

Пример:

```
U   E  5.3      Mit Eingang E5.3
SI  T  56       wird die Zeit T56 gestartet
```

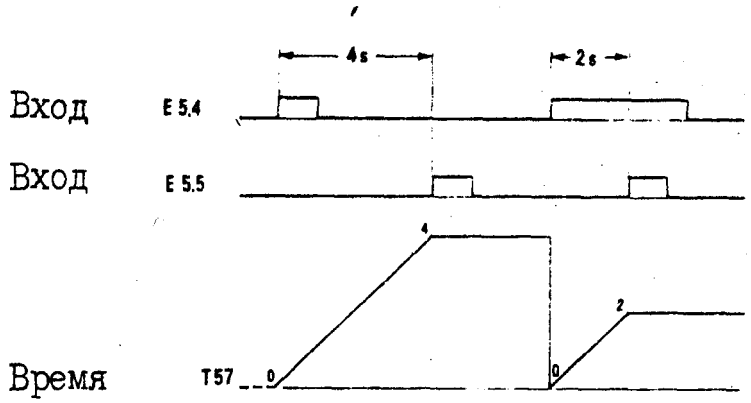
через вход E5.3
запуск таймера T56



При запуске таймера с помощью операции SVT отсчет времени ведется с запоминанием, т.е. даже если логическое состояние при стартовой . операции снова равно «0», отсчет продолжается. В этом случае, чтобы остановить отсчет, необходимо запрограммировать операцию RT.

Пример:

```
U   E  5.4
SV  T  57      запуск таймера в режиме удлиненного импульса
U   E  5.5
R   T  57      сброс
```



Счетные функции

Установка счетчиков Z0 - Z15 на установленную на блоке или записанную в памяти величину производится с помощью операции SZ. С помощью операции ZRZ ведется отсчет от установленного значения с понижением на одну единицу. Наименьшее число равно нуля. С помощью операции RZ производится прямой сброс счетчика на ноль.

Индикация сигналов о сбоях

На инженерную панель 391 можно выдавать 32 сигнала. Эти сигналы на индикаторе имеют номера от 64 до 95. Частота мигания 2 Гц.

Номера сигналов разделены на 2 диапазона. Диапазон 1 охватывает номера от 64 до 79. А номера от 80 до 95 составляют диапазон 2. Номера сигналов определяются по указанию байта периферии. Бит № 6 этого байта вызывает нужный диапазон: состояние «0» означает диапазон I, состояние «1» - диапазон 2. Остальные биты этого байта могут иметь любые значения и на индикацию не влияют.

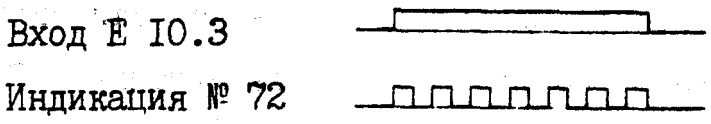
Перенос байта периферии	Бит 6 = "0" диапазон I	Бит 6 = "I" диапазон 2
T PB 32	Nr. 64	Nr. 80
33	65	81
34	66	82
35	67	83
36	68	84
37	69	85
38	70	86
39	71	87
40	72	88
41	73	89
42	74	90
43	75	91
44	76	92
45	77	93
46	78	94
47	79	95

При поступлении нескольких сигналов, индицируется тот, который в программе будет найден первым. Только после снятия этого сигнала покажется следующий.

Индикация сбоев без квитирования

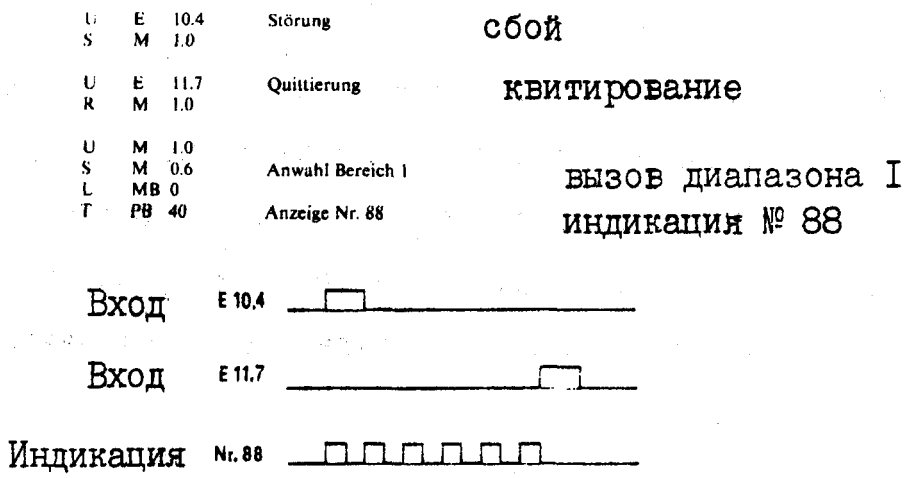
При сигнале «1» на входе E 10.3 на индикации должен появляться номер 72.

U	E 10.3	Störung	сбой
R	M 0.6	Anwahl Bereich 1	вызов диапазона I
L	MB 0		
T	PB 40	Anzeige Nr. 72	индикация № 72



Индикация сбоев с квитированием

При сигнале «1» на входе. E 10.4 должен появляться номер 88. Он может исчезнуть только при появлении на входе E 11.7 состояния «1» /"квитирование"/.



12.8 Шаговые или сблокированные системы управления

В отличие от систем с логическими связями в сблокированных системах выдается разрешение на обработку только одной определенной части программы /одного шага из целой последовательности/. Переход к следующему шагу происходит только при выполнении условий предыдущего шага.

Отдельные шаги цепочки обозначаются вспомогательными метками /"шаговыми метками"/. Установление метки шага обозначает, что выходные данные соответствующего шага только что выданы и "разрешены" к использованию условия следующего шага. После того, как условия следующего шага выполнены, шаговая метка сбрасывается и устанавливается в следующем шаге.

Изображение сблокированных цепочек

Для обозначения характеристики на каждом выходе изображаемой цепочки ставится s, NS или ST.

S „speichernd" "запоминающая"

Выход устанавливается в течение нескольких шагов сохраняет состояние «1», пока не произойдет новый сброс. Поэтому к этой характеристике добавляется спецификация "вкл" (SA) или "откл" (RA).

NS „nicht speichernd" "без запоминания"

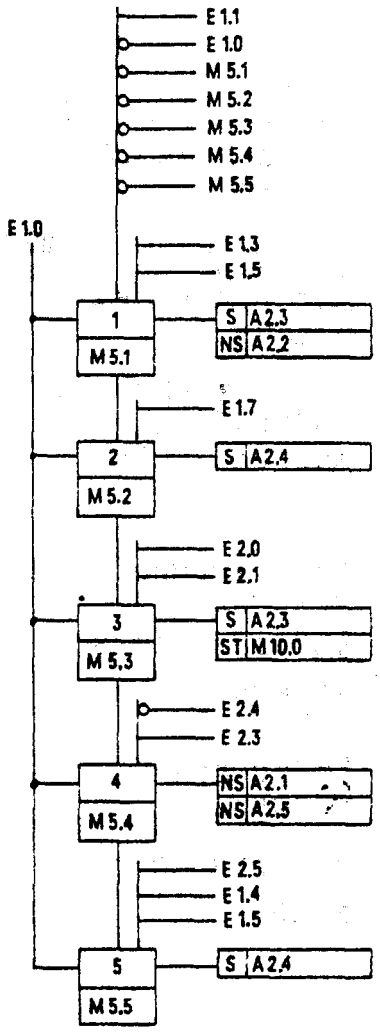
На выходе должно быть состояние «1», если выполнены условия очередного шага. При переключении на следующий шаг он должен снова иметь состояние «0».

Эту функцию нельзя запрограммировать командой =A, так как, как только устанавливается очередной шаг, его условия больше не выполняются". Выход оставался бы установленным только в течение одного цикла. Поэтому программируют SA, а в следующем шаге RA.

ST „Impuls" (stored terminated) "импульс"

Выход устанавливается только как импульс. Для внешних импульсов для реализации установки берется звено времени. Для внутренних импульсов достаточно, если состояние «1» сохраняется только в течение одного цикла. Поэтому программируют =A /см.выше/.

Структура сблокированной цепочки



Программирование

```
U   E  1.0     LOESCHEN DER ABLAUFKETTE          Сброс сблокированной цепочки
R   M  5.1     RUECKSETZEN DER SCHRITTMERKER     сброс метки шага
R   M  5.2
R   M  5.3
R   M  5.4
R   M  5.5
R   A  2.1     RUECKSETZEN DER AUSGAENGE         сброс выходов
R   A  2.2
R   A  2.3
R   A  2.4
R   A  2.5

U   E  1.1     START DER ABLAUFKETTE             запуск сблокированной цепочки
UN  E  1.0     WENN KEIN RUECKSETZEN ANLIEGT     если нет команды на сброс
UN  M  5.1     UND KEIN SCHRITT GESETZT IST      и нет установки шага
UN  M  5.2
UN  M  5.3
UN  M  5.4
UN  M  5.5

U   E  1.3     BEDINGUNGEN      SCHRITT 1        условия   шаг I
U   E  1.5
S   M  5.1     SCHRITTMERKER                     шаговая метка
S   A  2.3     AUSGABEN                          выдача
S   A  2.2

U   M  5.1     "FREIGABE"       SCHRITT 2        "разрешение"   шаг 2
U   E  1.7     BEDINGUNG                         условие
S   M  5.2     SCHRITTMERKER                     шаговая метка
R   M  5.1
S   A  2.4     AUSGABEN                          выдача
R   A  2.2

U   M  5.2     "FREIGABE"       SCHRITT 3        "разрешение"   шаг 3
U   E  2.0     BEDINGUNGEN                       условия
U   E  2.1
S   M  5.3     SCHRITTMERKER                     шаговая метка
R   M  5.2
R   A  2.3     AUSGABEN                          выдача
*   M  10.0

U   M  5.3     "FREIGABE"       SCHRITT 4        "разрешение"   шаг 4
UN  E  2.4     BEDINGUNGEN                       условия
U   E  2.3
S   M  5.4     SCHRITTMERKER                     шаговая метка
R   M  5.3
S   A  2.1     AUSGABEN                          выдача
S   A  2.5

U   M  5.5     "FREIGABE"       SCHRITT 5        "разрешение"   шаг 5
U   E  2.5     BEDINGUNGEN                       условия
U   E  1.4
U   E  1.5
S   M  5.5     SCHRITTMERKER                     шаговая метка
R   M  5.4
R   A  2.4     AUSGABEN                          выдача
R   A  2.1
R   A  2.5
```

В нашем примере цепочка с входом Е 1.0 приводится в основное состояние, т.е. все шаговые метки сбрасываются. Только после того, как все метки сброшены, а на входе Е 1.0 состояние «0», через вход Е 1.1 дается разрешение в цепочку. Если условия первого шага выполнены /на входах Е I.? и Е 1.5 состояние «1»/, устанавливается первый шаг. Затем устанавливаются выходы /А2.3 и А2.2/ и выдается разрешение на выполнение следующего шага. Таким образом, каждый шаг в цепочке состоит из опроса предыдущей шаговой метки /"разрешение"/ условий /дальнейшей коммутации/ этого шага установки шаговой метки и сброса предыдущей шаговой метки и из выдачи.

12.9 Операции окончания блока

Программа в устройстве автоматизации. 130 А или 130 К завершается операцией "окончание блока".

Абсолютное окончание блока BE

Операцией BE заканчивается обработка программы STEP-5. Процессор устанавливает счетчик адресов в нулевое положение и начинает обработку программы с ее начала. С помощью операции BE дополнительно передаются состояния сигналов регистра отображения процесса в блоки выводов /см.раздел 12.5/. Кроме того, операция BE перебрасывает каскад времени на контроль времени цикла.

Программирующие устройства 650 и 651 пишут операцию BE в конце программы автоматически. Поэтому при работе с этими устройствами ее не нужно вводить специально.

Условное окончание блока BEB

С помощью операции BEB можно вызвать преждевременно окончание обработки программы STEP-5. Эта операция выполняется только при условии, что логический результат при обработке операции равен «1». Тогда операция будет выполнена так же, как и при абсолютном окончании блока BE, однако без переноса содержимого регистра отображения процесса.

Если логический результат «0», BEB не выполняется и продолжается обработка программы по циклу. Однако логический результат становится равным "1". После операции BEB, например, можно непосредственно установить или сбросить метку. Если после BEB стоит опрос, например UE, то этот опрос является первичным; с него начинается новая логическая операция /см.раздел 2.3/.

Благодаря операции BEB можно ускорить время реакции управляющего устройства. Для этого в программе STEP-5 опрашивают входы, которые считаются "входами прерываний", и затем в зависимости от логического результата делается скачок обратно на начало программы. Таким образом, критичные по времени части программы - следует писать в ее начале.

Этот обратный скачок, однако, не должен быть статичным, т.е. происходить постоянно, так как в противном случае операция BE не будет обрабатываться в течение длительного времени /тогда сработает контроль времени цикла. Поэтому в качестве входов лучше всего подходят "динамические" входы /например, цифровой ввод 432 со сборным сигналом/. Эти входы при смене состояния сигнала дают результат «1» только при первом опросе. При отсутствии динамических входов перед операцией BEB следует программировать обработку фронтов.

13 ПРОГРАММИРОВАНИЕ УСТРОЙСТВ АВТОМАТИЗАЦИИ S5-110S И S5-130W

Устройства автоматизации СИМАТИК S5-110S и СИМАТИК S5-130W различаются в основном по возможностям подключения периферийных блоков. Тогда как в устройствах S5-110S применяются блоки периферии устройств 110A /в блочном исполнении/, блоки периферии, которые можно вставлять в центральное устройство S5-130W , должны быть в компактном исполнении. Подключаемые к устройству S5-130W приборы расширения могут иметь как защищенное /в капсулах/, так и компактное /в стойке ES-902/ исполнение. За исключением различных областей операндов объем операций, выполняемых обоими устройствами, примерно одинаков. В этой главе рассматривается объем операций данных приборов. Запас операций обоих устройств приводится отдельно в разделе 13.1. Логические операции, непосредственно программируемые с помощью обоих устройств, описаны в разделе 13.2. Функции памяти и отображения процессов для входов и выходов описаны в разделе 13.3. Следующий раздел /13.4/ раскрывает программирование функций времени и счета. В разделе 13.5 дается обобщенное описание дискретных функций, программируемых с помощью устройств S5-110S и S5-130W. к дискретным функциям относятся функции загрузки и переноса, а также функции сравнения. Представление чисел, как правило, шестнадцатиразрядное с фиксированной запятой (F). В последнем разделе приведены различные структуры программ, реализуемых в этих устройств с помощью команд языка STEP-5 .

Общие замечания по устройству, S5-110S Устройство автоматизации S5-110S при полном развитии может работать с 256 входами, 256 выходами, 128 таймерами и 128 счетчиками. Кроме того, допускается пользование 1024 сохраняющимися и 1024 не сохраняющимися метками.

Отображение процессов существует как для входов, так и для выходов. Адресация блоков входов/выходов производится независимо от разъемов, куда вставляются блоки. Адресация производится на блоках. Метки, счетчики и таймера сразу входят в состав центрального блока. Каких-либо специальных блоков периферии для них не требуется.

Общие замечания по устройству S5-130E

Устройство автоматизации S5-130W при полном развитии может иметь 512 входов, 512 выходов, 12.8 таймеров и 64 счетчика. Кроме того, возможно использование 1024 реманентных /сохраняющихся/ и 1024 не сохраняющихся /не защищенных батареей от исчезновения напряжения/ меток.

Как для входов, так и для выходов существуют регистры отображения процессов. Адресование блоков ввода/вывода происходит независимо от номера разъема блока в общей стойке. Адресование производится на блоках. Метки, таймеры и счетчики находятся непосредственно в центральном блоке. Специальных блоков периферии для них не требуется Устройство s5-l30W рассчитано на подключение аналоговых блоков ввода/ вывода. В общей сложности может быть 64 канала /аналоговых входов или выходов/.

13.1 Запас операций

По запасу операций оба устройства различаются в основном различными областями операндов.

Запас операций устройства автоматизации S5-110S

Логические операции		
U	-	операция И, опрос на состояние "I"
O	-	операция ИЛИ, опрос на состояние "I"
UN	-	операция И опрос на состсяние "О"
ON	-	операция ИЛИ, опрос на состояние "О"
	E 0.0- 3I.7	входа
	A 0.0- 3I.7	выхода
	M 0.0-127.7	области меток незащищенной
	M 128.0-255.7	области меток защищенной
	T 0 -127	таймера
	Z 0 -I27	счетчика
U(		операция И над выражением в скобках
O(		операция ИЛИ над выражением в скобках
)		скобки закрыть
O		операция ИЛИ над функцией И

Функции памяти			
S	-		установка
R	-		сброс
=	-		присвоение
	E	0.0- 31.7	входа
	A	0.0- 31.7	выхода
	M	0.0-127.7	области меток незащищенной
	M	128.0-255.7	области меток защищенной
Функции времени			
SI	-		запуск таймера в режиме короткого импульса
SV	-		запуск таймера в режиме продленного импульса
SE	-		запуск таймера в режиме задержки включения
SS	-		запуск таймера в режиме задержки включения с запоминанием
SA	-		запуск таймера в режиме задержки отключения
R	-		сброс
	T	0-127	уставки времени
Функции счета			
ZV	-		прямой счет
ZR	-		обратный счет
S	-		установка
R	-		сброс
	Z	0-127	счетчика

Функции загрузки и переноса

L	—		Laden	Загрузка
T	—		Transferieren	Перенос
	EB	0— 31	eines Eingangsbytes	байта входов
	EW	0— 30	eines Eingangsworts	слова входов
	AB	0— 31	eines Ausgangsbytes	байта выходов
	AW	0— 30	eines Ausgangsworts	слова выходов
	MB	0—127	eines nicht remanenten Merkerbytes	несохр.байта меток
	MB	128—255	eines remanenten Merkerbytes	сохран.байта меток
	MW	0—126	eines nicht remanenten Merkerworts	несохр.слова меток
	MW	128—254	eines remanenten Merkerworts	сохран.слова меток
	DR	0—255	eines rechten Datenbytes	правого байта данных
	DL	0—255	eines linken Datenbytes	левого байта данных
	DW	0—255	eines Datenworts	слова данных
	PB	0— 31	eines Peripheriebytes	байта периферии

L	—		Laden	Загрузка
	T	0—127	eines Zeitwerts	параметра времени
	Z	0—127	eines Zählwerts	параметра счета
	KF	−32768 bis +32767	einer Konstanten als Festpunktzahl	константы как числа с фикс.запятой
	KB	0 bis 255	einer Konstanten als ein Byte	константы в виде байта
	KY	0 bis 255, 0 bis 255	einer Konstanten als zwei Bytes	константы в виде двух байтов
	KH	0000 bis FFFF	einer Konstanten als Hexamuster	константы в виде I6-разр.числа
	KM	0000 0000 0000 0000 bis 1111 1111 1111 1111	einer Konstanten als Bitmuster	константы в виде комбинации битов
	KC	<ASCII-Zeichen> <ASCII-Zeichen>	einer Konstanten als zwei Zeichen	константы в виде двух знаков
	KT	000.0 bis 999.3	einer Konstanten als Zeitwert	константы как параметра времени
	KZ	000 bis 999	einer Konstanten als Zählwert	константы как параметра счета
LC	—		Laden codiert	Кодированная загрузка
	T	0—127	eines Zeitwerts	параметра времени
	Z	0—127	eines Zählwerts	параметра счета

Функции сравнения

!=F	Vergleich auf „Gleich" (16 Bits)	Сравнение на"равно" /I6 разр./
<F	Vergleich auf „Kleiner" nach 16-Bit-Festpunkt-charakteristik	Сравнение на "меньше/ по I6-разр.характеристике с фикс.запятой
>F	Vergleich auf „Größer" nach 16-Bit-Festpunkt-charakteristik	Сравнение на "больше" по I6-разр.характеристике с фикс.запятой

Арифметические функции

+F	Addiere 16-Bit-Festpunktzahlen	Сложение I6-разр.чисел с фикс.зап.
−F	Subtrahiere 16-Bit-Festpunktzahlen	Вычитание I6-разр.чисел с фикс.зап.

Организационные функции

SPA	—		absoluter Aufruf	абсолютный вызов
SPB	—		bedingter Aufruf	условный вызов
	PB	0—127	eines Programmbausteins	программного блока
	FB	0— 47	eines Funktionsbausteins	функционального блока
A	DB	0— 63	Aufruf eines Datenbausteins	вызов блока данных
BE			Bausteinende	конец блока
BEB			bedingtes Bausteinende	условный конец блока
STP			Stopp	стоп

Запас операций устройства автоматизации S5-130W

Логические операции

U	–		UND-Verknüpfung, Abfrage auf Signalzustand „1"	И, опрос на "I"
O	–		ODER-Verknüpfung, Abfrage auf Signalzustand „1"	ИЛИ, опрос на "I"
UN	–		UND-Verknüpfung, Abfrage auf Signalzustand „0"	И, опрос на "0"
ON	–		ODER-Verknüpfung, Abfrage auf Signalzustand „0"	ИЛИ, опрос на "0"
	E	0.0– 63.7	eines Eingangs	входа
	A	0.0– 63.7	eines Ausgangs	выхода
	M	0.0–127.7	eines nicht remanenten Merkers	несохран.метки
	M	128.0–255.7	eines remanenten Merkers	сохран.метки
	T	0 –127	einer Zeit	таймера
	Z	0 – 63	eines Zählers	счетчика
U(			UND-Verknüpfung von Klammerausdrücken	И перед скобками
O(			ODER-Verknüpfung von Klammerausdrücken	ИЛИ перед скобками
)			Klammer zu	закрыть скобку
O			ODER-Verknüpfung von UND-Funktionen	ИЛИ над И

Функции памяти

S	–		Setzen	Установка
R	–		Rücksetzen	Сброс
=	–		Zuweisung	Присвоение
	E	0.0– 63.7	eines Eingangs	входа
	A	0.0– 63.7	eines Ausgangs	выхода
	M	0.0–127.7	eines nicht remanenten Merkers	несохран.метки
	M	128.0–255.7	eines remanenten Merkers	сохран.метки

Функции времени

SI	–		Starten als Impuls	Запуск коротким импульсом
SV	–		Starten als verlängerter Impuls	Запуск продленным импульсом
SE	–		Starten als Einschaltverzögerung	Запуск задержки включения
SS	–		Starten als speichernde Einschaltverzögerung	Запуск задержки вкл.с памятью
SA	–		Starten als Ausschaltverzögerung	Запуск задержки отключения
R	–		Rücksetzen	Сброс
	T	0–127	einer Zeit	таймера

Функции счета

ZV	–		Zählen vorwärts	Прямой счет
ZR	–		Zählen rückwärts	Обратный счет
S	–		Setzen	Установка
R	–		Rücksetzen	Сброс
	Z	0–63	eines Zählers	счетчика

Функции загрузки и переноса

L	–		Laden	Загрузка
T	–		Transferieren	Перенос
	EB	0– 63	eines Eingangsbytes	байта входов
	EW	0– 62	eines Eingangsworts	слова входов
	AB	0– 63	eines Ausgangsbytes	байта выходов
	AW	0– 62	eines Ausgangsworts	слова выходов
	MB	0–127	eines nicht remanenten Merkerbytes	несохран.байта меток
	MB	128–255	eines remanenten Merkerbytes	сохран.байта меток
	MW	0–126	eines nicht remanenten Merkerworts	несохран.слова меток
	MW	128–254	eines remanenten Merkerworts	сохран.слова меток
	DR	0–255	eines rechten Datenbytes	правого байта данных
	DL	0–255	eines linken Datenbytes	левого байта данных
	DW	0–255	eines Datenworts	слова данных
	PB	0–255	eines Peripheriebytes	байта периферии
L	–		Laden	Загрузка
	T	0–127	eines Zeitwerts	параметра времени
	Z	0– 63	eines Zählwerts	параметра счета
	KF	–32768 bis +32767	einer Konstanten als Festpunktzahl	константы как числа с фикс.зап.
	KB	0 bis 255	einer Konstanten als ein Byte	константы как байта
	KY	0 bis 255, 0 bis 255	einer Konstanten als zwei Bytes	константы как двух байтов
	KH	0000 bis FFFF	einer Konstanten als Hexamuster	константы в виде I6-разр.числа
	KM	0000 0000 0000 0000 bis 1111 1111 1111 1111	einer Konstanten als Bitmuster	константы как комбинации битов
	KC	<ASCII-Zeichen> <ASCII-Zeichen>	einer Konstanten als zwei Zeichen	константы как два знака
	KT	000.0 bis 999.3	einer Konstanten als Zeitwert	константы как параметра времени
	KZ	000 bis 999	einer Konstanten als Zählwert	константы как параметра счета
LC	–		Laden codiert	Кодированная загрузка
	T	0–127	eines Zeitwerts	параметра времени
	Z	0– 63	eines Zählwerts	параметра счета

Функции сравнения

!=F	Vergleich auf „Gleich" (16 Bits)	Сравнение на "равно" /I6 разр./
<F	Vergleich auf „Kleiner" nach 16-Bit-Festpunkt-charakteristik	Сравнение на "меньше" по I6-разр характеристике с фикс.запятой
>F	Vergleich auf „Größer" nach 16-Bit-Festpunkt-charakteristik	Сравнение на "больше" по I6-разр характеристике с фикс.запятой

Арифметические функции

+F	Addiere 16-Bit-Festpunktzahlen	Сложение I6-разр.чисел с фикс.з.
–F	Subtrahiere 16-Bit-Festpunktzahlen	Вычитание I6-разр.чисел с фикс.з.

Организационные функции

SPA	–		absoluter Aufruf	абсолютный вызов
SPB	–		bedingter Aufruf	условный вызов
	PB	0–255	eines Programmbausteins	программного блока
	FB	0–255	eines Funktionsbausteins	функционального блока
A	DB	0–241	Aufruf eines Datenbausteins	вызов блока данных
BE			Bausteinende	конец блока
BEB			bedingtes Bausteinende	условный конец блока
STP			Stopp	стоп

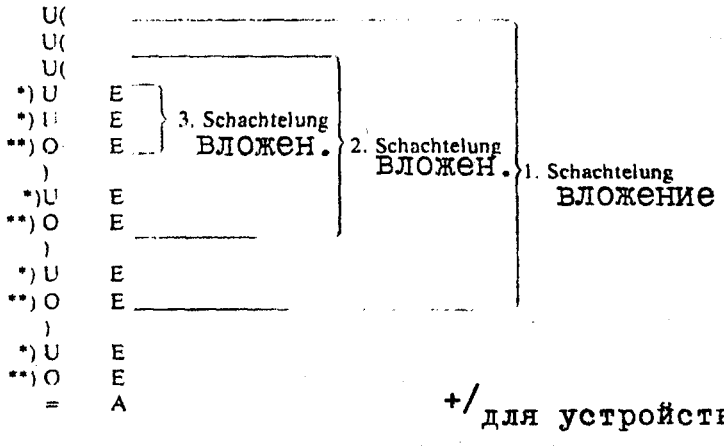
13.2 Логические операции

В устройствах автоматизации S5-110S и S5-130W допускаются все описанные в главе 2 логические функции.

Таблица команд

При записи таблицы команд разрешается программировать три раза подряд команду "скобку открыть" без предварительного завершения выражения в скобках /три уровня суперпозиционирования или вложения/.

Пример: /для 3 уровней/



+/для устройства S5-110S

В устройствах S5-130W команду "скобку открыть" можно писть 6 раз подряд /шесть уровней суперпозиционирования/

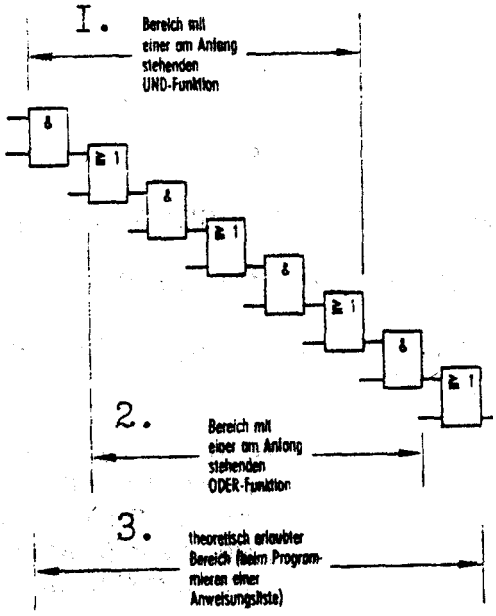
') Вместо этих команд можно запрограммировать производное количество операций И или И-НЕ.

") Вместо этих команд можно запрограммировать произвольное количество операций ИЛИ или ИЛИ-НЕ и, отделяя отдельными командами ИЛИ, произвольное количество функций И.

Функциональная схема

При программировании в виде функциональной схемы можно пользоваться всей имеющейся в распоряжении шириной дисплея. Начинать разрешается как с функции И, так и с функции ИЛИ.

Пример:



1. Область, начинающаяся с функции И

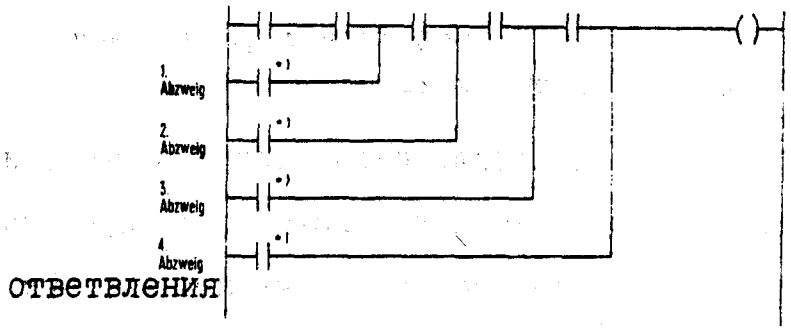
2. Область, начинающаяся с функции ИЛИ

3. Теоретически разрешенная область /при программировании таблицы команд/.

Контактная схема

При программировании в виде контактной схемы разрешается одновременно открывать в общей сложности 4 ответвления в устройстве S5-110S и 7 ответвлений в устройстве S5-130W.

Пример:



Вместо изображенных контактов разрешается /насколько позволяет

ширина экрана дисплея/ последовательно включать другие контакты.

*)Вместо этих контактов параллельно /к главной ветви/ разрешается включать другие отдельные или последовательно соединенные контакты.

Команда "ИЛИ" скобку открыть o(

В чисто логических операциях функция "открыть скобку" всегда связана с какой-либо функцией И, так как "открыть скобку" пишется только в том случае, если функцию ИЛИ нужно поставить перед функцией И. Язык программирования STEP-5 позволяет также использовать скобки и в тех случаях, когда нужно произвести промежуточное запоминание логического результата /см.раздел 3.3/. В этом случае может также встречаться сочетание функции "открыть скобку" с функцией ИЛИ. Тогда пишется команда "O('. В разделе 14.3 приводится пример к этому пункту.

13.3 Функции памяти

В устройствах автоматизации S5-110S и допускается использование всех функций памяти, описанных в разделе 3.

Отображение процесса для входов

Устройства автоматизации S5-110S и S5-130W располагают регистрами отображения процессов на входах и на выходах.

В начале циклической обработки программы /но еще до вызова организационного блока 0B1/ происходит опрос блоков-входов и запоминание состояний сигналов на них в регистре отображения процесса.

Прикладная программа теперь работает с отображением процесса. В случае изменения на входе во время обработки программы изменений в регистре не происходит. Отображение процесса остается в прежнем состоянии. Новые состояния сигналов будут загружены лишь в начале программы.

С помощью команды "загрузка периферии" LPB к блокам вводов можно также обращаться напрямую /в обход регистра отображения процесса/.

Таким же образом непосредственно можно регистрировать изменения
сигналов на входах. Это имеет большое значение, например, для обработки по прерываниям.

Благодаря регистру отображения процесса для входов входы можно устанавливать и сбрасывать как выходы /см.раздел 3.8/.

Метки

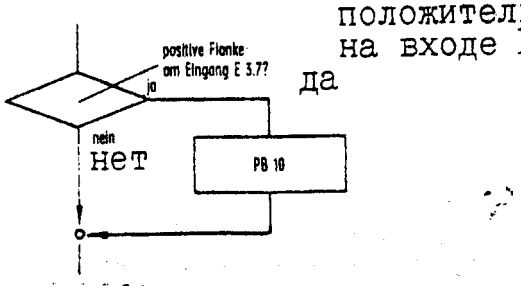
Область меток в обоих устройствах автоматизации разделена на 2 зоны - зону, которая защищена от исчезновения напряжения батареей, и незащищенную зону.

Метки в незащищенной зоне /M 0.0 -M 127.7/ как при новом, так и повторных запусках процессора устанавливаются в состояние «0». Таким образом, они ведут себя аналогично выходам. Метки в защищенной зоне /M 128.0 - M 255.7/ сбрасываются только при новом запуске. При повторном запуске они имеют то же состояние, что и до отключения аппарата. Это состояние они сохраняют около 4 недель /получая питание от встроенной батареи/. Обработка фронтов Условные ВЫЗОВЫ блоков В обоих устройствах автоматизации ПОЗВОЛЯЮТ вести обработку целых частей программы только при появлении фронта сигнала.

Пример:

Если вход E 3.7 меняет состояние с «0» на «1», должен обрабатываться блок PB 10.

Структура программы



положительный фронт на входе E 3.7?

да

нет

Программа

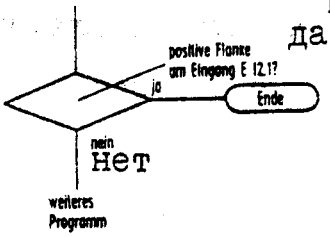
```
U    E   3.7
UN   M   50.3
=    M   100.0      Импульсная метка

U    M   100.0
S    M   50.3

UN   E   3.7        Метка фронта
R    M   50.3

U    M   100.0      Блок PB IO будет обработан только при условии, что
SPB  PB  10         импульсная метка имеет состояние "I".
```

Если импульсная метка M 100.0 опрашивается на состояние «0», то условный вызов выполняется всегда, за исключением фронта сигнала.

Пример:

При смене на входе E 12.1 состояния с «0» на «1» должна быть обработана следующая программа. В противном случае обработку блока
следует прекратить.

Структура программы



на входе E I2.I
положительный фронт?
да конец
Нет
дальнейшая
программа

Программа

```
U     E   12.1
UN    M   50.4
=     M   100.1
U     M   100.1
S     M   50.4
UN    E   12.1
R     M   50.4
UN    M   100.1
BEB
```

Импульсная метка

Метка фронта

При отсутствии переднего фронта обработка программы прекращается в этом блоке.

13.4 Функции времени и счета

В устройствах автоматизации S5-110S можно программировать 128 установок времени и 128 функций счета; в устройствах S5-130W имеется 128 таймеров и 64 счетчика. Таймеры и счетчики являются областями операндов в памяти центрального устройства. В этой связи для реализации функций времени и счета никаких дополнительных периферийных блоков не требуется.

Как таймер, так и счетчик изображаются в этой области 16-разрядным словом* Таймеры и счетчики программируются полным объемом функций, как это описано в главах 4 и 5.

Пример:

Если вход E22.0 будет иметь состояние «1» дольше 5 секунд, должна произойти установка выхода A7.3. Сброс выхода должен произойти при появлении на входе E22.0 сигнала «0».

Программа

U	E	22.0		
L	KT	500.0	Laden des Zeitwerts in den Akkumulator	Загрузка параметра времени в аккум.
SE	T	5	Starten der Zeit T5 als Einschaltverzögerung mit 5 Sekunden	Запуск таймера T5 с задержкой вкл. 5 секунд
U	T	5	Abfragen der Zeit	Опрос таймера
=	A	7.3		

13.5 Цифровые ФУНКЦИИ /дискретные функции/

В устройствах автоматизации S5-110S и S5-130W допускается прямое программирование функций загрузки, переноса и сравнения. Внутреннее представление чисел, как правило, по шестнадцатеричной системе с Фиксированной запятой /см.раздел 7.2/.

Функции загрузки и переноса

Функции загрузки и переноса разрешается выполнять только с операндами, адресуемыми по байтам и словам. Загрузка и перенос двойных слов не допускаются.

Функции загрузки и переноса в сочетании с блоками периферии могут программироваться только по байтам /LPB или TPB/. Область операндов Q - "расширенная периферия процесса" в устройствах S5-110S и S5-130W не предусмотрена.

Функции загрузки и переноса в устройствах автоматизации S5-110S и S5-130W выполняются независимо от результата логической операции и влияния на него не оказывают.

Функции сравнения

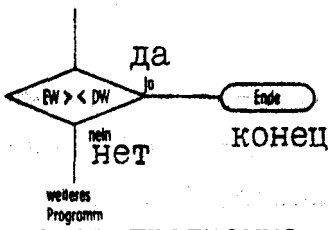
В устройствах автоматизации S5-110S и S5-130W разрешается прямое программирование только функций сравнения на "равно", "меньше" и "больше". Сравнения на "неравно", "больше - равно" и "меньше -равно" являются отрицаниями прямо программируемых сравнений:

- если не выполняется сравнение "равно", значит выполнено "неравно"
- если не выполняется сравнение "меньше", значит выполнено "больше - равно"
- если не выполняется сравнение "больше", значит выполнено "меньше - равно".

Пример:

Если значение слова входов EW 10 не равно значению слова данных DW 15,обработка блока должна быть прекращена.

Структура программы



след.программа

Программа

```
L     EW   10
L     DW   15
!=F                Vergleich auf Gleich
=     M    101.3
UN    M    101.3   Bei „Ungleich": Bausteinende
BEB
.
.
.
weiteres Programm
```

Сравнение на равенство

При неравенстве - конец блока

Следующая программа

При сравнения "меньше" и "больше" достаточно поменять операнды, чтобы реализовать сравнения "больше - равно" и "меньше - равно".

Пример:

Если значение слова меток MW 102 меньше или равно 105, должен быть вызван блок программы PB 27.

Программа

Вариант а

```
Variante a
L     MW    102
L     KF    +105
>F
=     M     27.3
UN    M     27.3
SPB   PB    27
```

Вариант б

```
Variante b
L     KF    +105
L     MW    102
>F
SPB   PB    27
```

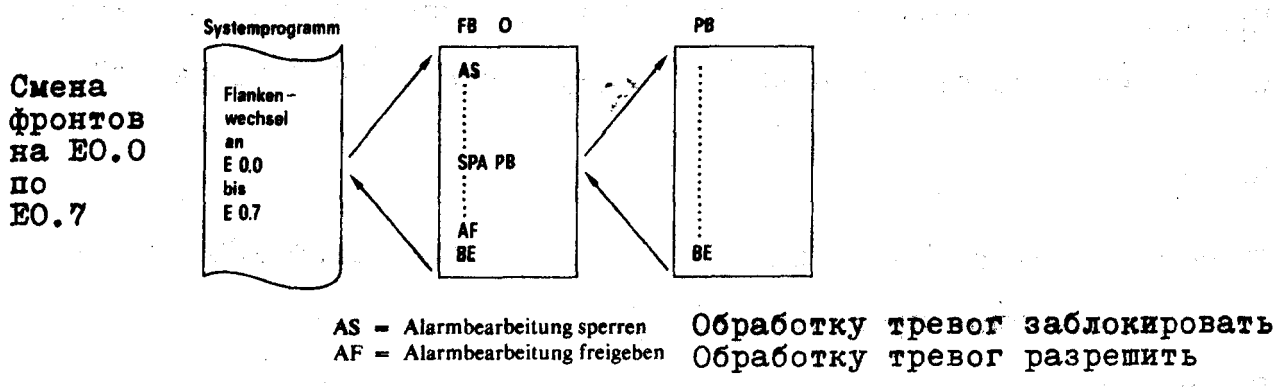
Вычислительные функции

В устройствах автоматизации S5-110S и S5-130W допускается выполнение только сложения +F : "сложение чисел с фиксированной запятой" и -F :"вычитание чисел с фиксированной запятой". Реализация остальных математических функций может производиться в функциональных блоках, где эти операции имитируются программным путем.

13.6 Структура программ

С помощью устройства автоматизации S5-110S допускается программировать 128 программных блоков, 48 функциональных блоков и 64 блока данных; в устройстве S5-130W - 255 программных блоков, 255 функциональных блоков и 242 блока данных.

В качестве организационных блоков в устройствах S5-110S и S5-130W допускается использование только организационного блока OB1 для циклической обработки программ /см.раздел 8.2/. Обработка по прерываниям /сигналам тревоги/ вызывается битами байта входов EB0. Однако обрабатываются не организационные блоки, а функциональный блок FB0 . В этом блоке и находится программа с управлением по прерываниям. Если в этой программе обработки сигналов тревоги вызываются другие блоки, то обработка тревог блокируется /с помощью операций AS и AF, см. "Программирование управляющих устройств на языке 8TEP-5,том 3/. Структура программы выглядит тогда следующим образом:

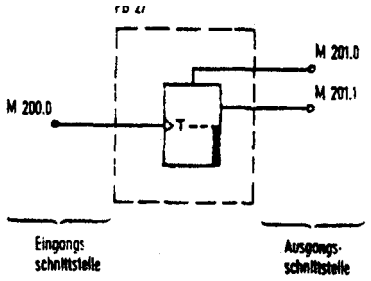


Обработка программ с управлением по времени не предусмотрена.

Функциональные блоки в обоих устройствах не параметрируются, т.е. их нельзя оснащать параметрами. Как, тем не менее, эти блоки можно многократно использовать в программе, показано ниже на примере одного из блоков программы /использование интерфейсных меток/. Функциональный блок в отличие от программного блока имеет преимущество, что его можно программировать расширенным запасом операций. Перечень этих операций описан в 3-м томе книги "Программирование управляющих устройств на языке STEP-5" "Самостоятельное программирование Функциональных блоков". Программу Функционального блока, однако, можно писать и документировать только в виде таблицы команд.

Пример: Использование интерфейсных меток

Программный блок PB 27 представляет собой пересчетную схему. В программе он будет многократно использоваться с различными операндами.
Программа блока PB 27 получает определенные метки. Эти метки образуют интерфейс между остальной программой и блоком PB 27. Прежде чем будет обрабатываться блок PB 27, устанавливаются интерфейсные метки входов; после окончания обработки будут опрошены интерфейсные метки выходов.

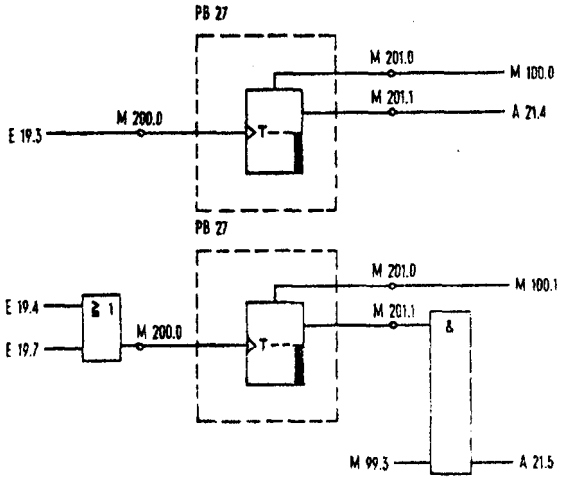


интерфейс входов — интерфейс выходов

Программа в блоке PB 27

```
UN   M  200.0   вход пересчетной схемы
R    M  201.0   метка фронта
ON   M  200.0 } эта функция ИЛИ имеет состояние "I"
O    M  201.0 } при условии отсутствия фронта
BEB             Если фронт отсутствует, блок PB 27
                заканчивается.

S    M  201.0   Установка метки фронта
UN   M  201.1   Эта часть программы обрабатывается только при наличии
=    M  201.1   фронта. Только тогда каждый раз изменяется также метка
                M 201.1, которая одновременно является выходом пере-
                счетной схемы.
BE              Конец блока
```

Пример многократного использования пересчетной схемы



Программа на языке STEP-5 для вышеуказанных функций гласит:

```
U    E   19.3
=    M   200.0      обеспечение интерфейса входов
U    M   100.0
=    M   201.0

SPA  PB  27         вызов и обработка пересчетной схемы

U    M   201.0
=    M   100.0      продолжение обработки интерфейса выходов
U    M   201.1
=    A   21.4

O    E   19.4
O    E   19.7
=    M   200.0      обеспечение интерфейса входов
U    M   100.1
=    M   201.0

SPA  PB  27         вызов и обработка пересчетной схемы

U    M   201.0
=    M   100.1      продолжение обработки интерфейса выходов
U    M   201.1
U    M   99.3
=    A   21.5
```

Условные вызовы блоков

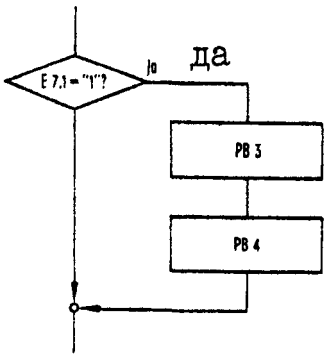
Внимание:

При одновременном использовании нескольких условных вызовов блоков следует каждый раз перед вызовом блока программировать условие вызова!

Пример:

Если на входе E 7.1 состояние "1", должны обрабатываться программные блоки PB 3 и PB 4.

Диаграмма



Программа

```
.
.
.
U    E  7.1    опрос
SPB  PB 3      условный вызов
U    E  7.1    опрос
SPB  PB 4      условный вызов
.
```

Если бы оба условных вызова были запрограммированы один за другим, то управляющее устройство прореагировало бы следующим образом:

А

```
U    E  7.1    на входе E 7.I состояние "0"
SPB  PB 3      Блок PB 3 не обрабатывается. Однако состояние изменяет-
               ся на "I" /см.разделы 9.3 и 9.4/.
SPB  PB 4      В связи с этим теперь обрабатывается блок PB 4
               /хотя вход E 7.I имеет состояние "0"/.
```

Б

```
U    E  7.1    на входе E 7.I состояние "I".
SPB  PB 3      Блок PB 3 обрабатывается.
SPB  PB 4      Теперь все зависит от того, при каком логическом
               результате был покинут блок PB 3.  Соответственно ему
               будет происходить обработка блока.
```

Рассмотренный выше пример справедлив и для условных вызовов функциональных блоков.

14 ПРОГРАММИРОВАНИЕ УСТРОЙСТВ АВТОМАТИЗАЦИИ S5-150A и S5-150K

Устройства автоматизации S5-150A и S5-150K отличаются друг от друга только по исполнению. Устройство S5-150A имеет защищенное исполнение /отдельные блоки помещены в металлические капсулы/;
а блоки устройства 150K размещены в стойке ES902 /компактное исполнение/. В обоих устройствах автоматизации используется один и тот же процессор, благодаря чему оба аппарата обладают одинаковым объемом операций.
В настоящей главе рассматриваются операции, реализуемые этими аппаратами. Перечень операций приводится в разделе 14.1. Следующий раздел посвящен логическим операциям, программируемым непосредственно в устройствах автоматизации S5-150A и S5-150K. В разделе 14.3 описаны функции памяти, а в разделе 14.4 - функции времени и счета. В разделе 14.5 дается обобщенное описание цифровых функций. К цифровым функциям относятся функции загрузки и переноса, а также функции сравнения. Представление чисел, как правило, дается в шестнадцатиричной системе с фиксированной запятой (F:). В последнем разделе раскрываются возможности построения программ на основе команд языка STEP-5.

Общие замечания

Устройства автоматизации S5-150A и S5-150K при полном развитии могут иметь 1024 входа, 1024 выхода, 128 таймеров и 128 счетчиков. В общей сложности в распоряжении имеются 2048 защищенных меток. Как для входов, так и для выходов существуют регистры отображения процессов. Адресование блоков ввода/вывода происходит независимо от номера разъема блока в общей стойке. Адресование производится на блоках.

Метки, таймеры и счетчики находятся непосредственно в центральном блоке. Специальных блоков периферии для них не требуется. Устройства S5-150A и S5-150K рассчитаны на подключение аналоговых блоков ввода/вывода. В общей сложности может быть 64 канала аналоговых вводов или выводов.

I4.I Запас операций

Логические операции			
U	-		операция И, опрос на состояние "I"
O	-		операция ИЛИ, опрос на состояние "I"
UN	-		операция И, опрос на состояние "0"
ON	-		операция ИЛИ, опрос на состояние "0"
	E	0.0-127.7	входа
	A	0.0-127.7	выхода
	M	0.0-255.7	меток
	T	0 -127	таймера
	Z	0 -127	счетчика
U(			операция И над выражением в скобках
O(			операция ИЛИ над выражением в скобках
)			скобку закрыть
O			операция ИЛИ над функцией И
Функции памяти			
S	-		установка
R	-		сброс
=	-		присвоение
	E	0.0-127.7	входа
	A	0.0-127.7	выхода
	M	0.0-255.7	метки
Функции времени			
SI	-		запуск таймера в режиме короткого импульса
SV	-		запуск таймера в режиме продленного импульса
SE	-		запуск таймера в режиме задержки включения
SS	-		запуск таймера в режиме задержки включения с запоминанием
SA	-		запуск таймера в режиме задержки отключения
R	-		сброс
	T	0-127	уставки времени

Функции сравнения

!=	-		сравнение на "равно"
<	-		сравнение на "меньше"
>	-		сравнение на "больше"
<>	-		сравнение на "неравно"
<=	-		сравнение на "меньше-равно"
>=	-		сравнение на "больше-равно"
	EB	0-127	байта входов
	EW	0-126	слова входов
	AB	0-127	байта выходов
	AW	0-126	слова выходов
	MB	0-255	байта меток
	MW	0-254	слова меток
	DR	0-255	правого байта данных
	DL	0-255	левого байта данных
	DW	0-255	слова данных
	PB	0-255	байта периферии
	PW	0-254	слова периферии
	T	0-127	параметра времени
	Z	0-127	параметра счета
	K	-32768 +32767	константы
!=C	-		закодированное сравнение на "равно"
<C	-		закодированное сравнение на "меньше"
>C	-		" " "больше"
<>C	-		" " "неравно"
<=C	-		" " "меньше-равно"
>=C	-		" " "больше-равно"
	T	0-127	параметра времени
	Z	0-127	параметра счета

Организационные функции

SPA	-		абсолютный вызов
SPB	-		условный вызов
	PB	0-255	блока программы
	FB	0-255	блока функций
A	DB	0-255	вызов блока данных
BE			абсолютное окончание блока
BEB			условное окончание блока
STP			стоп

Функции счета

ZV	–		Zählen vorwärts	Прямой счет
ZR	–		Zählen rückwärts	обратный счет
S	–		Setzen	установка
R	–		Rücksetzen	сброс
	Z	0–127	eines Zählers	счетчика

Функции загрузки и переноса

L	–		Laden	загрузка
T	–		Transferieren	перенос
	EB	0–127	eines Eingangsbytes	байта входов
	EW	0–126	eines Eingangsworts	слова входов
	AB	0–127	eines Ausgangsbytes	байта выходов
	AW	0–126	eines Ausgangsworts	слова выходов
	MB	0–255	eines Merkerbytes	байта меток
	MW	0–254	eines Merkerworts	слова меток
	DR	0–255	eines rechten Datenbytes	правого байта данных
	DL	0–255	eines linken Datenbytes	левого байта данных
	DW	0–255	eines Datenworts	слова данных
	PB	0–255	eines Peripheriebytes	байта периферии
	PW	0–254	eines Peripherieworts	слова периферии
L	–		Laden	Загрузка
	T	0–127	eines Zeitwerts	параметра времени
	Z	0–127	eines Zählwerts	параметра счета
	KF	–32768 bis +32767	einer Konstanten als Festpunktzahl	константы как числа с фикс.зап.
	KB	0 bis 255	einer Konstanten als ein Byte	константы как байта
	KY	0 bis 255, 0 bis 255	einer Konstanten als zwei Bytes	константы как 2 байта
	KH	0000 bis FFFF	einer Konstanten als Hexamuster	константы как I6-разр.комбинации
	KM	0000 0000 0000 0000 bis 1111 1111 1111 1111	einer Konstanten als Bitmuster	константы как комбинации битов
	KC	<ASCII-Zeichen> <ASCII-Zeichen>	einer Konstanten als zwei Zeichen	константы в виде двух знаков
	KT	000.0 bis 999.3	einer Konstanten als Zeitwert	константы как параметра времени
	KZ	000 bis 999	einer Konstanten als Zählwert	константы как параметра счета
LC	–		Laden codiert	Кодированная загрузка
	T	0–127	eines Zeitwerts	параметра времени
	Z	0–127	eines Zählwerts	параметра счета

Функции сравнения

!=F	Vergleich auf „Gleich" (16 Bits)	на равенство /I6 битов/
<F	Vergleich auf „Kleiner" nach 16-Bit-Festpunktcharakteristik	на меньше как I6-разр.ч.с„ф.з
>F	Vergleich auf „Größer" nach 16-Bit-Festpunktcharakteristik	на больше
><F	Vergleich auf „Ungleich" (16 Bits)	на неравенство /I6 битов/
<=F	Vergleich auf „Kleiner-Gleich" nach 16-Bit-Festpunktcharakteristik	на меньше-равно
>=F	Vergleich auf „Größer-Gleich" nach 16-Bit-Festpunktcharakteristik	на больше равно как I6-разр. число с фикс.запятой

Арифметические функции

+F	Addiere 16-Bit-Festpunktzahlen	сложение I6-разр. ч. с фикс.зап.
–F	Subtrahiere 16-Bit-Festpunktzahlen	вычитание I6-разр. ч. с фикс.зап.

Организационные функции

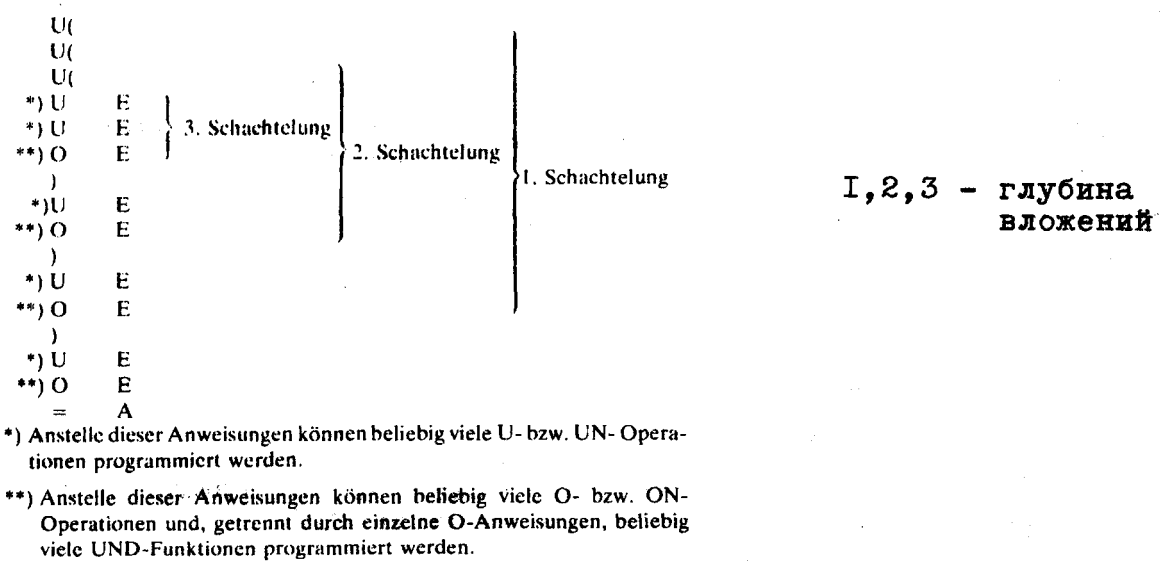
SPA	–		absoluter Aufruf	абсолютный вызов
SPB	–		bedingter Aufruf	условный вызов
	PB	0–255	eines Programmbausteins	программного блока
	FB	0–255	eines Funktionsbausteins	функционального блока
	SB	0–255	eines Schrittbausteins	шагового блока
A	DB	0–255	Aufruf eines Datenbausteins	вызов блока данных
BE			Bausteinende	конец блока
BEB			bedingtes Bausteinende	условный конец блока
STP			Stopp	стоп

14.2 Логические операции

В устройствах автоматизации 150А и 150К допустимы все логические операции, описанные в главе 2.

Глубина суперпозиционирования /вложений/ выражений в скобках составляет три ступени. Ниже показаны непосредственно программируемые логические операции:

Таблица команд



*) Anstelle dieser Anweisungen können beliebig viele U- bzw. UN- Operationen programmiert werden.

**) Anstelle dieser Anweisungen können beliebig viele O- bzw. ON-Operationen und, getrennt durch einzelne O-Anweisungen, beliebig viele UND-Funktionen programmiert werden.

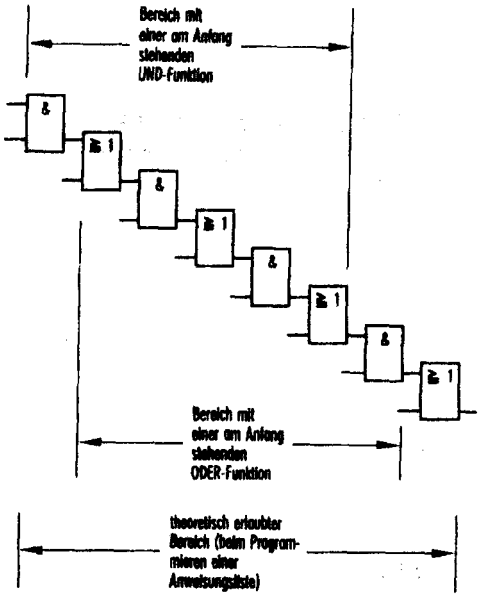
Вместо этой команды можно программировать произвольное количество операций И или И-НЕ.

Вместо этой команды можно программировать произвольное количество операций ИЛИ или ИЛИ-НЕ, а также, через разделение отдельными командами ИЛИ, произвольное число функций И.

Функциональная схема

При программировании в виде функциональной схемы можно пользоваться всей имеющейся шириной дисплея. Начинать можно как с функции И, так и с функции ИЛИ.

Пример:

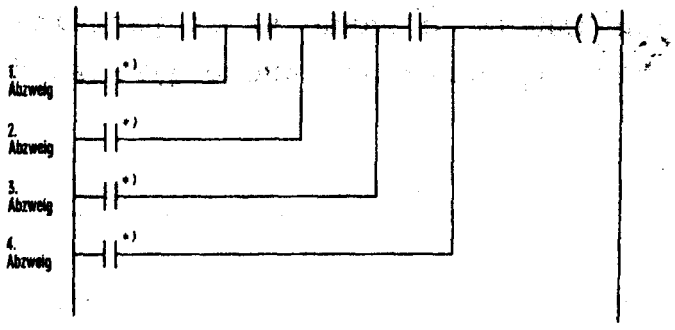


Область, начинающаяся с функции И

Область, начинающаяся с функции ИЛИ

Теоретически разрешенная область /при программировании таблицы команд/

Контактная схема



I,2,3,4 - ответвления

Вместо изображенных контактов можно последовательно включать другие контакты /на всю ширину экрана дисплея/.

*) Вместо этих контактов можно включать другие отдельные
или последовательно соединенные контакты параллельно к главной ветви.

Команда "ИЛИ скобку открыть 0(

В чисто логических соотношениях функция "открыть скобки" всегда связана функцией И, так как "открыть скобки" пишется только тогда, когда функцию ИЛИ нужно поставить перед функцией И. Язык программирования STEP-5 позволяет также использовать скобки для промежуточного хранения логического результата /см.раздел 3.3/. В таком случае функция "открыть скобки" может встречаться с функцией ИЛИ. Тогда пишется команда "0(". В разделе 13.2 приводится пример к этому пункту.

14.3 Функции памяти

В устройствах автоматизации S5-150A и S5-150K допустимы все функции памяти, описанные в 3 главе. Оба устройства, как и S5-130W, работают с регистрами отображения процессов для входов и выходов.

Область меток в этих устройствах автоматизации в основном делается защищенной. При подготовке программ следует обращать внимание, что при повторном запуске прибора /например, после восстановления напряжения/ метки сохраняют свое "старое" состояние. В определенных случаях в стандартных операциях повторного запуска можно предусматривать сброс некоторых или всех использованных меток /организационные блоки 0B 21 и 0B 22/.

Для записи незащищенных промежуточных результатов использовать отображение процесса на выходах нельзя. Программное воздействие тех выходов в регистре отображения процесса, которые не имеют присвоенных им блоков вывода, будет зарегистрировано процессором и интерпретировано им как сбой. Эта мера предусмотрена для защиты от ошибок в программирование /или от неисправных блоков/ и охватывает также входы.

Запоминание промежуточных результатов с помощью ЗУ скобок

Внутри выражений в скобках можно программировать не только логические операции, но и все основные функции языка программирования STEP-5. Однако нужно помнить, что выражение в скобках, как правило, заканчивается операцией "скобку закрыть".

С операцией "скобку открыть" процессор запоминает полученный к этому моменту логический результат и начинает новый шаг в логической связи. Результат, который был при операции "скобку закрыть" сопрягается с записанным результатом операции.

Пример:

```
O    E    22.0
O    E    18.3
O(
U    E    20.4   }
U    E    47.1   }
S    M    103.4  }  функция памяти
                 }
O    E    33.7   }
O    E    28.5   }
R    M    103.4  }
U    M    103.4     опрос ЗУ
)                   завершение выражения в скобках
O    E    31.6
=    A    50.3
```

Состояние метки M 103.4 сопрягается с состояниями входов E 22.0, E 18.3 и E 31.6 по функции ИЛИ и присваивается выходу A 50.3.

14.4 Функции времени и счета

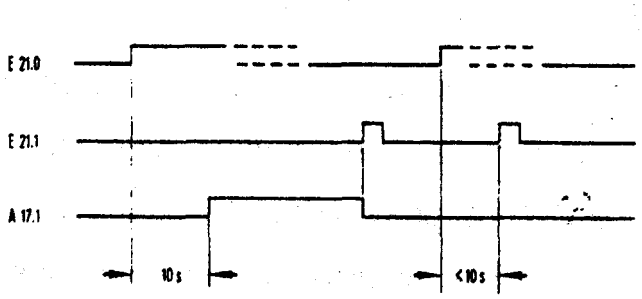
В устройствах автоматизации S5-150A и S5-150K можно программировать 128 таймеров и 128 счетчиков. Дополнительных блоков периферии для реализации Функций времени и счета не требуется.

Как таймер/ так и счетчик представляют собой 16-разрядное слово в этой области операндов. Таймер или счетчик в полном объеме программируются функциями, описанными в главах 4 и 5.

Пример:

Спустя 10 секунд после появления на входе E 21.0 состояния «1» должен установиться выход A 17.1, независимо от того, будет ли к этому моменту на входе E 21.0 состояние «1» или «0». Через вход E 21.1 выход должен быть снова сброшен или пока не устанавливаться.

Импульсная диаграмма



Программа

```
U    E    21.0
L    KT   100.1    загрузка параметра времени IO сек.
SS   T    6        запуск таймера T6 в режиме задержки включения
                   IO секунд с запоминанием
U    E    21.1
R    T    6        сброс таймера
U    T    6
=    A    17.1     опрос таймера
```

Пример:

Стоящее в слове входов EW52 закодированное в двоично-десятичном коде значение /3 декады, например, какого-либо цифрового задатчика/ нужно загрузить в двоичном коде в слово данных DW 37.

Преобразование кода десятичного в двоичный может быть простейшим образом реализовано стандартной микропрограммой, если преобразуемое значение не превышает трех декад /000 - 999/. Для этого можно использовать свободный счетчик /или же таймер/. Счетчик загружается /устанавливается/ двоично-десятичным значением. Параметр счета после этого загружается в двоичном коде в аккумулятор и подвергается дальнейшей обработке.

Так как счетчик устанавливается только при нарастающем /переднем/ фронте управляемого логического результата, необходимо всегда перед входом установки менять логический результат. В нашем примере это делает запрограммированная перед счетчиком пересчетная схема, которая меняет свое состояние при каждом прогоне программы. Как следствие, при каждом втором прогоне программы происходит актуализация параметра, стоящего в слове данных.

Программа

```
UN   M    250.1  }
=    M    250.1  }  пересчетная схема
L    EW   52       загрузка параметром в двоично-десятичном коде
S    Z    103      установка счетчика параметром в двоично-десятичном
                   коде
L    Z    103    }  перенос двоично-десятичного параметра в слово данных
T    DW   37     }
```

14.5 Цифровые функции

В устройствах автоматизации S5-150A и S5-150K допускается прямое программирование функций загрузки, переноса и сравнения. Внутреннее представление чисел, как правило, шестнадцатеричное с фиксированной запятой /см.раздел 7.2/.

Функции загрузки и переноса

Функции загрузки и переноса разрешается выполнять только с операндами, адресуемыми по байтам или по словам. Загрузка и перенос двойными словами не допускается. Область операндов Q - "расширенная периферия процесса" в устройствах S5-150A и S5-150K не предусматривается.

Функции загрузки и переноса в устройствах автоматизации S5-150A и S5-150K выполняются независимо от результата логической операции и влияния на него не оказывают.

Функции сравнения в устройствах автоматизации S5-150A и S5-150K разрешается не последственное программирование всех логических операций, описанных в 8 главе. Сравнение ограничивается операндами по байтам или словам с представлением чисел в шестнадцатиричной системе с фиксированной запятой.

Результат функции сравнения может не только управлять функциями памяти или участвовать в дальнейших логических операциях, но и выполнять условные вызовы блоков и условное завершение блоков /см.примеры в разделе 14.5/.

Вычислительные функции

В устройствах автоматизации 150А и 150К допустимы только математические функции "сложение по характеристике чисел с фиксированной запятой" +Р и вычитание по характеристике чисел с фиксированной запятой". Другие вычислительные операции могут выполняться в функциональных блоках, где эти операции имитируются программным путем.

СИМЕНС АГ предоставляет пользователям так называемые "стандартные функциональные блоки". Эти блоки описаны во втором томе книги "Программирование управляющих устройств на языке STEP-5". В частности, для вычислительных функций имеются следующие стандартные функциональные блоки:

FB 37	ADD: B4	4-Dekaden-BCD-Addierer	блоки сложения
FB 1	ADD: B8	8-Dekaden-BCD-Addierer	
FB 2	ADD: 16	16-Bit-Dualaddierer	
FB 3	ADD: 32	32-Bit-Dualcodierer	
FB 4	SUB: B4	4-Dekaden-BCD-Subtrahierer	блоки вычитания
FB 5	SUB: B8	8-Dekaden-BCD-Subtrahierer	
FB 6	SUB: 16	16-Bit-Dualsubtrahierer	
FB 7	SUB: 32	32-Bit-Dualsubtrahierer	
FB 8	MUL: B4	4-Dekaden-BCD-Multiplizierer	блоки умножения
FB 9	MUL: B8	8-Dekaden-BCD-Multiplizierer	
FB 10	MUL: 16	16-Bit-Dualmultiplizierer	
FB 11	MUL: 32	32-Bit-Dualmultiplizierer	
FB 12	DIV: B4	4-Dekaden-BCD-Dividierer	блоки деления
FB 13	DIV: B8	8-Dekaden-BCD-Dividierer	
FB 14	DIV: 16	16-Bit-Dualdividierer	
FB 15	DIV: 32	32-Bit-Dualdividierer	
FB 16	RAD: B4	4-Dekaden-BCD-Radizierer	блоки извлечения корня
FB 17	RAD: B8	8-Dekaden-BCD-Radizierer	
FB 18	RAD: 16	16-Bit-Dualradizierer	
FB 19	RAD: 32	32-Bit-Dualradizierer	

Als Codewandler stehen zur Verfügung:

В качестве кодовых преобразователе имеются следующие блоки:

FB 20	COD: B4	Codewandler 4-Dekaden-BCD in 16-Bit-Festpunkt
FB 21	COD: B8	Codewandler 8-Dekaden-BCD in 32-Bit-Festpunkt
FB 22	COD: 16	Codewandler 16-Bit-Festpunkt in 6-Dekaden-BCD
FB 23	COD: 32	Codewandler 32-Bit-Festpunkt in 10-Dekaden-BCD

14.6 Структура программ

С помощью устройств автоматизации S5-150A и S5-150K можно программировать 255 блоков программ, 255 блоков функций и 255 блоков данных.

В качестве организационных блоков в устройствах S5-150A и S5-150K допущены организационный блок 0B 1 для циклической обработки программ, организационные блоки OB2 - OB9 для обработки программ по прерываниям и организационные блоки 0B 13 - 0B 18 - для обработки программ с управлением по времени /см.раздел 9.2/. Кроме того, в распоряжении пользователя имеются следующие организационные блоки:

Организационный блок	Основание для обращения
0B I9	вызов незагруженного блока
0B 20	первый запуск
0B 2I	ручной повторный запуск
0B 22	автоматический повторный запуск /после восстановления напряжения в сети/
0B 23	задержка квитирования при единичном обращении к периферии процесса
0B 24	задержка квитирования при актуализации отображения процесса
0B 25	ошибка в адресовании
0B 26	превышение времени цикла
0B 27	ошибка в замещении
0B 28	задержка квитирования в байте входов EB 0

Блоки функций в устройствах автоматизации S5-150A и S5-150K можно параметрировать. Т.е. во время вызова им можно непосредственно присваивать операнды, с которыми должны работать блоки функций /см. Также "Программирование.том 3".

Здесь, так же как и в устройстве автоматизации S5-130W, могут использоваться интерфейсные метки. Пример и более подробные объяснения приведены в разделе 15.6.

15 ПРОГРАММИРОВАНИЕ УСТРОЙСТВА АВТОМАТИЗАЦИИ S5-150S

Устройство автоматизации S5-150S является самым крупным в системе СИМАТИК S5. Этот прибор работает со всем объемом операций, предусмотренным в языке STEP-5. Он: приводится в разделе 14.1.

В логических операциях /раздел 14.2/ и функциях памяти /раздел 14.3/ в дополнение к операндам входов, выходов и меток двоичное Адресование имеет также область операндов данных D. В разделе 14.4 описываются операции времени и счета, в разделе 14.5-цифровые функции. В устройстве S5-150S операнды можно давать в виде двойных слов /32 бита/. Представление чисел также может быть как 16-разрядным с фиксированной запятой, так и 32-разрядным со скользящей запятой.

В последнем разделе приводится структура программ, реализуемых в этом устройстве автоматизации с помощью команд языка STEP-5.

Общие замечания

При полном развитии устройство автоматизации S5-150S может работать с 1024 входами, 1024 выходами, 255 таймерами и 255 счетчиками. Оно располагает в общей сложности 2048 защищенными метками.

Как для входов, так и для выходов существуют регистры отображения процесса. Адресование блоков ввода/вывода происходит независимо от номера разъема блока в общей стойке. Адресование выполняется непосредственно на блоке.

Метки, таймеры и счетчики входят в состав центрального блока. Специальных блоков периферии для них не требуется. Устройство автоматизации S5-110S рассчитано на подключение аналоговых блоков ввода/вывода. В общей сложности напрямую можно реализовать 64 канала ввода или вывода /аналоговых/. Благодаря второй шине периферии можно увеличить количество блоков на периферии процесса /цифровые вводы, цифровые выводы, аналоговые вводы и аналоговые выводы/. Для этого в языке программирования STEP-5 предусматривается область операндов Q -"Расширенная периферия". Цифровые входы и выходы из расширенной области регистров отображения процесса не имеют.

15.1 Перечень операций

Логические операции

U	-		операция И, опрос на состояние "I"
O	-		операция ИЛИ, опрос на состояние "I"
UN	-		операция И, опрос на состояние "0"
ON	-		операция ИЛИ, опрос на состояние "0"
	E	0.0-127.7	входа
	A	0.0-127.7	выхода
	M	0.0-255.7	меток
	T	0 -255	таймера
	Z	0 -255	счетчика
	D	0.0-255.15	данных
U(			операция И над выражением в скобках
O(			операция ИЛИ над выражением в скобках
)			скобку закрыть
O			операция ИЛИ над функцией И

Функции памяти

S	-		установить
R	-		сбросить
=	-		присвоить
	E	0.0-127.7	вход
	A	0.0-127.7	выход
	M	0.0-255.7	метку
	D	0.0-255.15	данные

Функции времени

SI	-		запуск таймера в режиме короткого импульса
SV	-		запуск таймера в режиме удлиненного импульса
SE	-		запуск таймера в режиме задержки включения
SS	-		запуск таймера в режиме задержки включения с запоминанием
SA	-		запуск таймера в режиме задержки отключения
R	-		сброс
	T	0-255	таймера

Функции счета

ZV	-		прямой счет
ZR	-		обратный счет
S	-		установка
R	-		сброс
	Z	0-255	счетчика

Функции загрузки и переноса

Lade- und Transferfunktionen

L	–		Laden	загрузка
T	–		Transferieren	перенос
	EB	0–127	eines Eingangsbytes	байта входов
	EW	0–126	eines Eingangsworts	слова входов
	ED	0–124	eines Eingangsdoppelworts	двойного слова входов
	AB	0–127	eines Ausgangsbytes	байта выходов
	AW	0–126	eines Ausgangsworts	слова выходов
	AD	0–124	eines Ausgangsdoppelworts	двойного слова выходов
	MB	0–255	eines Merkerbytes	байта меток
	MW	0–254	eines Merkerworts	слова меток
	MD	0–252	eines Merkerdoppelworts	двойного слова меток
	DR	0–255	eines rechten Datenbytes	правого байта данных
	DL	0–255	eines linken Datenbytes	левого байта данных
	DW	0–255	eines Datenworts	слова данных
	DD	0–254	eines Datendoppelworts	двойного слова данных
	PB	0–255	eines Peripheriebytes	байта периферии
	PW	0–254	eines Peripherieworts	слова периферии
	QB	0–255	eines Peripheriebytes aus dem erweiterten Bereich	байта периф.из расшир.области
	QW	0–254	eines Peripherieworts aus dem erweiterten Bereich	слова периф.из расшир.области
	KF	–32768 bis +32767	einer Konstanten als Festpunktzahl	константы как числа с ф.з.
	KB	0 bis 255	einer Konstanten als ein Byte	константы как байта
	KY	0 bis 255, 0 bis 255	einer Konstanten als zwei Bytes	константы в виде 2 байтов
	KH	0000 bis FFFF	einer Konstanten als Hexamuster	константы в I6-выражении
	KM	0000 0000 0000 0000 bis 1111 1111 1111 1111	einer Konstanten als Bitmuster	константы как комб.битов
	KC	<ASCH-Zeichen> <ASCH-Zeichen>	einer Konstanten als zwei Zeichen	константы в виде 2 знаков
	KT	000.0 bis 999.3	einer Konstanten als Zählwert	константы как параметра счета
	KZ	000 bis 999	einer Konstanten als	
	KG	±1701411±39	einer Konstanten als Gleitpunktzahl	константы в виде числа со скользящей запятой
	–		Laden	загрузка
LC	–		Laden codiert	кодированная загрузка
	T	0–255	eines Zeitwerts	параметра времени
	Z	0–255	eines Zählwerts	параметра счета

Функции сравнения

Vergleichsfunktionen

!=F	Vergleich auf „Gleich" (16 Bits)	сравнение на равенство /16 битов/
<F	Vergleich auf „Kleiner" nach der 16-Bit-Festpunktcharakteristik	сравнение на "меньше"по характерист. 16-разр.числа с фикс.запятой
>F	Vergleich auf „Größer" nach der 16-Bit-Festpunktcharakteristik	сравнение на "больше" по той же характ.
><F	Vergleich auf „Ungleich" (16 Bits)	сравнение на неравенство /16 битов/
<=F	Vergleich auf „Kleiner-Gleich" nach der 16-Bit-Festpunktcharakteristik	сравнение на "меньше-равно" по характ. 16-разр.числа с фикс.запятой
>=F	Vergleich auf „Größer-Gleich" nach der 16-Bit-Festpunktcharakteristik	сравнение на"больше-равно" по той же хар.
!=D	Vergleich auf „Gleich" (32 Bits)	сравнение на равенство /32 бита/
<D	Vergleich auf „Kleiner" nach der 32-Bit-Festpunktcharakteristik	сравнение на "меньше" по хар.32-разр. числа с фикс.зап.
>D	Vergleich auf „Größer" nach der 32-Bit-Festpunktcharakteristik	сравнение на "больше" по той же характ.
><D	Vergleich auf „Ungleich" (32 Bits)	сравнение на неравенство /32 бита/
<=D	Vergleich auf „Kleiner-Gleich" nach der 32-Bit-Festpunktcharakteristik	сравнение на "меньше-равно"по той же хар.
>=D	Vergleich auf „Größer-Gleich" nach der 32-Bit-Festpunktcharakteristik	сравнение на "больше-равно"по той же хар.
!=G	Vergleich auf Gleich (32 Bits)	сравнение на равенство /32 бита/
<G	Vergleich auf „Kleiner" nach der 32-Bit-Gleitpunktcharakteristik	сравнение на "меньше" по характ.32-разр. числа со скользящей запятой
>G	Vergleich auf „Größer" nach der 32-Bit-Gleitpunktcharakteristik	сравнение на"больше" по той же характ.
><G	Vergleich auf „Ungleich" (32 Bits)	сравнение на неравенство /32 бита/
<=G	Vergleich auf „Kleiner-Gleich" nach der 32-Bit-Gleitpunktcharakteristik	сравнение на "меньше-равно" по характ. 32-разр.числа со скользящей запятой
>=G	Vergleich auf „Größer-Gleich" nach der 32-Bit-Gleitpunktcharakteristik	сравнение на "больше-равно" по той же хар.

Вычислительные функции

Rechenfunktionen

+F	Addiere Festpunktzahlen	сложение чисел с фикс.запятой
-F	Subtrahiere Festpunktzahlen	вычитание чисел с фикс.запятой
XF	Multipliziere Festpunktzahlen	умножение чисел с фикс.запятой
:F	Dividiere Festpunktzahlen	деление чисел с фикс.запятой
+G	Addiere Gleitpunktzahlen	сложение чисел со скольз.запятой
-G	Subtrahiere Gleitpunktzahlen	вычитание " " "
XG	Multipliziere Gleitpunktzahlen	умножение " " "
:G	Dividiere Gleitpunktzahlen	деление " " "

Organisatorische Funktionen — Организационные функции

SPA	–		absoluter Aufruf	абсолютный вызов
SPB	–		bedingter Aufruf	условный вызов
	PB	0–255	eines Programmbausteins	программного блока
	FB	0–255	eines Funktionsbausteins	функционального блока
	SB	0–255	eines Schrittbausteins	шагового блока
A	DB	0–255	Aufruf eines Datenbausteins	вызов блока данных
BE			Bausteinende	конец блока
BEB			bedingtes Bausteinende	условный конец блока
STP			Stopp	стоп

15.2Логические операции

В устройстве автоматизации S5-150S раз решены все логические операции, описанные во второй главе.

Глубина вложений в выражениях в скобках составляет уровней. Логические операции, непосредственно программируемые благодаря этом в виде таблицы команд, Функциональной или контактной схемы такие же, как ив устройстве автоматизации S5-130W /см.раздел Л3.2/.

В чисто логических операциях функция "скобку открыть" всегда связана с функцией "И" так как "скобку открыть" пишется только тогда, когда перед функцией И нужно поставить функцию ИЛИ. Язык программирования STEP5 позволяет также пользоваться скобкам для промежуточного запоминания Логических результатов /см.раздел 3.3/. -В таких случаях может также встречаться сочетание функции "скобку открыть" с функцией ИЛИ. Тогда пишут команду "0(". В разделе 13.2 приводится пример к этому пункту. В устройстве автоматизации в логических операциях разрешена область операндов данных D. Обращаться можно к каждому отдельному биту слова данных. При этом предварительно нужно обращать внимание на правильный вызов блока данных.

Пример:

Бит № 6 слова данных DW17 и биты 0 и № 12 слова Данных DW28 нужно соотнести по функции И. Результат операции присвоить выходу А 87.3.

Программа:

```
U D 17.6
U D 28.0
U D 28.12
= A 87.3
```

15.3 Функции памяти

В устройстве автоматизации разрешены все функции памяти, описанные в разделе 3. Регистры отображения процессов существуют как для входов, так и для выходов.

Область меток в этом устройстве, как правило, делается защищенной. При составлении программы следует учитывать, что при повторном запуске прибора /напр. после восстановления напряжения в сети/ метки сохраняют "старое" состояние. В случае необходимости в стандартных операциях повторного запуска /организационные блоки OB21 и 0B 22/ можно предусмотреть сброс некоторых или всех использованных меток.

Для запоминания не сохраняющихся промежуточных результатов регистром отображения процесса на выходах пользоваться нельзя. Программное воздействие тех выходов в регистре отображения процесса, которые не имеют присвоенных им блоков вывода, будет зарегистрировано процессором и интерпретировано им как сбой. Эта мера предусмотрена для защиты от ошибок в программировании /или от использования неисправных блоков/ и охватывает также входы.

Запоминание промежуточных результатов с помощью ЗУ скобок.

Внутри выражений в скобках можно программировать не только логические операции, но и все основные функции языка программирования STEP-5. Однако нужно помнить, что выражение в скобках, как правило, заканчивается операцией "скобку закрыть".

С операцией "скобку открыть" процессор запоминает полученный к этому моменту логический результат и начинает новый шаг в логической связи. Результат, который был при операции "скобку закрыть", сопрягается с записанным результатом операции. Пример к этому случаю приводится в разделе 14.3.

15.4 Функции времени и счета

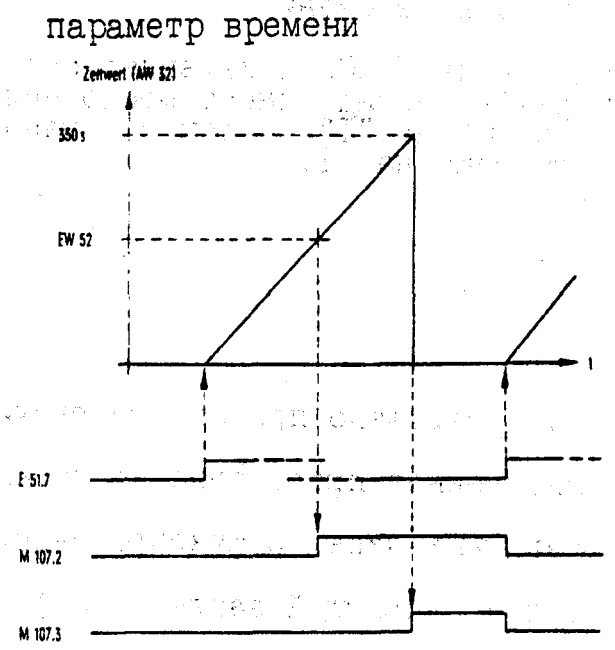
В устройстве автоматизации S5-150S можно программировать 255 таймеров и 255 счетчиков. Дополнительных блоков периферии для реализации функций времени и счета не требуется.

Как таймер, так и счетчик представляют собой 16-разрядное слово в этой области операндов. Таймер или счетчик в полном объеме программируются функциями, описанными в главах 4 и 5.

Пример:

Через выход E 51.7 требуется запустить "вперед" считающий таймер и вывести его показания /в секундах/ в слово выходов AW32. После того, как слово входов EW53 достигнет устанавливаемого значения, должна произойти установка метки M 107.2. После достижения жестко установленной продолжительности времени 350 секунд должна установиться метка M 107.3 и прекратиться обработка таймера. С установлением состояния «1» на. входе E 51.7 происходит новый запуск таймера и сброс меток M 107.2 и M 107.3.

Функциональная диаграмма



Временная функция реализуется с помощью счетчика Z73. Запуск и опрос счетчика производится по циклической программе /напр. в программном блоке PB 61/. Отсчет времени вверх б секундах ведется с помощью управляемой по времени обработки организационного блока 0B 15 /ритм обработки 0,5 с/. Распределение битов в словах входов и выходов приведено в разделах 5.1 и 6.3.

Программа в блоке PB 6I

```
      .
      .
      .
      U    E    51.7
      UN   M    200.0
      S    M    200.0     Обработка времени ведется,
      R    M    107.2     если установлена метка M 200.0
      R    M    107.3
      U    M    107.3
      R    M    200.0

      F:C  Z    73        Если счетчик Z73 достиг величины, установленной
      >=   EW   52        в слове входов EW52,
      S    M    107.2     установится метка M 107.2.

      F:   Z    73        При достижением счетчиком Z73 постоянной величины
      >=   K    350       350 устанавливается метка M I07.3 и
      S    M    107.3     счетчик сбрасывается.
      R    Z    73

      LC   Z    73        Текущее значение времени
      T    AW   32        постоянно индицируется в слове выходов AW 32
                          в двоично-десятичном коде.
      .
      .
      .
```

Программа в OB I5 /дискретность обработки 0,5 с/

```
U    M   200.0      Разрешение на обработку времени
UN   M   200.1  |   Каждые 0,5 секунд метка M 200.I изменяет свое
=    M   200.1  |   состояние. Один раз в секунду имеет место поло-
ZV   Z   73         жительная смена /с "0" на "I"/, причем показа-
BE                  ния счетчика возрастают на +I.
```

15.5 Цифровые функции

В устройстве автоматизации S5-150S разрешено прямое программирование всех цифровых функций. Внутреннее представление чисел может происходит как в виде 16-разрядных чисел с фиксированной запятой (F) или 32-разрядных чисел с плавающей запятой (G).

Функции загрузки и переноса

Функции загрузки и переноса реализуются в полном объеме. Они не зависят от логического результата и влияния на него не оказывают.

В устройстве-автоматизации S5-150S с помощью второго интерфейса расширения в центральном устройстве к последнему можно подключать вторую шину периферии. Обращение к блокам ввода и вывода, подключенным к этой шине периферии, происходит "через операнды Q /периферия расширенной области/. Таким образом можно охватывать управлением дополнительные цифровые и аналоговые блоки ввода и вывода.

Цифровые входы и выходы из расширенной области не располагают регистрами отображения процесса. Если над этими операндами требуется произвести логические операции, их нужно загрузить в область меток или данных, где и произойдет их дальнейшая обработка.

Пример:

Состояние сигнала на входе E 17.3 из расширенной области требуется связать по функции И с состоянием метки M 51.0. Результат присвоить выходу A 23.0 в расширенной области.

Программа

```
L    OB   17      Загрузка байта входов в
T    MB   198     байт меток
U    M    198.3   Состояние сигнала на входе E I7.3 из расширенной обл.

U    M    51.0
=    M    199.0   Установка метки, отвечающей выходу A 23.7 из рас-
                  ширенной области.
...               После того, как все метки этого байта установлены,
                  этот байт меток переносится в расширенную область
L    MB   199     периферии.
T    OB   23
```

В устройствах автоматизации S5-150S **константы могут загружаться в виде чисел со скользящей /плавающей/ запятой. Они изображаются 7-значной мантиссой и 2-значным экспонентом и разделяются знаком экспонента.**

Пример:

```
L    KG   +55+01
          y   x
```

$KG = 0.y \cdot 10^{x} = 0.55 \cdot 10^{1} = 5.5$

Числа могут задаваться в следующих пределах:

KG_{max} +1701411+39
KG_{min} −1701411+39

Функции сравнения

В устройстве автоматизации S5-15CS разрешается непосредственное программирование всех функций сравнения, описанных в главе 8. Сравнения можно производить с цифрами, представленными в 16-разрядной системе с фиксированной запятой и 32-разрядной системе со скользящей запятой.

С помощью результата функции сравнения можно не только управлять функциями памяти или производить логические операции, но и выполнять условные вызовы блоков и условное окончание блоков /см.пример в разделе 13.б/.

Вычислительные функции

В устройстве автоматизации разрешается прямое программирование всех вычислительных функций, описанных в главе 9. Вычислительные функции выполняются как с числами с фиксированной запятой, 16 разрядов, так и с числами со скользящей запятой, 32 разряда. Запас операций дополняющих функций, которые описаны в 3 томе настоящей книги, позволяет входить в стеки ЗУ вычислительных операций, благодаря чему внутри функциональных блоков можно также выполнять все арифметические действия.

15.6 Структура программ

С помощью устройства автоматизации S5-150S можно программировать 255 блоков программ, 255 блоков функций и 255 блоков данных. В качестве организационных блоков в устройстве S5-150S допущен организационный блок 0В 1-для циклической обработки программ, и организационные блоки 0В 2 - 0В 9 - для обработки программ с управлением по прерываниям /см.раздел 9.2/. При обработке с управлением по времени могут подключаться организационные блоки 0В 10 -0В 12 /через перемычку в процессоре/. Организационные блоки 0В 13 - 0В 18 обрабатываются с той же дискретностью времени, что и в устройствах автоматизации S5-150A и S5-150K.

Кроме того, в распоряжении пользователя имеются следующие организационные блоки:

Организационный блок	Основание для обращения
0В I9	вызов незагруженного блока
0В 20	первый запуск
0В 2I	ручной повторный запуск
0В 22	автоматический повторный запуск /после восстановления напряжения в сети/
0В 23	задержка квитирования при единичном обращении к периферии процесса
0В 24	задержка квитирования при актуализации отображения процесса
0В 25	ошибка в адресовании
0В 26	превышение времени цикла
0В 27	ошибка в замещении
0В 28	задержка квитирования в байте входов ЕВ 0

Блоки функций в устройстве автоматизации S5-150S можно параметрировать. Т.е. во время вызова им можно непосредственно присваивать операнды, с которыми они должны работать /см.также "Программирование, том 3".

Здесь, так же как и в устройстве автоматизации S5-130W, могут использоваться интерфейсные метки. Пример и более подробные объяснения приведены в разделе 13.6.