

# Набор команд S7–200

# 6

В этой главе описаны наборы команд SIMATIC и IEC 1131 для микро-ПЛК S7–200.

## В этой главе

Соглашения, используемые для описания команд	69
Области памяти и функции S7–200	70
Битовые логические операции	72
Контакты	72
Катушки	75
Стековые операции	77
Функциональный блок с двумя устойчивыми состояниями: преимущество установки и преимущество сброса	79
Команды, связанные со временем	80
Команды для обмена данными	83
Команды чтения из сети и записи через сеть	83
Команды передачи и приема (свободно программируемый обмен данными)	88
Команды получения и установки адреса порта	97
Команды сравнения	98
Сравнение числовых величин	98
Сравнение строк	100
Команды преобразования	101
Стандартные команды преобразования	101
Команды преобразования ASCII	105
Команды преобразования строк	109
Команды кодирования и декодирования	114
Команды счета	115
Счетчик SIMATIC	115
Счетчик IEC	118
Скоростные счетчики	120
Команда вывода импульсов	135
Арифметические операции	142
Операции сложения, вычитания, умножения и деления	142
Умножение целых чисел с представлением результата в виде двойного целого числа и деление целых чисел с остатком	144
Числовые функции	145
Команды инкрементирования и декрементирования	146
Пропорционально-интегрально-дифференциальный (PID) регулятор	147
Команды прерывания	155
Логические операции	163
Операции инвертирования	163
Поразрядные логические операции И, ИЛИ и исключающее ИЛИ	164
Команды пересылки	166
Пересылка байта, слова, двойного слова или вещественного числа	166
Непосредственное чтение или запись и пересылка байта	167
Команды групповой пересылки	168
Команды управления программой	169
Условное завершение	169

Останов	169
Сброс контроля времени	169
Команды формирования программного цикла For-Next	171
Команды перехода	173
Команды для реле управления очередностью (SCR)	174
Команда управления диагностическим светодиодом	180
Команды сдвига и циклического сдвига	181
Команды сдвига вправо и сдвига влево	181
Команды циклического сдвига вправо и циклического сдвига влево	181
Команда перемещения значения бита в регистр сдвига	183
Команда перестановки байтов в слове	185
Строковые команды	186
Табличные команды	191
Ввод значения в таблицу	191
Удаление первой записи из таблицы и удаление последней записи из таблицы	192
Заполнение памяти комбинацией двоичных разрядов	194
Поиск значения в таблице	195
Таймерные команды	198
Таймерные команды SIMATIC	198
Таймерные команды IEC	203
Интервальные таймеры	205
Операции с подпрограммами	206

## Соглашения, используемые для описания команд

На рис. 6-1 представлены типичное описание команды и указания на места в тексте, где описана эта команда и ее действие. На изображении команды показан ее формат в LAD, FBD и STL. В таблице операндов приведены операнды команды и показаны допустимые типы данных, области памяти и размеры для каждого операнда.

Операнды EN и ENO и их типы данных в таблице операндов команды не представлены, так как эти операнды одинаковы для всех команд LAD и FBD.

- Для LAD: EN и ENO являются потоками сигнала и имеют тип BOOL.
- Для FBD: EN и ENO являются потоками сигнала или I, Q, V, M, SM, S, T, C, L и имеют тип BOOL.

Описание команды и операндов

Команда STL

Команды LAD и FBD

**Transmit and Receive Instructions**

The Transmit instruction (XMT) is used in Freeport mode to transmit data by means of the communication port(s).

The Receive instruction (RCV) initiates or terminates the Receive Message service. You must specify a start and an end condition for the Receive box to operate. Messages received through the specified port (PORT) are stored in the data buffer (TBL). The first entry in the data buffer specifies the number of bytes received.

**Error conditions that set ENO = 0**

- SM56.6 or SM186.6 error bit set (RCV parameter error)
- 0006 (indirect address)
- 0009 (simultaneous XMT/RCV on port 0)
- 000B (simultaneous XMT/RCV on port 1)
- S7-200 CPU not in Freeport mode

Список сбойных состояний, влияющих на ENO, и соответствующих битов специальной памяти

Table 6-11 Valid Operands for the Transmit and Receive Instructions

Inputs/Outputs	Data Type	Operands
TBL	BYTE	VB, IB, QB, MB, SB, SMB, *VD, *AC, *LD
PORT	BYTE	Constant for CPU 221, CPU 222, CPU 226: 0 for CPU 226 and CPU 226X1A: 0 or 1

**Using Freeport Mode to Control the Serial Communication Port**

You can select the Freeport mode to control the serial communication port of the S7-200 by means of the ladder program. When you select Freeport mode, your program controls the operation of the communication port. The use of the receive interrupt, the transmit interrupt, the Transmit instruction, and the Receive instruction. The communication protocol is entirely controlled by the ladder program while in Freeport mode. SMB30 (for port 0) and SMB130 (for port 1) if your S7-200 has two ports) are used to control the baud rate and parity.

The Freeport mode is disabled and normal communication is re-established (for example, programming of the S7-200) when the Freeport mode is disabled. Freeport mode is used by the device with which the S7-200 communicates while in Freeport mode.

Freeport communication is possible only when the S7-200 is in the RUN mode. Enable the Freeport mode by setting a value of 01 in the protocol select field of SMB30 (Port 0) or SMB130 (Port 1). While in Freeport mode, communication with the programming device is not possible.

**Tip**

Entering Freeport mode can be controlled using special memory bit SM0.7, which reflects the current position of the operating mode switch. When SM0.7 is equal to 0, the switch is in TERM position; when SM0.7 = 1, the operating mode switch is in RUN position. If you enable Freeport mode only when the switch is in RUN position, you can use the programming device to monitor or control the S7-200 operation by changing the switch to any other position.

Операнды для команды

Допустимые типы данных

Допустимые области памяти и размеры операндов

Рис. 6-1. Описание команд

## Области памяти и функции S7–200

Таблица 6–1. Области памяти и функции CPU S7–200

Описание	CPU 221	CPU 222	CPU 224	CPU 224XP	CPU 226
Размер программы пользователя с редактированием в режиме RUN без редактирования в режиме RUN	4096 байт 4096 байт	4096 байт 4096 байт	8192 байта 12288 байт	12288 байт 16384 байта	16384 байта 24576 байт
Размер данных пользователя	2048 байт	2048 байт	8192 байта	10240 байт	10240 байт
Регистр входов образа процесса	I0.0 – I15.7				
Регистр выходов образа процесса	Q0.0 – Q15.7				
Аналоговые входы (только чтение)	AIW0 – AIW30	AIW0 – AIW30	AIW0 – AIW62	AIW0 – AIW62	AIW0 – AIW62
Аналоговые выходы (только запись)	AQW0 – AQW30	AQW0 – AQW30	AQW0 – AQW62	AQW0 – AQW62	AQW0 – AQW62
Память переменных (V)	VB0 – VB2047	VB0 – VB2047	VB0 – VB8191	VB0 – VB10239	VB0 – VB10239
Локальная память (L) <sup>1</sup>	LB0 – LB63				
Битовая память (M)	M0.0 – M31.7				
Специальная память (SM)  только чтение	SM0.0 – SM179.7 SM0.0 – SM29.7	SM0.0 – SM299.7 SM0.0 – SM29.7	SM0.0 – SM549.7 SM0.0 – SM29.7	SM0.0 – SM549.7 SM0.0 – SM29.7	SM0.0 – SM549.7 SM0.0 – SM29.7
Таймеры Задержка включения с запоминанием	256 (T0 – T255)				
1 мс	T0, T64				
10 мс	T1 – T4 и T65 – T68				
100 мс	T5 – T31 и T69 – T95				
Задержка вкл/выкл					
1 мс	T32, T96				
10 мс	T33 – T36 и T97 – T100				
100 мс	T37 – T63 и T101 – T255				
Счетчики	C0 – C255				
Скоростные счетчики	HC0 – HC5				
Реле последовательного управления (S)	S0.0 – S31.7				
Аккумуляторные регистры	AC0 – AC3				
Переходы/Метки	0 – 255	0 – 255	0 – 255	0 – 255	0 – 255
Вызовы/Подпрограмма	0 – 63	0 – 63	0 – 63	0 – 63	0 – 127
Программы обработки прерываний	0 – 127	0 – 127	0 – 127	0 – 127	0 – 127
Нарастающие/падающие фронты	256	256	256	256	256
PID-регуляторы	0 – 7	0 – 7	0 – 7	0 – 7	0 – 7
Порты	Порт 0	Порт 0	Порт 0	Порт 0, порт 1	Порт 0, порт 1

<sup>1</sup> LB60 – LB63 зарезервированы для STEP 7-Micro/WIN, версии 3.0 и выше.

Таблица 6-2. Диапазоны операндов CPU S7-200

Метод доступа	CPU 221	CPU 222	CPU 224	CPU 224 XP	CPU 226	
Доступ к биту (байт.бит)	I	0.0 – 15.7	0.0 – 15.7	0.0 – 15.7	0.0 – 15.7	
	Q	0.0 – 15.7	0.0 – 15.7	0.0 – 15.7	0.0 – 15.7	
	V	0.0 – 2047.7	0.0 – 2047.7	0.0 – 8191.7	0.0 – 10239.7	0.0 – 10239.7
	M	0.0 – 31.7	0.0 – 31.7	0.0 – 31.7	0.0 – 31.7	0.0 – 31.7
	SM	0.0 – 165.7	0.0 – 299.7	0.0 – 549.7	0.0 – 549.7	0.0 – 549.7
	S	0.0 – 31.7	0.0 – 31.7	0.0 – 31.7	0.0 – 31.7	0.0 – 31.7
	T	0 – 255	0 – 255	0 – 255	0 – 255	0 – 255
	C	0 – 255	0 – 255	0 – 255	0 – 255	0 – 255
	L	0.0 – 63.7	0.0 – 63.7	0.0 – 63.7	0.0 – 63.7	0.0 – 63.7
Доступ к байту	IB	0 – 15	0 – 15	0 – 15	0 – 15	
	QB	0 – 15	0 – 15	0 – 15	0 – 15	
	VB	0 – 2047	0 – 2047	0 – 8191	0 – 10239	0 – 10239
	MB	0 – 31	0 – 31	0 – 31	0 – 31	0 – 31
	SMB	0 – 165	0 – 299	0 – 549	0 – 549	0 – 549
	SB	0 – 31	0 – 31	0 – 31	0 – 31	0 – 31
	LB	0 – 63	0 – 63	0 – 63	0 – 63	0 – 63
	AC	0 – 3	0 – 3	0 – 3	0 – 255	0 – 255
	KB (константа)					
	Доступ к слову	IW	0 – 14	0 – 14	0 – 14	0 – 14
QW		0 – 14	0 – 14	0 – 14	0 – 14	
VW		0 – 2046	0 – 2046	0 – 8190	0 – 10238	0 – 10238
MW		0 – 30	0 – 30	0 – 30	0 – 30	0 – 30
SMW		0 – 164	0 – 298	0 – 548	0 – 548	0 – 548
SW		0 – 30	0 – 30	0 – 30	0 – 30	0 – 30
T		0 – 255	0 – 255	0 – 255	0 – 255	0 – 255
C		0 – 255	0 – 255	0 – 255	0 – 255	0 – 255
LW		0 – 62	0 – 62	0 – 62	0 – 62	0 – 62
AC		0 – 3	0 – 3	0 – 3	0 – 3	0 – 3
AIW		0 – 30	0 – 30	0 – 62	0 – 62	0 – 62
AQW		0 – 30	0 – 30	0 – 62	0 – 62	0 – 62
KW (константа)		KW (константа)				
Доступ к двойному слову	ID	0 – 12	0 – 12	0 – 12	0 – 12	
	QD	0 – 12	0 – 12	0 – 12	0 – 12	
	VD	0 – 2044	0 – 2044	0 – 8188	0 – 10236	0 – 10236
	MD	0 – 28	0 – 28	0 – 28	0 – 28	0 – 28
	SMD	0 – 162	0 – 296	0 – 546	0 – 546	0 – 546
	SD	0 – 28	0 – 28	0 – 28	0 – 28	0 – 28
	LD	0 – 60	0 – 60	0 – 60	0 – 60	0 – 60
	AC	0 – 3	0 – 3	0 – 3	0 – 3	0 – 3
	HC	0 – 5	0 – 5	0 – 5	0 – 5	0 – 5
	KD (константа)					

## Битовые логические операции

### Контакты

#### Стандартные контакты

Команды "Нормально открытый контакт" (LD, A и O) и "Нормально замкнутый контакт" (LDN, AN, ON) получают исходное значение из памяти или из регистра образа процесса. Стандартные контакты получают исходное значение из памяти (или из регистра образа процесса, если типом данных является I или Q).

Нормально открытый контакт замкнут (включен), когда бит равен 1, а нормально замкнутый контакт замкнут (включен), когда бит равен 0. В FBD к блокам И и ИЛИ может быть подключено не более 32 входов. В STL команды, представляющие нормально открытый контакт, загружают значение адресного бита в вершину стека или выполняют логическое сопряжение значения адресного бита со значением в вершине стека в соответствии с таблицей истинности логического И или ИЛИ, а команды, представляющие нормально замкнутый контакт, загружают логическое отрицание значения адресного бита в вершину стека или выполняют логическое сопряжение логического отрицания значения адресного бита со значением в вершине стека в соответствии с таблицей истинности логического И или ИЛИ.

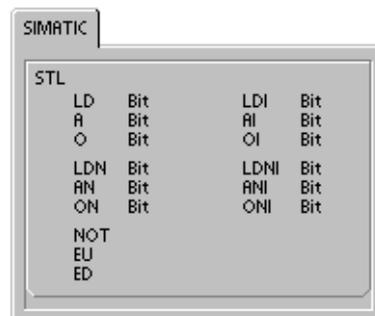
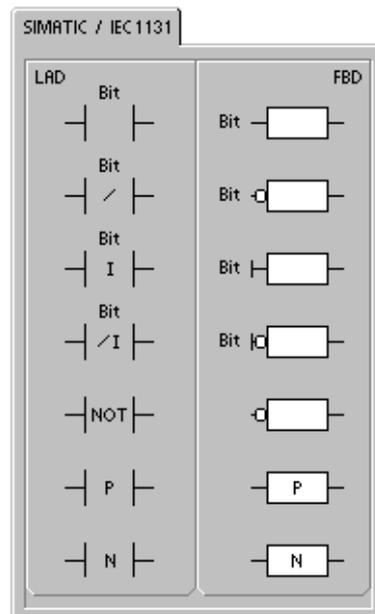
#### Непосредственно управляемые контакты

Непосредственно управляемый контакт при своей актуализации не зависит от цикла S7-200, его значение обновляется немедленно. Команды "Непосредственно управляемый нормально открытый контакт" (LDI, AI и OI) и "Непосредственно управляемый нормально замкнутый контакт" (LDNI, ANI и ONI) при выполнении команды получают значение физического входа, однако, регистр образа процесса не обновляется.

Непосредственно управляемый нормально открытый контакт замкнут (включен), когда физический вход (бит) находится в состоянии 1, а непосредственно управляемый нормально замкнутый контакт замкнут (включен), когда физический вход (бит) находится в состоянии 0. Команды, представляющие непосредственно управляемый нормально открытый контакт, непосредственно загружают значение физического входа в вершину стека или выполняют логическое сопряжение значения физического входа со значением в вершине стека в соответствии с таблицей истинности логического И или ИЛИ, а команды, представляющие непосредственно управляемый нормально замкнутый контакт, непосредственно загружают логическое отрицание значения физического входа в вершину стека или выполняют логическое сопряжение отрицания значения физического входа со значением в вершине стека в соответствии с таблицей истинности логического И или ИЛИ.

#### Команда NOT [NE]

Команда отрицания (NOT) изменяет состояние входа потока сигнала (т.е. она изменяет значение в вершине стека с 0 на 1 или с 1 на 0).



### Положительный и отрицательный фронт

Контакт "Положительный фронт" (EU) пропускает поток сигнала в течение одного цикла при каждом появлении положительного фронта. Контакт "Отрицательный фронт" (ED) пропускает поток сигнала в течение одного цикла при каждом появлении отрицательного фронта. У команды "Положительный фронт" при обнаружении перехода значения в вершине стека с 0 на 1 значение в вершине стека устанавливается в 1; в противном случае оно устанавливается в 0. У команды "Отрицательный фронт" при обнаружении перехода значения в вершине стека с 1 на 0 значение в вершине стека устанавливается в 1; в противном случае оно устанавливается в 0.

При редактировании программы в режиме RUN необходимо ввести параметр для команд обнаружения положительного и отрицательного фронта. Подробную информацию о редактировании в режиме RUN вы найдете в главе 5.

Таблица 6-3. Допустимые операнды для битовых логических операций

Входы/выходы	Тип данных	Операнды
Бит	BOOL	I, Q, V, M, SM, S, T, C, L, поток сигнала
Бит (непосредственно управляемый)	BOOL	I

Как показано на рис. 6-2, S7-200 использует логический стек для решения задач логики управления. В этих примерах «iv0» – «iv7» обозначают начальные значения логического стека, «nv» обозначает новое значение, поставяемое командой, а «S0» обозначает рассчитанное значение, сохраняемое в логическом стеке.



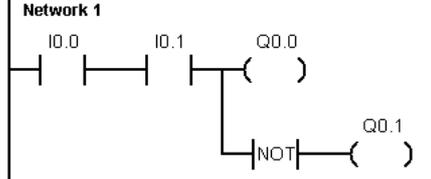
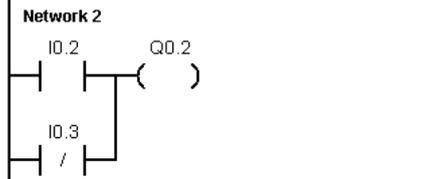
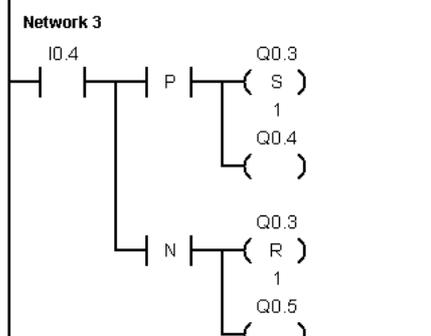
<sup>1</sup> S0 обозначает рассчитанное значение, сохраняемое в логическом стеке.

<sup>2</sup> После выполнения операции загрузки значение iv8 теряется.

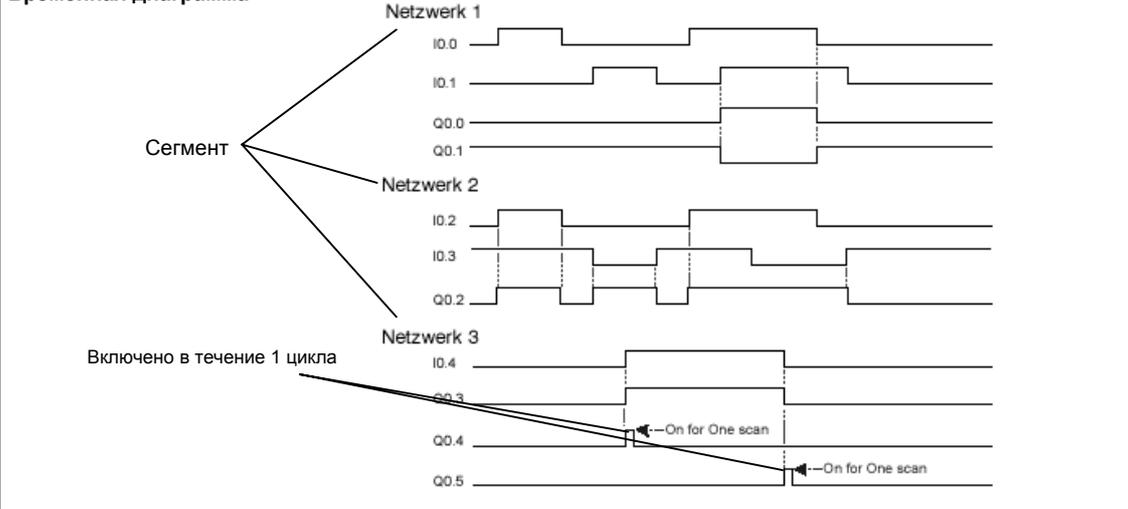
Рис. 6-2. Принцип действия контактов.

	<p><b>Совет</b></p> <p>Так как команды обнаружения положительного и отрицательного фронта требуют переключения, соответственно, с 0 на 1 или с 1 на 0, то нет возможности обнаружить нарастающий или падающий фронт в первом цикле. В первом цикле состояние указанного в этих командах бита устанавливает S7-200. В следующих циклах эти команды сами могут распознать фронты для указанного бита.</p>
---	---

**Пример: Контакты**

<p><b>Network 1</b></p>  <p><b>Network 2</b></p>  <p><b>Network 3</b></p> 	<p>Network 1 //Нормально открытые контакты I0.0 И I0.1 должны //быть замкнуты, чтобы активизировать Q0.0. //Команда NOT действует как инвертор. В режиме //RUN Q0.0 и Q0.1 имеют противоположные //логические состояния.</p> <pre>LD I0.0 A I0.1 = Q0.0 NOT = Q0.1</pre> <p>Network 2 //Чтобы активизировать Q0.2, должен быть включен //нормально открытый контакт I0.2 или выключен //нормально замкнутый контакт I0.3. Чтобы //активизировать выход, одна или более //параллельных ветвей цепной логической схемы //(LAD) (входов ИЛИ) должна принимать значение //"Истина".</p> <pre>LD I0.2 ON I0.3 = Q0.2</pre> <p>Network 3 //Положительный фронт на контакте P или //отрицательный фронт на контакте N приводят к //появлению импульса длительностью в 1 цикл. В // режиме RUN изменения импульсного состояния //Q0.4 и Q0.5 происходят слишком быстро, чтобы их //можно было увидеть в режиме просмотра статуса //программы. Выходы установки (S) и сброса (R) //сохраняют импульс в Q0.3 и делают изменение //состояния видимым в режиме просмотра статуса //программы.</p> <pre>LD I0.4 LPS EU S Q0.3, 1 = Q0.4 LPP ED R Q0.3, 1 = Q0.5</pre>
--	--

**Временная диаграмма**



## Катушки

### Присваивание

Команда присваивания (=) записывает новое значение для выходного бита в регистр образа процесса. При выполнении команды присваивания S7-200 устанавливает или сбрасывает выходной бит в регистре образа процесса. В LAD и FBD указанный бит устанавливается равным потоку сигнала. В STL значение, находящееся в вершине стека, копируется в указанный бит.

### Непосредственное присваивание битового значения

Команда непосредственного присваивания битового значения (=I) при своем выполнении записывает новое значение как в физический выход, так и в образ процесса.

Когда выполняется команда непосредственного присваивания битового значения, физический выход (бит) немедленно устанавливается в соответствии с состоянием потока сигнала. В STL команда непосредственного присваивания битового значения непосредственно копирует значение, находящееся в вершине стека, в указанный физический выход. Символ "I" означает непосредственный доступ; при исполнении команды новое значение записывается в физический выход и в соответствующую ячейку регистра образа процесса. Здесь имеется отличие от других видов доступа, которые записывают новое значение только в регистр образа процесса.

### Установка и сброс

Команды установки (S) и сброса (R) устанавливают (включают) или сбрасывают (выключают) указанное количество входов или выходов (N), начиная с указанного адреса (бита). Вы можете установить или сбросить от 1 до 255 входов и выходов.

Если команда сброса указывает на бит таймера (T) или счетчика (C), то команда сбрасывает бит таймера или счетчика и стирает текущее значение таймера или счетчика.

#### Сбойные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)
- 0091 (операнд вышел за пределы допустимого диапазона)

### Непосредственная установка и непосредственный сброс

Команды непосредственной установки и непосредственного сброса непосредственно устанавливают (включают) или непосредственно сбрасывают (выключают) указанное количество входов или выходов (N), начиная с указанного адреса (бита). Вы можете непосредственно и немедленно установить или сбросить от 1 до 128 входов и выходов. Символ "I" означает непосредственный доступ; при исполнении команды новое значение записывается в физический выход и в соответствующую ячейку регистра образа процесса. Здесь имеется отличие от других видов доступа, которые записывают новое значение только в регистр образа процесса.

#### Сбойные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)
- 0091 (операнд вышел за пределы допустимого диапазона)

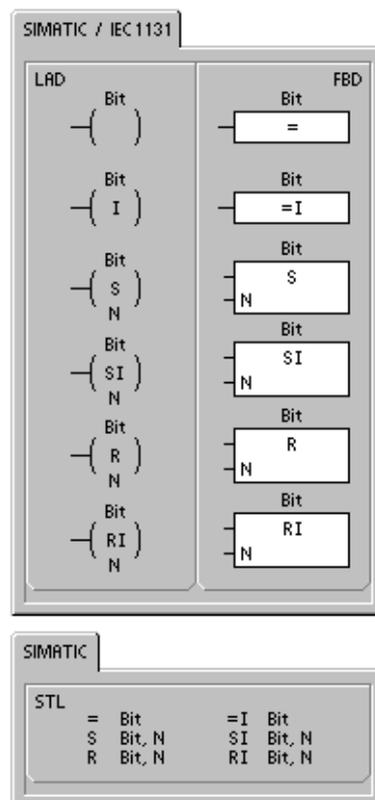
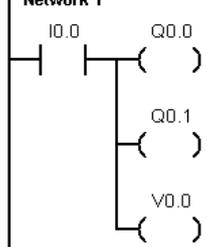
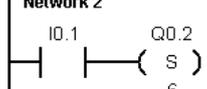
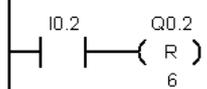
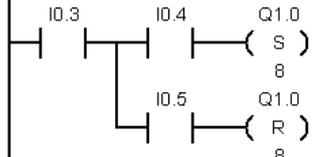
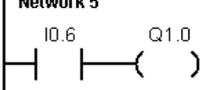


Таблица 6-4. Допустимые операнды для битовых логических операций

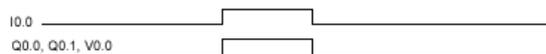
Входы/выходы	Тип данных	Операнды
Бит	BOOL	I, Q, V, M, SM, S, T, C, L
Бит (непосредственно управляемый)	BOOL	Q
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа

**Пример: Катушки**

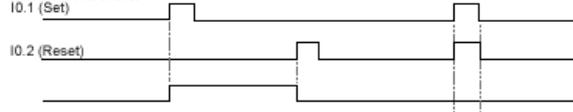
<p><b>Network 1</b></p>  <p><b>Network 2</b></p>  <p><b>Network 3</b></p>  <p><b>Network 4</b></p>  <p><b>Network 5</b></p> 	<p>Network 1 //Команды присваивания присваивают битовые значения //внешним входам и выхода (I, Q) и внутренней памяти //(M, SM, T, C, V, S, L).</p> <pre>LD I0.0 = Q0.0 = Q0.1 = V0.0</pre> <p>Network 2 //Установка в 1 группы из 6 последовательных битов. //Указание адреса начального бита и количества битов, //подлежащих установке. //Индикатор состояния программы для команды установки //(S) включен, когда значение первого бита (Q0.2) равно 1.</p> <pre>LD I0.1 S Q0.2, 6</pre> <p>Network 3 //Сброс в 0 группы из 6 последовательных битов. //Указание адреса начального бита и количества битов, //подлежащих сбросу. //Индикатор состояния программы для команды сброса (R) //включен, когда значение первого бита (Q0.2) равно 0.</p> <pre>LD I0.2 R Q0.2, 6</pre> <p>Network 4 //Установка и сброс группы из 8 выходных битов //(Q1.0 – Q1.7).</p> <pre>LD I0.3 LPS A I0.4 S Q1.0, 8 LPP A I0.5 R Q1.0, 8</pre> <p>Network 5 //Команды установки и сброса выполняют функцию реле с //блокировкой. //Обеспечьте, чтобы биты установки и сброса не были //переписаны другой командой присваивания. В этом //примере сегмент (Network) 4 устанавливает и сбрасывает //8 выходных битов (Q1.0 – Q1.7) группой. //В режиме RUN сегмент 5 может заменять значение бита //Q1.0 и управлять индикаторами состояния программы для //установки и сброса в сегменте 4.</p> <pre>LD I0.6 = Q1.0</pre>
--	---

**Временная диаграмма**

**Сегмент 1**



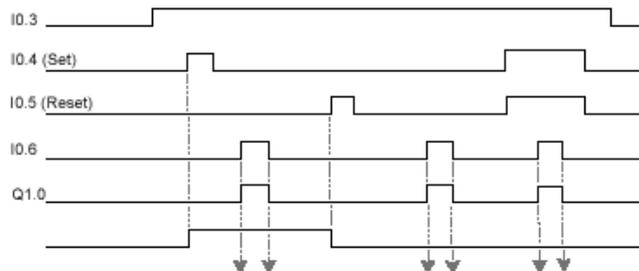
**Сегменты 2 и 3**



Set – установка  
Reset - Сброс

Сброс в 0 перезаписывает установку в 1, так как программа выполняет сегмент 3, выполняющий сброс, после сегмента 2, выполняющего установку.

**Сегменты 4 и 5**

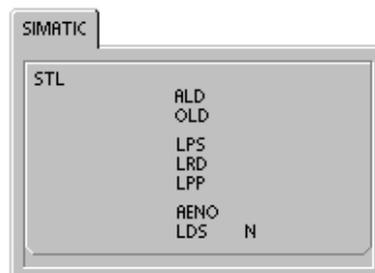


Команда присваивания (=) в сегменте 5 перезаписывает первый бит (Q1.0), устанавливаемый и сбрасываемый в сегменте 4, так как программа выполняет сегмент 5 последним.

## Стековые операции

### Логическое сопряжение 1-го и 2-го уровня по И

Команда логического сопряжения первого и второго уровня стека по И (ALD) логически сопрягает значения в первом и втором уровне стека, используя логическую операцию И. Результат загружается в вершину стека. После выполнения ALD глубина стека уменьшается на единицу.



### Логическое сопряжение 1-го и 2-го уровня по ИЛИ

Команда логического сопряжения первого и второго уровня стека по ИЛИ (OLD) логически сопрягает значения в первом и втором уровне стека, используя логическую операцию ИЛИ. Результат загружается в вершину стека. После выполнения OLD глубина стека уменьшается на единицу.

### Дублирование вершины логического стека

Команда дублирования вершины логического стека (LPS) дублирует значение вершины стека и помещает это значение в стек. Дно стека выталкивается и теряется.

### Копирование второго уровня стека

Команда копирования второго уровня стека (LRD) копирует второй уровень стека в его вершину. В стек ничего не помещается и из него ничего не извлекается, но его вершина замещается копией.

### Извлечение вершины стека

Команда извлечения вершины стека (LPP) извлекает одно значение из стека. Второй уровень становится новой вершиной стека.

### Логическое И для бита ENO

Команда логического И для бита ENO (AENO) выполняет логическое сопряжение бита ENO и вершины стека с использованием операции И, производя такой же эффект, как и бит ENO в блоке LAD или FBD. Результатом операции И является новая вершина стека.

ENO – это булев выход для блоков в LAD и FBD. Если блок имеет поток сигнала на входе EN и выполняется без ошибок, то выход ENO передает поток сигнала следующему элементу. ENO может быть использован как деблокирующий бит, указывающий на успешное завершение команды. Бит ENO вместе с вершиной стека используется для воздействия на поток сигнала для выполнения последующих команд. У команд STL нет входа EN. Для выполнения условных операций вершина стека должна быть равна 1. В STL нет выхода ENO, но команды STL, соответствующие командам LAD и FBD с выходами ENO обязательно устанавливают специальный бит ENO. Доступ к этому биту осуществляется с помощью команды AENO.

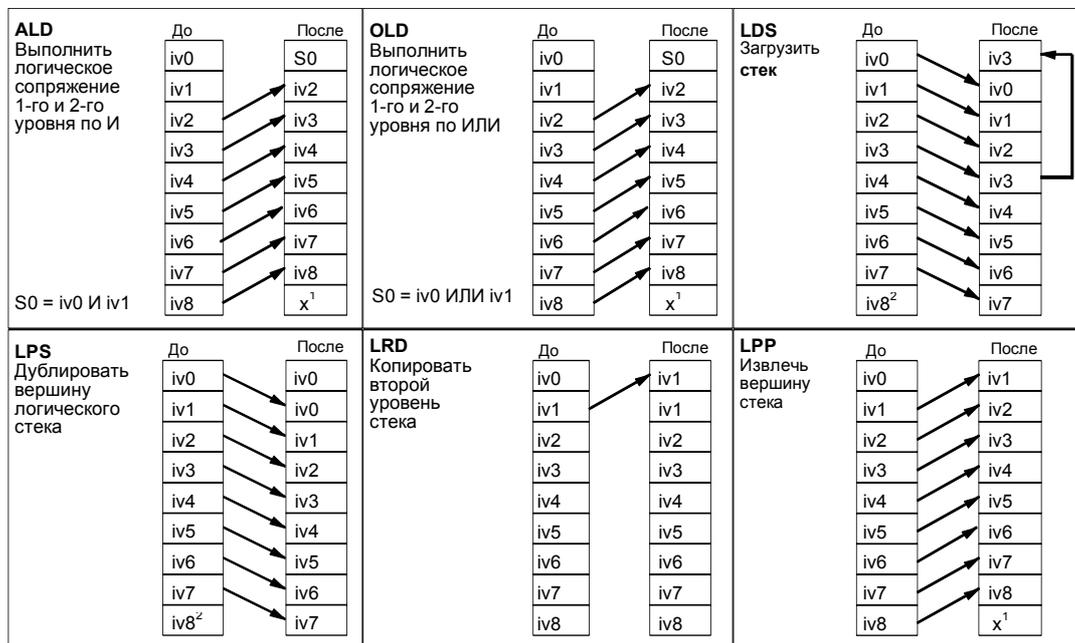
### Загрузка стека

Команда загрузки стека (LDS) дублирует бит стека (N) и помещает это значение в вершину стека. Дно стека выталкивается и теряется.

Таблица 6–5. Допустимые операнды для команды загрузки стека

Входы/выходы	Тип данных	Операнды
N	BYTE	Константа (от 0 до 8)

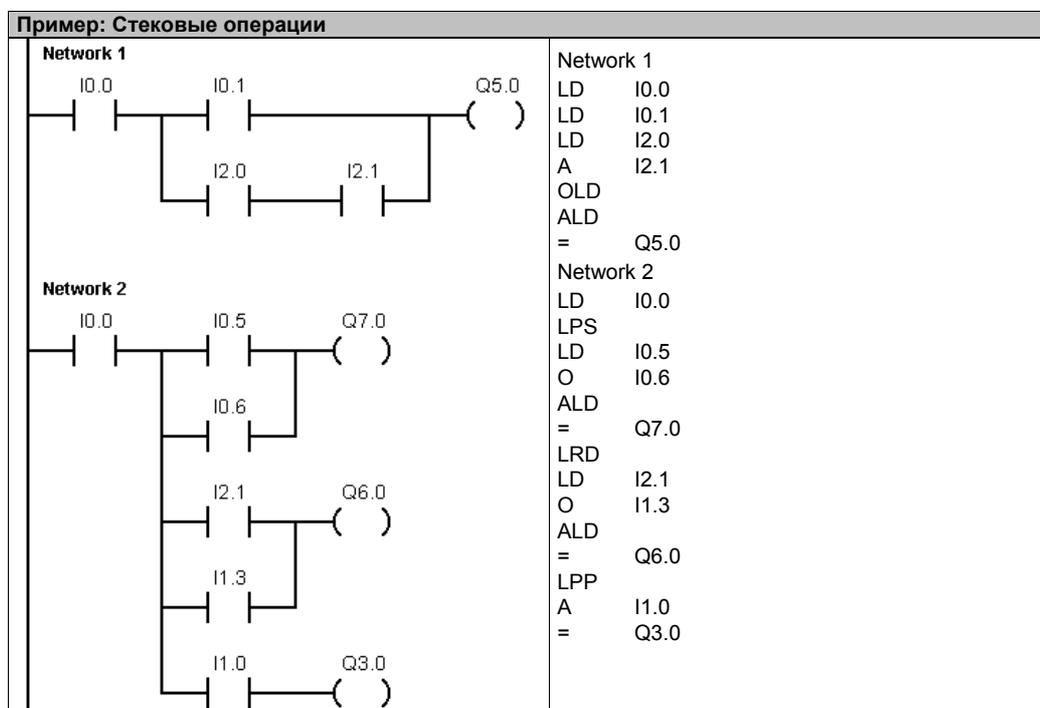
Как показано на рис. 6–3, S7–200 использует логический стек для решения задач логики управления. В этих примерах «iv0» «iv7» обозначают начальные значения логического стека, «nv» обозначает новое значение, поставляемое командой, а «S0» обозначает рассчитанное значение, сохраняемое в логическом стеке.



<sup>1</sup>Значение неизвестно (оно может быть равно 0 или

<sup>2</sup>После выполнения команды "Дублировать вершину логического стека" или "Загрузить стек" значение iv8 теряется.

Рис. 6–3. Принцип действия стековых операций



### Функциональный блок с двумя устойчивыми состояниями: преимущество установки и преимущество сброса

Функциональный блок с двумя устойчивыми состояниями и преимуществом установки представляет собой триггер, у которого доминирует установка. Если сигнал установки (S1) и сигнал сброса (R) одновременно принимают значение истина, то выход (OUT) принимает значение истина.

Функциональный блок с двумя устойчивыми состояниями и преимуществом сброса представляет собой триггер, у которого доминирует сброс. Если сигнал установки (S) и сигнал сброса (R1) одновременно принимают значение истина, то выход (OUT) принимает значение ложь.

Параметр Bit представляет собой булев параметр, который устанавливается или сбрасывается. Дополнительный выход отражает сигнальное состояние параметра Bit.

В таблице 6–7 представлены состояния функциональных блоков для программы-примера.

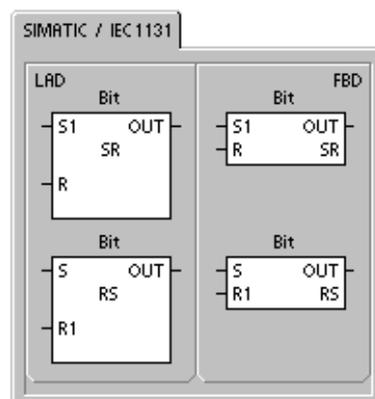


Таблица 6–6. Допустимые операнды для функционального блока с двумя устойчивыми состояниями

Входы/выходы	Типы данных	Операнды
S1, R	BOOL	I, Q, V, M, SM, S, T, C, поток сигнала
S, R1, OUT	BOOL	I, Q, V, M, SM, S, T, C, L, поток сигнала
Bit	BOOL	I, Q, V, M, S

Пример: Функциональный блок с двумя устойчивыми состояниями: преимущество установки и преимущество сброса

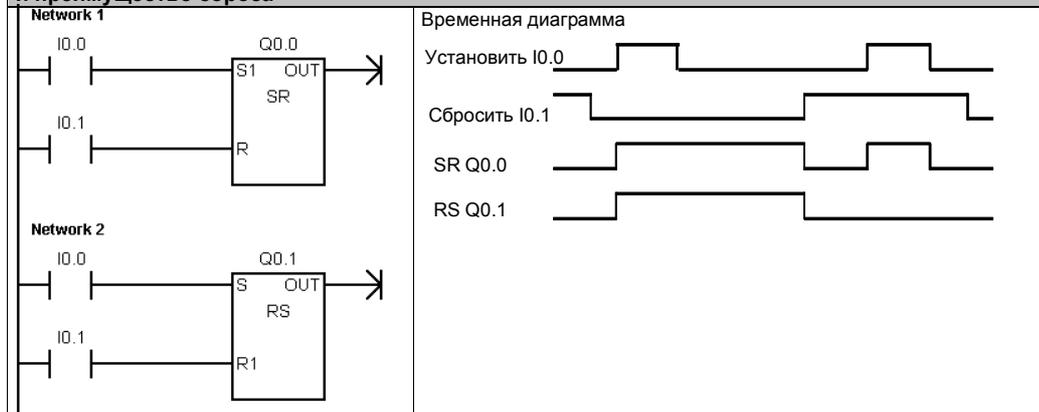


Таблица 6–7. Таблица состояний для функциональных блоков с двумя устойчивыми состояниями: преимущество установки и преимущество сброса

Команда	S1	R	Out (Bit)
Функциональный блок с двумя устойчивыми состояниями и преимуществом установки (SR)	0	0	Предыдущее состояние
	0	1	0
	1	0	1
	1	1	1
Команда	S	R1	Out (Bit)
Функциональный блок с двумя устойчивыми состояниями и преимуществом сброса (RS)	0	0	Предыдущее состояние
	0	1	0
	1	0	1
	1	1	0

## Команды, связанные со временем

### Чтение и установка часов реального времени

Команда чтения часов реального времени (TODR) считывает текущее время и дату из аппаратных часов и загружает их в 8-байтовый буфер времени, начиная с адреса T. Команда установки часов реального времени (TODW) записывает в аппаратные часы текущее время и дату, загруженные в 8-байтовый буфер, начинающийся с адреса, указанного параметром T.

Все значения даты и времени необходимо записывать в формате BCD (например, 16#97 для 1997 года). На рис. 6–4 показан формат 8-байтового буфера времени (T).

После продолжительного отключения питания или потери памяти часы реального времени (TOD) инициализируют следующую дату и время:

Дата: 01-Янв-90  
 Время: 00:00:00  
 День недели: Воскресенье

#### Сбойные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)
- 0007 (ошибка даты TOD) *Только для установки часов реального времени*
- 000С (часы отсутствуют)

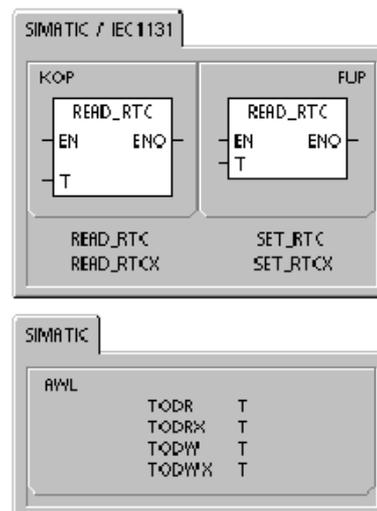
Таблица 6–8. Допустимые операнды для команд, связанных со временем

Входы/выходы	Типы данных	Операнды
T	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC

T	T+1	T+2	T+3	T+4	T+5	T+6	T+7
Год: 00 – 99	Месяц: 01 – 12	День: 01 – 31	Часы: 00 – 23	Минуты: 00 – 59	Секунды: 00 – 59	0	День недели: 0 – 7*

\*T+7 1=воскресенье, 7=суббота  
 0 деактивирует день недели.

Рис. 6–4. Формат 8-байтового буфера времени (T)



**Совет**

CPU S7-200 не проверяет соответствие дня недели дате. Могут быть восприняты неверные даты, например, 30 февраля. Правильность даты должны обеспечивать вы сами.

Не используйте команду TODR/TODW одновременно в главной программе и программе обработки прерывания. Эта команда не будет исполнена в программе обработки прерывания, которая пытается ее выполнить, когда действует другая команда TODR/TODW. Если делается попытка одновременно двух обращений к часам, то устанавливается SM4.3 (нефатальная ошибка 0007).

Часы реального времени в S7-200 используют только две младшие значащие цифры для года, так что 2000-й год представлен как 00. ПЛК S7-200 никак не использует информацию о годе. Однако пользовательские программы, которые используют арифметические операции или сравнения со значением года, должны учитывать двухзначное представление и изменение столетия.

Високосные годы обрабатываются правильно до 2096 года.

### Чтение расширенных часов реального времени

Команда чтения расширенных часов реального времени (TODRX) считывает текущую конфигурацию времени, даты и летнего времени из ПЛК и загружает их в 19-байтовый буфер, начинающийся по адресу, указанному параметром T.

#### Сбойные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)
- 000C (отсутствует модуль часов)
- 0091 (ошибка диапазона)

### Установка расширенных часов реального времени

Команда установки расширенных часов реального времени (TODWX) записывает текущую конфигурацию времени, даты и летнего времени в ПЛК из 19-байтового буфера, находящегося по адресу, указанному в параметре T.

Все значения даты и времени необходимо записывать в формате BCD (например, 16#02 для 2002 года). В таблице 6-9 показан формат 19-байтового буфера времени (T).

После продолжительного отключения питания или потери памяти часы реального времени инициализируют следующую дату и время:

#### Сбойные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)
- 0007 (ошибка данных TOD)
- 000C (отсутствует модуль часов)
- 0091 (ошибка диапазона)

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

Таблица 6–9. Формат 19–байтового буфера времени (TI)

Т-байт	Описание	Данные, содержащиеся в байте
0	год (0-99)	текущий год (в формате BCD)
1	месяц (1-12)	текущий месяц (в формате BCD)
2	день (1-31)	текущий день (в формате BCD)
3	час (0-23)	текущий час (в формате BCD)
4	минута (0-59)	текущая минута (в формате BCD)
5	секунда (0-59)	текущая секунда (в формате BCD)
6	00	резерв – всегда устанавливается в 00
7	день недели (1-7)	текущий день недели, 1=воскресенье (в формате BCD)
8	режим (00H-03H, 08H, 10H-13H, FFH)	Режим корректировки: 00H = корректировка заблокирована 01H = Европа (смещение часового пояса от UTC = 0 часов) <sup>1</sup> 02H = Европа (смещение часового пояса от UTC = +1 час) <sup>1</sup> 03H = Европа (смещение часового пояса от UTC = +2 часа) <sup>1</sup> 04H-07H = резерв 08H = Европа (смещение часового пояса от UTC = -1 час) <sup>1</sup> 09H-0FH = резерв 10H = США <sup>2</sup> 11H = Австралия <sup>3</sup> 12H = Австралия (Тасмания) <sup>4</sup> 13H = Новая Зеландия <sup>5</sup> 14H-FEH = резерв FFH = определяется пользователем (с использованием значений в байтах 9-18)
9	часы корректировки (0-23)	размер корректировки, часы (в формате BCD)
10	минуты корректировки (0-59)	размер корректировки, минуты (в формате BCD)
11	первый месяц (1-12)	первый месяц летнего времени (в формате BCD)
12	первый день (1-31)	первый день летнего времени (в формате BCD)
13	первый час (0-23)	первый час летнего времени (в формате BCD)
14	первая минута (0-59)	первая минута летнего времени (в формате BCD)
15	последний месяц (1-12)	последний месяц летнего времени (в формате BCD)
16	последний день (1-31)	последний день летнего времени (в формате BCD)
17	последний час (0-23)	последний час летнего времени (в формате BCD)
18	последняя минута (0-59)	последняя минута летнего времени (в формате BCD)

<sup>1</sup> Европейская конвенция: Переводить часы на один час вперед в последнее воскресенье марта в 1:00 час ночи UTC (Universal Time Coordinated = универсальное синхронизированное время).

Переводить часы на один час назад в последнее воскресенье октября в 2:00 часа ночи UTC. (Местное время выполнения корректировки зависит от смещения часового пояса от UTC).

<sup>2</sup> Конвенция США: Переводить часы на один час вперед в первое воскресенье апреля в 2:00 часа ночи местного времени. Переводить часы на один час назад в последнее воскресенье октября в 2:00 часа ночи местного времени.

<sup>3</sup> Австралийская конвенция: Переводить часы на один час вперед в последнее воскресенье октября в 2:00 часа ночи местного времени. Переводить часы на один час назад в последнее воскресенье марта в 3:00 часа ночи местного времени.

<sup>4</sup> Австралийская конвенция (Тасмания): Переводить часы на один час вперед в первое воскресенье октября в 2:00 часа ночи местного времени. Переводить часы на один час назад в последнее воскресенье марта в 3:00 часа ночи местного времени

<sup>5</sup> Новозеландская конвенция: Переводить часы на один час вперед в первое воскресенье октября в 2:00 часа ночи местного времени. Переводить часы на один час назад 15 марта, если этот день приходится на воскресенье, или в первое воскресенье после 15 марта в 3:00 часа ночи местного времени

## Команды для обмена данными

### Команды чтения из сети и записи через сеть

Команда чтения из сети (NETR) инициирует коммуникационную операцию для получения данных из удаленного устройства через указанный порт (PORT), как указано в таблице (TBL). Команда записи через сеть (NETW) инициирует коммуникационную операцию для записи данных в удаленное устройство через указанный порт (PORT), как указано в таблице (TBL).

**Сбойные состояния, устанавливающие ENO = 0 :**

- 0006 (косвенный адрес)
- Функция возвращает ошибку и устанавливает бит E бита состояния таблицы (см. рис. 6-5)

Команда NETR может прочитать до 16 байтов информации из удаленной станции, а команда NETW может записать до 16 байтов информации в удаленную станцию.

В программе можно иметь любое количество команд NETR/NETW, но одновременно можно активизировать не более восьми команд NETR и NETW. Например, в данном S7-200 в одно и то же время у вас могут активными четыре NETR и четыре NETW или два NETR и шесть NETW.

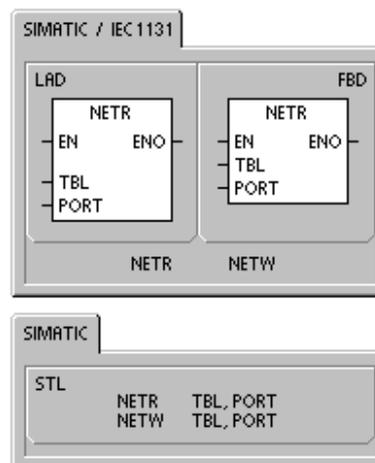
С помощью мастера команд для операций чтения из сети и записи через сеть вы можете создать счетчик. Для запуска этого мастера команд выберите команду меню **Tools > Instruction Wizard [Инструменты > Мастер команд]**, а затем в окне мастера команд выберите опцию Network Read/Network Write [Чтение из сети/Запись через сеть].



Мастер команд

Таблица 6-10. Допустимые операнды для команд чтения из сети и записи через сеть

Входы/выходы	Тип данных	Операнды
TBL	BYTE	VB, MB, *VD, *LD, *AC
PORT	BYTE	константа      для CPU 221, CPU 222, CPU 224:      0 для CPU 224XP, CPU 226:      0 или 1



На рис. 6–5 описана таблица, к которой обращается параметр TBL, а в таблице 6–11 приведены коды ошибок.

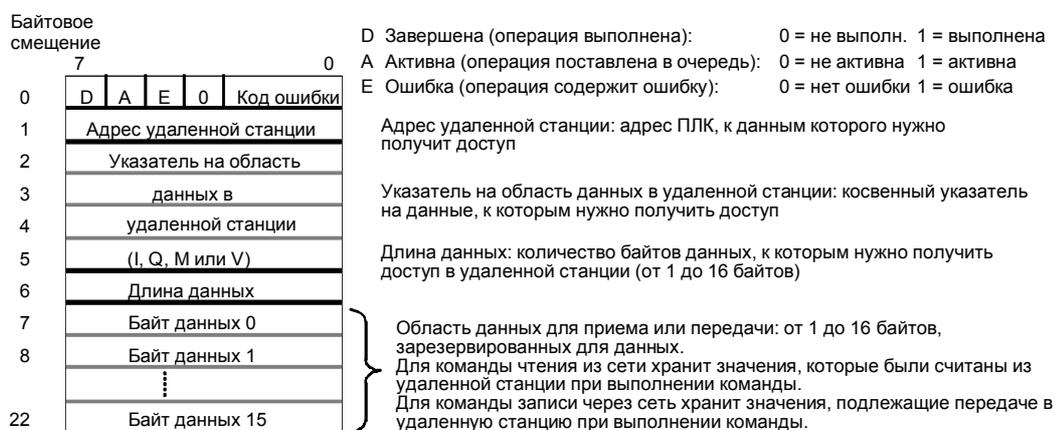


Рис. 6–5. Параметр TBL для команд чтения из сети и записи через сеть

Таблица 6–11. Коды ошибок для параметра TBL

Код	Определение
0	Нет ошибки.
1	Ошибка истечения времени ожидания: удаленная станция не отвечает.
2	Ошибка приема: ошибка четности, кадрирования или контрольной суммы в ответе.
3	Ошибка с переходом в автономный режим: конфликт, вызванный двойным адресом станции или неисправностью аппаратуры.
4	Ошибка переполнения очереди: активизировано более 8 команд NETR/NETW.
5	Нарушение протокола: попытка выполнить команду NETR или NETW, без разблокировки привилегированного режима PPI в SMB30 или SMB130.
6	Недопустимый параметр: параметр TBL содержит недопустимое или ошибочное значение.
7	Нет ресурса: удаленная станция занята (Идет процесс загрузки в станцию или из станции.)
8	Ошибка 7-го уровня: нарушение протокола приложения
9	Ошибка сообщения: неверный адрес данных или неправильная длина данных
A – F	Не используются. (Зарезервированы)

На рис. 6–6 показан пример, иллюстрирующий использование команд NETR и NETW. В этом примере речь идет о производственной линии, где стаканчики заполняются маслом и передаются одной из четырех упаковочных машин. Упаковочная машина пакует по восемь стаканчиков с маслом в одну картонную коробку. Направляющее устройство управляет потоком стаканчиков с маслом, направляя их к той или иной упаковочной машине. Для управления упаковочными машинами используются четыре модуля S7–200, а для управления направляющим устройством используется один S7–200, оборудованный интерфейсом оператора TD 200.

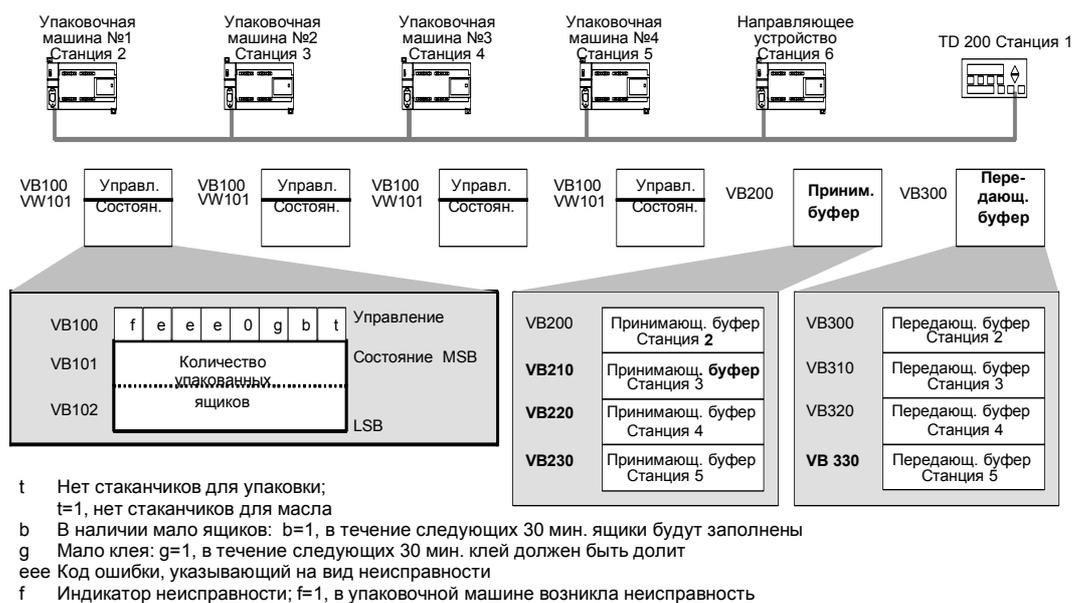
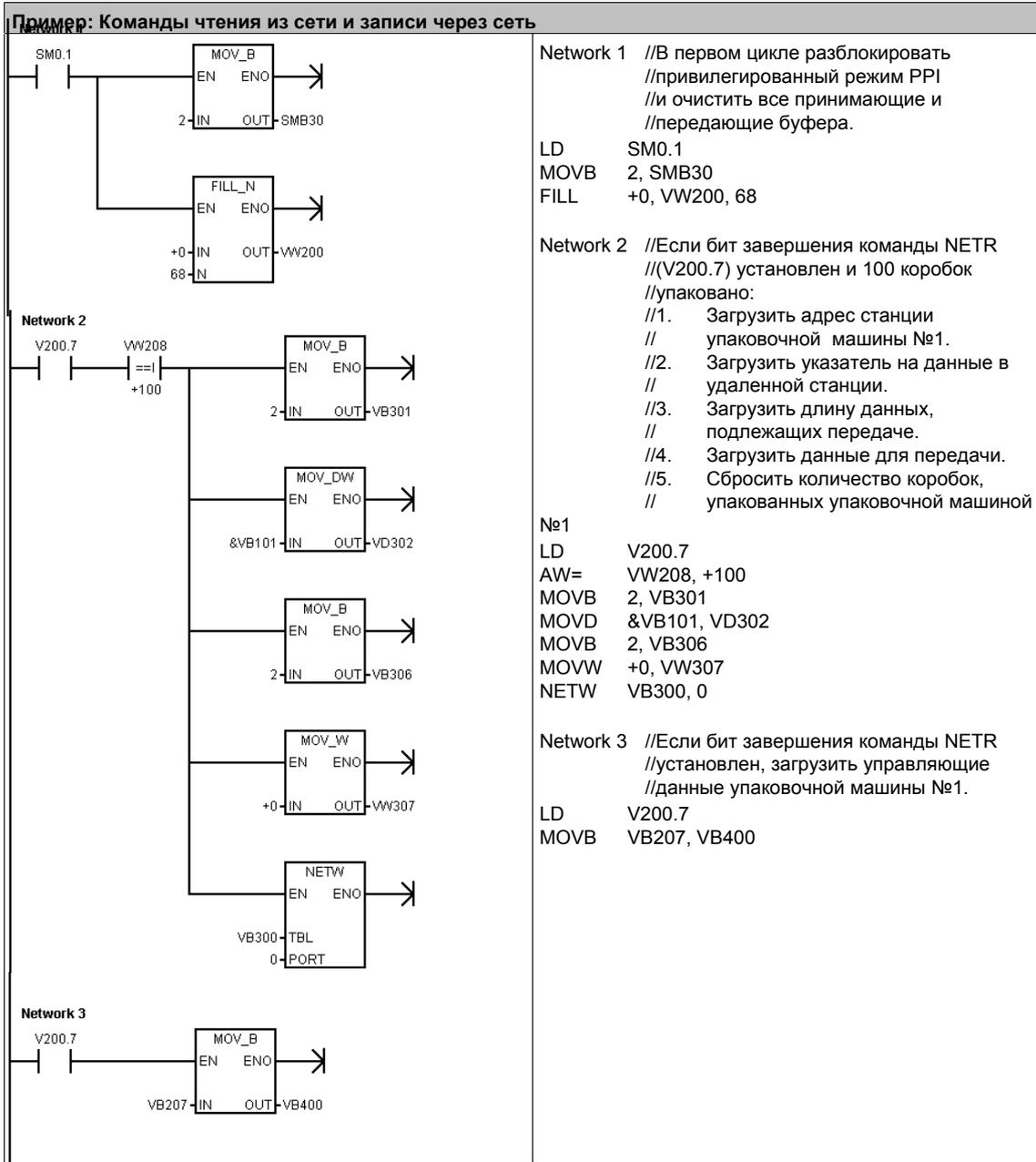


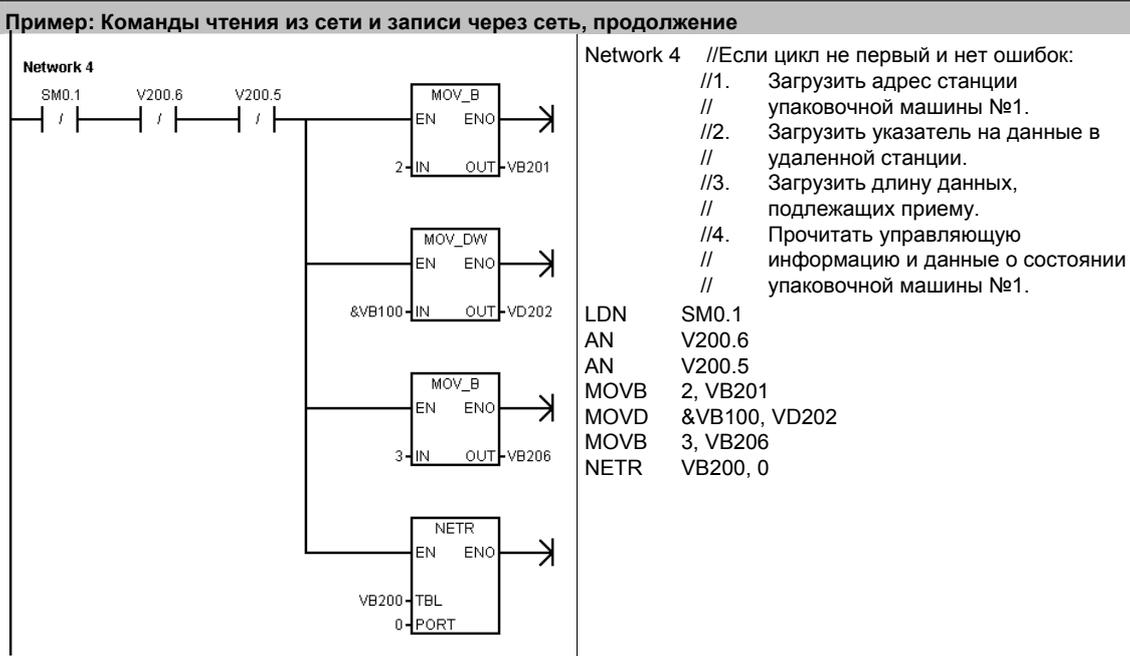
Рис. 6–6. Пример команд чтения из сети и записи через сеть

На рис. 6–7 показан принимающий буфер (VB200) и передающий буфер (VB300) для доступа к данным в станции 2. S7-200 использует команду NETR для регулярного чтения управляющей информации и информации о состоянии из каждой упаковочной машины. Каждый раз, когда упаковочная машина упаковывает 100 коробок, направляющее устройство замечает это и с помощью команды NETW передает сообщение для сброса слова состояния.

Принимающий буфер для чтения из упаковочной машины №1		Передающий буфер для сброса счетчика упаковочной машины №1									
	7	0									
VB200	D	A	E	0	Код ошибки	VB300	D	A	E	0	Код ошибки
VB201	Адрес удаленной станции = 2					VB301	Адрес удаленной станции = 2				
VB202	Указатель на					VB302	Указатель на				
VB203	область данных					VB303	область данных				
VB204	в					VB304	в				
VB205	удаленной станции = (&VB100)					VB305	удаленной станции = (&VB101)				
VB206	Длина данных = 3 байта					VB306	Длина данных = 2 байта				
VB207	<b>Control</b>					VB307	0				
VB208	Состояние (старший байт)					VB308	0				
VB209	Состояние (младший байт)										

Рис. 6–7. Образец данных TBL для примера команд чтения из сети и записи через сеть





## Команды передачи и приема (свободно программируемый обмен данными)

Команда передачи (XMT) используется в режиме свободно программируемого обмена данными для передачи данных через коммуникационный порт(ы).

Команда приема (RCV) инициирует или завершает функцию приема сообщения. Чтобы блок приема мог работать, вы должны указать условия начала и конца сообщения. Сообщения, получаемые через указанный порт (PORT), сохраняются в буфере данных (TBL). Первая запись в буфере данных указывает количество принятых байтов.

### Сбойные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)
- 0009 (одновременная передача и прием через порт 0)
- 000B (одновременная передача и прием через порт 1)
- Ошибка параметра приема устанавливает SM86.6 или SM186.6
- S7-200 CPU не находится в режиме свободно программируемого обмена данными

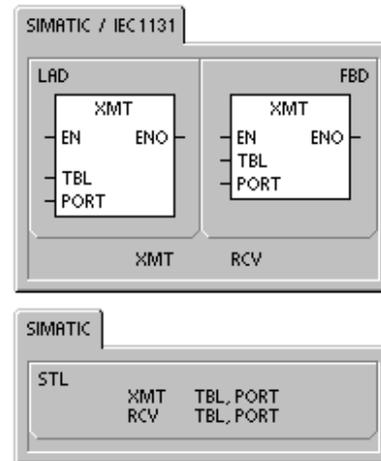


Таблица 6–12. Допустимые операнды для команд передачи и приема

Входы/выходы	Тип данных	Операнды
TBL	BYTE	IB, QB, VB, MB, SMB, SB, *VD, *LD, *AC
PORT	BYTE	константа для CPU 221, CPU 222, CPU 224: 0 для CPU 224XP, CPU 226: 0 или 1

Дополнительную информацию об использовании режима свободно программируемого обмена данными вы найдете в разделе "Создание протоколов, определяемых пользователем, при свободно программируемом обмене данными" на стр. 226 в главе 7.

### Использование режима свободно программируемого обмена данными для управления последовательным коммуникационным портом

Режим свободно программируемого обмена данными можно выбрать для управления последовательным коммуникационным портом S7-200 с помощью программы пользователя. Если вы выбираете режим свободно программируемого обмена данными, то ваша программа управляет работой коммуникационного порта путем использования прерываний приема, прерываний передачи, команды передачи и команды приема. В режиме свободно программируемого обмена данными протокол связи полностью управляется программой, представленной в виде цепной логической схемы (LAD). Для выбора скорости передачи и способа контроля четности используются SMB30 (для порта 0) и SMB130 (для порта 1, если ваш S7-200 имеет два порта).

Режим свободно программируемого обмена данными блокируется, и восстанавливается обычная связь (например, доступ через устройство программирования), когда S7-200 находится в состоянии STOP.

В простейшем случае вы можете послать сообщение на принтер или дисплей, используя только команду передачи (XMT). К другим примерам относятся связь с устройством для считывания штрихового кода, весами или сварочным аппаратом. В каждом случае вы должны написать программу для поддержки протокола, используемого устройством, с которым S7-200 поддерживает связь при нахождении в режиме свободно программируемого обмена данными.

Свободно программируемый обмен данными возможен только тогда, когда S7-200 находится в режиме RUN. Разблокируйте режим свободно программируемого обмена данными установкой значения 01 в поле выбора протокола SMB30 (порт 0) или SMB130 (порт 1). В режиме свободно программируемого обмена данными связь с устройством программирования невозможна.



#### Совет

Режимом свободно программируемого обмена данными можно управлять с помощью бита специальной памяти SM0.7, который отражает текущее положение переключателя режимов работы. Если SM0.7 равен 0, то переключатель находится в положении TERM; если SM0.7 = 1, то переключатель режимов работы находится в положении RUN. Если вы разблокируете режим свободно программируемого обмена данными только тогда, когда переключатель находится в положении RUN, то вы можете использовать устройство программирования для контроля и управления работой S7-200 путем перевода переключателя в любое другое положение.

### Переход от обмена данными через PPI к свободно программируемому обмену данными

SMB30 и SMB130 конфигурируют коммуникационные порты 0 и 1 соответственно для свободно программируемого обмена данными, предоставляя возможность выбора скорости передачи, способа контроля четности и количества битов данных. На рис. 6–8 описан управляющий байт для свободно программируемого обмена данными. Для всех конфигураций генерируется один стоповый бит.

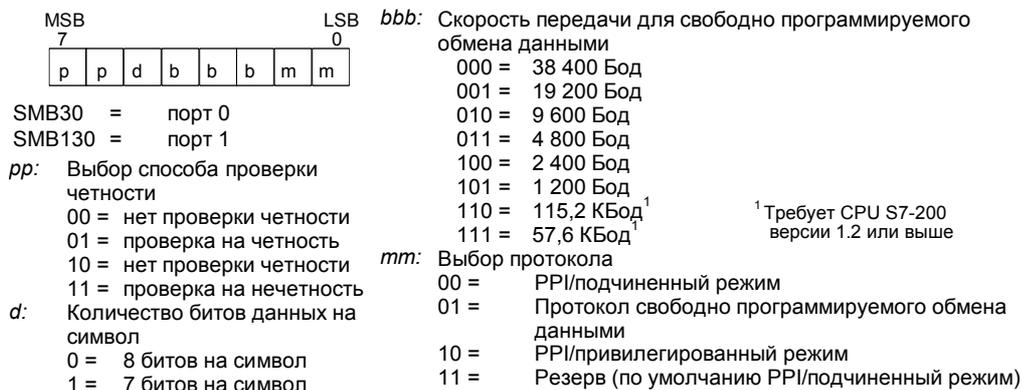


Рис. 6–8. Управляющий байт для свободно программируемого обмена данными (SMB30 или SMB130)

### Передача данных

Команда передачи дает возможность передать буфер, состоящий из одного или нескольких символов (не более 255).

На рис. 6–9 показан формат буфера передачи.

После того как передан последний символ буфера, S7-200 генерирует прерывание (прерывающее событие 9 для порта 0 и прерывающее событие 26 для порта 1), если событию «Передача завершена» поставлена в соответствие программа обработки прерывания.

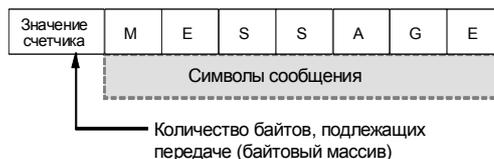


Рис. 6–9. Формат буфера передачи

Вы можете производить передачу и без использования прерываний (например, посылая сообщение на принтер), контролируя SM4.5 или SM4.6, сигнализирующих о завершении передачи.

Команда передачи может быть использована для генерирования условия паузы (BREAK) путем установки количества символов в ноль, а затем исполнения команды передачи. Это генерирует условие паузы в строке для передачи 16 битов с текущей скоростью передачи. Передача паузы обрабатывается так же, как передача любого другого сообщения, причем при завершении передачи паузы генерируется прерывание, а SM4.5 или SM4.6 отображают текущее состояние передачи.

### Прием данных

Команда приема дает возможность принять буфер, состоящий из одного или нескольких символов (не более 255).

На рис. 6–10 показан формат буфера приема.

После того как принят последний символ буфера, генерируется прерывание (прерывающее событие 23 для порта 0 и прерывающее событие 24 для порта 1), если событию «Прием сообщения завершен» поставлена в соответствие программа обработки прерывания.



Рис. 6–10. Формат буфера приема

Вы можете принимать сообщения и без использования прерываний, контролируя SMB86 (порт 0) или SMB186 (порт 1). Этот байт не равен нулю, если приема не активен или завершен. Он равен нулю, когда происходит прием.

Как показано в таблице 6–12, команда приема позволяет выбирать условия начала и окончания сообщения с помощью SMB86 – SMB94 для порта 0 и SMB186 – SMB194 для порта 1.

	<p><b>Совет</b></p> <p>Функция приема сообщений автоматически завершается при превышении количества символов или ошибке четности. Вы должны определить условие начала и условие конца (максимальное количество символов), чтобы функция приема сообщений могла работать.</p>
---	--

Таблица 6–13. Байты буфера приема (от SMB86 до SMB94 и от SM1B86 до SMB194)

Порт 0	Порт 1	Описание										
SMB86	SMB186	<p>Байт состояния приема сообщения</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">MSB 7</td> <td style="text-align: center;">LSB 0</td> </tr> <tr> <td style="text-align: center;">n</td> <td style="text-align: center;">r</td> </tr> <tr> <td style="text-align: center;">e</td> <td style="text-align: center;">0</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">t</td> </tr> <tr> <td style="text-align: center;">c</td> <td style="text-align: center;">p</td> </tr> </table> <p>n: 1 = Функция приема сообщений завершена: пользователь дал команду на деактивизацию.                      r: 1 = Функция приема сообщений завершена: ошибка во входном параметре или отсутствует условие начала или конца                      e: 1 = Принят символ конца                      t: 1 = Функция приема сообщений завершена: истекло время                      c: 1 = Функция приема сообщений завершена: достигнуто максимальное количество символов                      p: 1 = Функция приема сообщений завершена: ошибка четности</p>	MSB 7	LSB 0	n	r	e	0	0	t	c	p
MSB 7	LSB 0											
n	r											
e	0											
0	t											
c	p											
SMB87	SMB187	<p>Байт управления приемом сообщения</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">MSB 7</td> <td style="text-align: center;">LSB 0</td> </tr> <tr> <td style="text-align: center;">en</td> <td style="text-align: center;">sc</td> </tr> <tr> <td style="text-align: center;">ec</td> <td style="text-align: center;">il</td> </tr> <tr> <td style="text-align: center;">c/m</td> <td style="text-align: center;">tmr</td> </tr> <tr> <td style="text-align: center;">bk</td> <td style="text-align: center;">0</td> </tr> </table> <p>en: 0 = функция приема сообщений заблокирована.                      1 = функция приема сообщений разблокирована.                      Бит разблокировки/блокировки приема сообщений проверяется при каждом исполнении команды RCV.                      sc: 0 = игнорировать SMB88 или SMB188.                      1 = использовать значение SMB88 или SMB188 для обнаружения начала сообщения.                      ec: 0 = игнорировать SMB89 или SMB189.                      1 = использовать значение SMB89 или SMB189 для обнаружения конца сообщения.                      il: 0 = игнорировать SMW90 или SMW190.                      1 = использовать значение SMW90 или SMW190 для обнаружения бездействия линии                      c/m: 0 = таймер измеряет время между символами                      1 = таймер измеряет время сообщения.                      tmr: 0 = игнорировать SMW92 или SMW192.                      1 = завершить прием, если превышен интервал времени, указанный в SMW92 или SMW192.                      bk: 0 = игнорировать условия паузы                      1 = использовать условие паузы как начало обнаружения сообщения.</p>	MSB 7	LSB 0	en	sc	ec	il	c/m	tmr	bk	0
MSB 7	LSB 0											
en	sc											
ec	il											
c/m	tmr											
bk	0											
SMB88	SMB188	Символ начала сообщения.										
SMB89	SMB189	Символ конца сообщения.										
SMW90	SMW190	Время бездействия линии в миллисекундах. Первый символ, принятый по истечении времени бездействия линии, является началом нового сообщения.										
SMW92	SMW192	Значение контроля времени при измерении времени между символами и времени сообщения в миллисекундах. Если этот интервал времени истек, то прием сообщения завершается.										
SMB94	SMB194	Максимальное количество символов, которое должно быть принято (от 1 до 255 байтов). Этот диапазон должен быть установлен на ожидаемый максимальный размер буфера, даже если завершение сообщения с помощью подсчета символов не используется.										

### Условия начала и окончания сообщения для команды приема

Для определения условий начала и окончания сообщения команда приема использует биты байта управления приемом сообщений (SMB87 или SMB187).



#### Совет

Если во время выполнения команды приема коммуникационный порт занят другими устройствами, то функция приема сообщений может начать прием символа в середине этого символа, что может привести к ошибке при контроле четности и завершению приема сообщения. Если проверка четности не активизирована, то принятое сообщение может содержать неправильные символы. Эта ситуация может возникнуть, если в качестве начального условия назначен определенный начальный символ или любой символ, как это описано ниже в пунктах 2 и 6.

Команда приема поддерживает несколько условий начала сообщения. Если указанное вами условие начала приема содержит паузу или распознавание бездействия линия, то описанной выше проблемы удастся избежать, принуждая функцию приема сообщений синхронизировать начало сообщения с началом символа, прежде чем символ будет помещен в буфер сообщений.

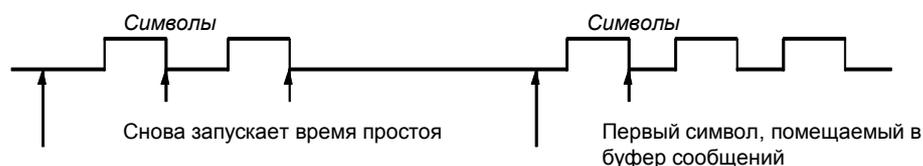
Команда приема поддерживает несколько условий начала сообщения:

1. *Распознавание бездействия линии:* Условие бездействия линии определяется как время простоя или перерыва в работе линии передачи. Прием начинается, если линия связи находилась в простое или имела перерыв в работе в течение некоторого количества миллисекунд, указанного в SMW90 или SMW190. Когда в вашей программе выполняется команда приема, то функция приема сообщений иницирует поиск условия бездействия линии. Если прием каких-либо символов происходит до истечения времени простоя линии, то функция приема сообщений игнорирует эти символы и вновь запускает таймер простоя линии с временем, полученным из SMW90 или SMW190. См. рис. 6–11. По истечении времени простоя линии функция приема сообщений сохраняет все последующие полученные символы в буфере сообщений.

Время простоя линии всегда должно быть больше, чем время, необходимое для передачи одного символа (начальный бит, биты данных, биты контроля четности и завершения) при заданной скорости передачи. Типичным значением для времени простоя линии является время, необходимое для передачи трех символов при заданной скорости передачи.

Распознавание бездействия линии используется в качестве начального условия для двоичных протоколов, для протоколов, не имеющих определенного начального символа, или в случае, когда протокол требует минимального времени между сообщениями.

Настройка:  $il = 1$ ,  $sc = 0$ ,  $bk = 0$ , SMW90/SMW190 = простой линии в мс



Команда приема выполняется:  
запускается отсчет времени простоя

Обнаружено время простоя:  
запускается функция приема сообщений

Рис. 6–11. Запуск команды приема с распознаванием бездействия линии

2. *Распознавание начального символа:* начальный символ – это любой символ, используемый в качестве первого символа сообщения. Сообщение начинается, когда принимается начальный символ, указанный в SMB88 или SMB188. Функция приема сообщений хранит начальный символ в буфере приема в качестве первого символа сообщения. Функция приема сообщений игнорирует все символы, принятые ранее начального символа. Начальный символ и все принятые после него символы сохраняются в буфере сообщения.

Обычно распознавание начального символа используется в протоколах ASCII, в которых все сообщения начинаются с одного и того же символа.

Настройка: il = 0, sc = 1, bk = 0, SMW90/SMW190 = не имеют значения, SMB88/SMB188 = начальный символ

3. *Простой линии и начальный символ:* Команда приема может начинать сообщение комбинацией простоя линии и начального символа. При выполнении команды приема функция приема сообщений ищет условие простоя линии. После нахождения условия простоя линия функция приема сообщений ожидает заданного начального символа. Если принимается любой символ, отличный от начального, функция приема сообщений снова начинает поиск условия простоя линии. Все символы, принятые до обнаружения условия простоя линии и до приема начального символа игнорируются. Начальный символ помещается в буфер сообщения вместе со всеми последующими символами.

Время простоя линии всегда должно быть больше, чем время, необходимое для передачи одного символа (начальный бит, биты данных, биты контроля четности и завершения) при заданной скорости передачи. Типичным значением для времени простоя линии является время, необходимое для передачи трех символов при заданной скорости передачи.

Обычно этот вид начальных условий используется с протоколом, который задает минимальное и максимальное время между сообщениями, и первым символом сообщения является адрес или что-либо иное, определяющее конкретное устройство. Это особенно полезно, если реализуется протокол, в котором в обмене данными участвует несколько устройств. В этом случае команда приема запускает прерывание только в том случае, когда сообщение принимается для конкретного адреса или конкретного устройства, указанного начальным символом.

Настройка: il = 1, sc = 1, bk = 0, SMW90/SMW190 > 0, SMB88/SMB188 = начальный символ

4. *Обнаружение паузы:* Признаком паузы является прием данных с нулевым значением в течение времени, превышающего длительность передачи полного символа. Время передачи полного символа определяется как общее время для передачи начального бита, битов данных и битов контроля четности и завершения. Если команда приема была сконфигурирована так, что началом сообщения является прием условия паузы, то все символы, принятые после обнаружения условия паузы, помещаются в буфер сообщения. Все символы, принятые до обнаружения условия паузы, игнорируются.

Обычно обнаружение паузы используется в качестве начального условия только в том случае, если этого требует протокол.

Настройка: il = 0, sc = 0, bk = 1, SMW90/SMW190 = не имеют значения, SMB88/SMB188 = не имеют значения

5. *Пауза и начальный символ:* Команда приема может быть сконфигурирована так, чтобы начинать прием символов после обнаружения условия паузы, за которым следует начальный символ. После обнаружения условия паузы функция приема сообщений ищет заданный начальный символ. Если принимается любой символ, кроме начального, то функция приема сообщений снова начинает искать условие паузы. Все символы, принятые до обнаружения условия паузы и до получения начального символа, игнорируются. Начальный символ помещается в буфер сообщений вместе со всеми последующими символами.

Настройка: il = 0, sc = 1, bk = 1, SMW90/SMW190 = не имеют значения, SMB88/SMB188 = начальный символ

6. *Любой символ*: Команда приема может быть сконфигурирована так, чтобы немедленно принимать все символы и помещать их в буфер сообщения. Это особый случай обнаружения бездействия линии. В этом случае время простоя линии (SMW90 или SMW190) устанавливается в ноль. Это заставляет команду приема начинать прием символов немедленно после начала исполнения.

Настройка:  $il = 1, sc = 0, bk = 0, SMW90/SMW190 = 0, SMB88/SMB188 = \text{не имеют значения}$

Если в качестве начала сообщения используется любой символ, то можно использовать таймер сообщений для контроля времени приема сообщения. Это полезно в случаях, когда для реализации основной части протокола используется свободно программируемый обмен данными, и требуется контроль времени на случай, если в течение определенного промежутка времени не будет получено ответа от slave-устройства. Таймер сообщения запускается, когда выполняется команда приема, так как время простоя линии было установлено в ноль. Таймер сообщения заканчивает отсчет времени и завершает функцию приема сообщений, если не выполнено какое-нибудь другое условие окончания приема.

Настройка:  $il = 1, sc = 0, bk = 0, SMW90/SMW190 = 0, SMB88/SMB188 = \text{не имеют значения}, c/m = 1, tmr = 1, SMW92 = \text{выдержка времени таймера в мс}$

Команда приема поддерживает несколько способов завершения сообщения. Сообщение может быть завершено одним или комбинацией нескольких из следующих условий:

1. *Обнаружение конечного символа*: Конечный символ – это любой символ, используемый для обозначения конца сообщения. После обнаружения начального условия команда приема проверяет каждый полученный символ на совпадение с конечным символом. Если получен конечный символ, то он помещается в буфер сообщения, и прием завершается.

Обычно обнаружение конечного символа используется в протоколах ASCII, где каждое сообщение завершается определенным символом. Для завершения сообщения обнаружение конечного символа можно использовать в сочетании с межсимвольным таймером, таймером сообщения или обнаружением достижения максимального количества символов.

Настройка:  $ec = 1, SMB89/SMB189 = \text{конечный символ}$

2. *Межсимвольный таймер*: Время между символами – это промежуток времени от конца одного символа (стопового бита) до конца следующего символа (стопового бита). Если время между символами (включая второй символ) превышает количество миллисекунд, указанное в SMW92 или SMW192, то функция приема сообщений завершает работу. Межсимвольный таймер перезапускается после приема каждого символа. См. рис. 6–12.

Межсимвольный таймер можно использовать для завершения сообщения в протоколах, не имеющих специального символа конца сообщения. Этот таймер должен быть установлен на значение, большее, чем время передачи одного символа при выбранной скорости передачи, так как этот таймер всегда включает в себя время, необходимое для получения полного символа (начальный бит, биты данных, биты контроля четности и завершения).

Для завершения сообщения межсимвольный таймер можно использовать в сочетании с обнаружением конечного символа и обнаружением достижения максимального количества символов.

Настройка:  $c/m = 0, tmr = 1, SMW92/SMW192 = \text{выдержка времени в мс}$



Рис. 6–12. Использование межсимвольного таймера для завершения команды приема

3. **Таймер сообщения:** Таймер сообщения завершает сообщение через определенное время после его начала. Таймер сообщения запускается, как только выполнено начальное условие или условия для функции приема сообщений. Таймер сообщения завершает отсчет времени по истечении количества миллисекунд, указанных в SMW92 или SMW192. См. рис. 6–13.

Обычно таймер сообщения используется, если устройства связи не могут гарантировать, что между символами не будет временных разрывов, или при работе через модемы. Для модемов можно использовать таймер сообщения, чтобы задать максимально допустимое время для приема сообщения после его начала. Типичным значением для таймера сообщения является полуторакратное время, необходимое для приема наиболее длинного возможного сообщения при выбранной скорости передачи.

Для завершения сообщения таймер сообщения можно использовать в сочетании с обнаружением конечного символа и обнаружением достижения максимального количества символов.

Настройка:  $c/m = 1$ ,  $tmr = 1$ , SMW92/SMW192 = выдержка времени в мс



Рис. 6–13. Использование таймера сообщений для завершения команды приема

4. **Значение для наибольшего числа символов:** Команде приема можно сообщить максимальное количество символов, которое должно быть принято (SMB94 или SMB194). Если это значение достигнуто или превышено, то функция приема сообщений завершает работу. Команда приема требует, чтобы пользователь определил максимальное количество символов, даже если оно специально не используется в качестве условия завершения. Это необходимо потому, что команде приема необходимо знать максимальный размер принимаемого сообщения, чтобы данные пользователя, помещенные после буфера сообщений, не были перезаписаны.  
Максимальное количество символов может использоваться для завершения сообщений в протоколах, в которых длина сообщений известна и всегда одинакова. Максимальное количество символов всегда используется в сочетании с обнаружением конечного символа, межсимвольного таймера или таймера сообщения.
5. **Ошибки, обнаруженные при проверке четности:** Функция приема автоматически завершает работу, если аппаратура сообщает об ошибке четности в принятом символе. Ошибки при проверке четности возможны только в том случае, если в SMB30 или SMB130 активизирована проверка четности. Эту функцию невозможно выключить.
6. **Завершение по инициативе пользователя:** Программа пользователя может завершить функцию приема сообщений, выполняя другую функцию приема сообщений и установив в ноль бит деблокировки (EN) в SMB87 или SMB187. Это приводит к немедленному завершению функции приема сообщений.

### Прием данных, управляемый прерываниями

Для достижения полной гибкости в поддержке протоколов вы также можете принимать данные под управлением прерываний. Каждый принимаемый символ генерирует прерывание. Принятый символ помещается в SMB2, а результат контроля четности (если активизирован) помещается в SM3.0 непосредственно перед исполнением программы обработки прерывания, назначенной событию «Символ принят». SMB2 – это буфер для принятого символа при свободно программируемом обмене данными. Каждый символ, принятый в режиме свободно программируемого обмена данными, помещается по этому адресу для облегчения доступа к нему из программы пользователя. SMB3 используется для режима свободно программируемого обмена данными и содержит бит ошибки четности, который устанавливается, когда в принятом символе обнаруживается ошибка четности. Все остальные биты этого байта зарезервированы. Используйте этот бит для отклонения сообщения или для генерирования отрицательного квитирования этого сообщения.

Если управление с помощью прерываний используется при высоких скоростях передачи (от 38,4 до 115,2 Кбод), то время между прерываниями очень мало. Например, прерывание для скорости 38,4 Кбод составляет 260 микросекунд, для скорости 57,6 Кбод – 173 микросекунды, а для 115,2 Кбод – 86 микросекунд. Во избежание потери символов обеспечьте краткость программ обработки прерываний или используйте команду приема.



**Совет**

SMB2 и SMB3 совместно используются портами 0 и 1. Когда прием символа в порт 0 приводит к исполнению программы обработки прерывания, назначенной этому событию (прерывающее событие 8), SMB2 содержит символ, принятый портом 0, а SMB3 содержит результат контроля четности этого символа. Когда прием символа в порт 1 приводит к исполнению программы обработки прерывания, назначенной этому событию (прерывающее событие 25), SMB2 содержит символ, принятый портом 1, а SMB3 содержит результат контроля четности этого символа.

**Пример: Команды передачи и приема**

<p>M A I N</p>	<p>Network 1</p>	<p>Network 1 //Эта программа принимает строку символов, пока не //будет получен символ перевода строки. //Затем сообщение возвращается отправителю.</p> <p>LD SM0.1 //В первом цикле: MOV B 16#09, SMB30 //1. Инициализировать свободно // программируемый обмен данными: // - выбрать 9600 Бод. // - выбрать 8 битов данных. // - выбрать отсутствие контроля // четности.</p> <p>MOV B 16#B0, SMB87 //2. Инициализировать управляющий // байт для приема сообщений: // - Прием сообщений разблокирован. // - Обнаруживать символ конца // сообщения. // - Обнаруживать условие // бездействия линии, как условие // начала сообщения.</p> <p>MOV B 16#0A, SMB89 //3. В качестве символа конца // сообщения установить OAn // (перевод строки).</p> <p>MOV W +5, SMW90 //4. Установить время ожидания // бездействия линии равным 5 мс.</p> <p>MOV B 100, SMB94 //5. Установить максимальное // количество символов равным 100.</p> <p>ATCH INT_0, 23 //6. Назначить прерывание 0 // событию "Прием завершен".</p> <p>ATCH INT_2, 9 //7. Назначить прерывание 2 // событию "Передача завершена".</p> <p>ENI //8. Разблокировать прерывания со // стороны пользователя.</p> <p>RCV VB100, 0 //9. Разблокировать блок приема с // буфером в VB100.</p>
----------------------------	------------------	---

Пример: Команды передачи и приема, продолжение		
I N T 0	<p><b>Network 1</b></p>	<p>Network 1 //Программа обработки прерываний для //завершения приема:  //1. Если статус приема показывает, что принят  // конечный символ, то назначить  // 10-миллисекундный таймер для запуска  // передачи и возврата.  //2. Если прием завершен по любой другой  // причине, то начать новый прием.</p> <p>LDB= SMB86, 16#20  MOVB 10, SMB34  ATCH INT_1, 10  CRETI  NOT  RCV VB100, 0</p>
I N T 1	<p><b>Network 1</b></p>	<p>Network 1 //Прерывание, управляемое 10-миллисекундным //таймером:  //1. Отсоединить прерывание, управляемое //таймером.  //2. Передать сообщение обратно пользователю.</p> <p>LD SM0.0  DTCH 10  XMT VB100, 0</p>
I N T 2	<p><b>Network 1</b></p>	<p>Network 1 //Прерывание при завершении передачи:  //Разблокировать новый прием.</p> <p>LD SM0.0  RCV VB100, 0</p>

## Команды получения и установки адреса порта

Команда получения адреса порта (GPA) считывает адрес станции из порта CPU S7-200, указанного в параметре PORT, и помещает значение по адресу, указанному в ADDR.

Команда установки адреса порта (SPA) устанавливает адрес порта станции (PORT) на значение, указанное в ADDR. Новый адрес не сохраняется постоянно. После нового пуска соответствующий порт возвращается к последнему адресу (к тому, который был загружен системным блоком).

**Сбойные состояния, устанавливающие ENO = 0 :**

- 0006 (косвенный адрес)
- 0004 (попытка выполнить команду установки адреса порта в программе обработки прерываний)

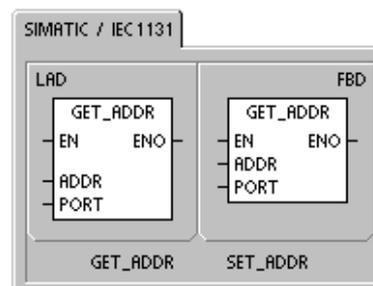


Таблица 6–14. Допустимые операнды для команд получения и установки адреса порта

Входы/выходы	Тип данных	Операнды
ADDR	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа (Постоянное значение действительно только для команды установки адреса порта.)
PORT	BYTE	константа для CPU 221, CPU 222, CPU 224: 0 для CPU 224XP, CPU 226: 0 или 1

## Команды сравнения

### Сравнение числовых величин

Команды сравнения используются для сравнения двух величин:

IN1 = IN2    IN1 >= IN2    IN1 <= IN2  
 IN1 > IN2    IN1 < IN2    IN1 <> IN2

Операции сравнения байтов не учитывают знака.  
 Операции сравнения целых учитывают знак.  
 Операции сравнения двойных слов учитывают знак.  
 Операции сравнения вещественных чисел учитывают знак.

*Для LAD и FBD:* Если сравнение истинно, команда сравнения включает контакт (LAD) или выход (FBD).

*Для STL:* Если сравнение истинно, то команда сравнения загружает «1» в вершину стека или выполняет логическое сопряжение значения «1» со значением в вершине стека в соответствии с таблицей истинности для И или ИЛИ (STL).

Если вы работаете с командами сравнения IEC, то вы можете использовать на входах различные типы данных. Однако значения на обоих входах должны быть одного типа.

#### Указание

При следующих условиях речь идет о фатальных ошибках, которые приводят к тому, что S7-200 немедленно останавливает обработку вашей программы:

- Обнаружен недопустимый косвенный адрес (в любой команде сравнения)
- Обнаружено недопустимое вещественное число (например, NAN) (сравнение вещественных чисел)

Во избежание появления таких состояний обратите внимание на правильную инициализацию всех указателей и величин, содержащих вещественные числа, перед выполнением команды сравнения.

Команды сравнения выполняются независимо от состояния потока сигнала.

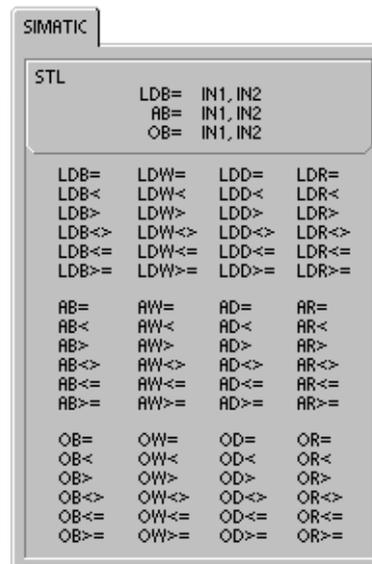
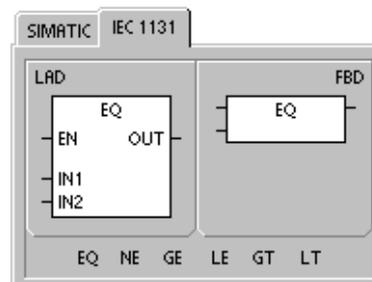
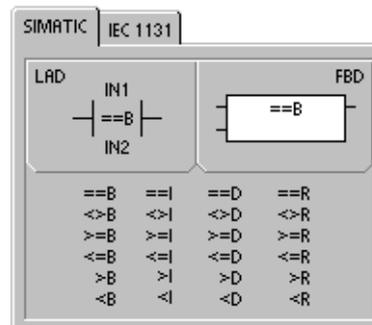


Таблица 6–15. Допустимые операнды для команд сравнения

Входы/выходы	Тип	Операнды
IN1, IN2	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, константа
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, константа
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, константа
Выход (или OUT)	BOOL	I, Q, V, M, SM, S, T, C, L, поток сигнала

**Пример: Команды сравнения**

<p><b>Network 1</b></p>	<p>Network 1 //Повернуть аналоговый потенциометр 0, чтобы //изменить значение байта SMB28. //Q0.0 активен, если значение SMB28 меньше //или равно 50. //Q0.1 активен, если значение SMB28 больше //или равно 150. //Индикатор состояния включен, если //сравнение истинно.</p>
<p><b>Network 2</b></p>	<p>LD I0.0 LPS AB&lt;= SMB28, 50 = Q0.0 LPP AB&gt;= SMB28, 150 = Q0.1</p>
<p><b>Network 3</b></p>	<p>Network 2 //Загрузить адреса из памяти переменных с //небольшими значениями, чтобы сравнения //были ложными и индикаторы состояния //выключились.</p> <p>LD I0.1 MOVW -30000, VW0 MOVD -200000000, VD2 MOVR 1.012E-006, VD6</p>
<p><b>Network 4</b></p>	<p>Network 3 //Загрузить адреса из памяти переменных с //большими значениями, чтобы сравнения были //истинными и индикаторы состояния //включились.</p> <p>LD I0.2 MOVW +30000, VW0 MOVD -100000000, VD2 MOVR 3.141593, VD6</p>
	<p>Network 4 //Сравнение целых чисел проверяет, истинно // ли, что VW0 &gt; +10000. //с помощью констант программы показываются //различные типы данных. Вы можете также // сравнить две величины, хранящиеся в //программируемой памяти, напр., VW0 &gt;</p> <p>VW100 LD I0.3 LPS AW&gt; VW0, +10000 = Q0.2 LRD AD&lt; -150000000, VD2 = Q0.3 LPP AR&gt; VD6, 5.001E-006 = Q0.4</p>

## Сравнение строк

Команда сравнения строк сравнивает между собой две строки символов ASCII:

IN1 = IN2     IN1 <> IN2

Если сравнение истинно, команда сравнения включает контакт (LAD) или выход (FBD) или загружает «1» в вершину стека или выполняет логическое сопряжение значения «1» со значением в вершине стека в соответствии с таблицей истинности для И или ИЛИ (STL).

### Указание

При следующих условиях речь идет о фатальных ошибках, которые приводят к тому, что S7-200 немедленно останавливает обработку вашей программы:

- Обнаружен недопустимый косвенный адрес (в любой команде сравнения)
- Встретилась строка, содержащая более 254 символов (команда сравнения строк)
- Строка, начальный адрес которой и длина таковы, что она не помещается в указанной области памяти (команда сравнения строк)

Во избежание появления таких состояний обратите внимание на правильную инициализацию всех указателей и адресов в памяти, которые должны содержать строки символов ASCII, перед выполнением команды сравнения, использующей эти значения. Обеспечьте, чтобы буфер, зарезервированный для строки символов ASCII, полностью помещался в указанной области памяти.

Команды сравнения выполняются независимо от состояния потока сигнала.

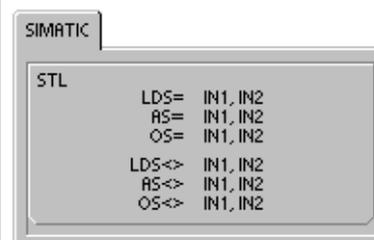
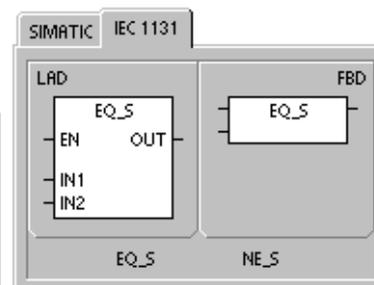
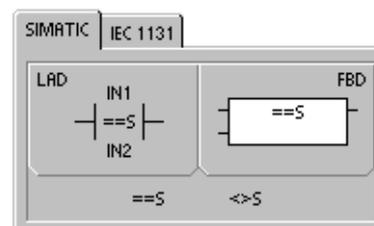


Таблица 6–16. Допустимые операнды для команд сравнения строк

Входы/выходы	Тип	Операнды
IN1	STRING	VB, LB, *VD, *LD, *AC, константа
IN2	STRING	VB, LB, *VD, *LD, *AC
Выход (OUT)	BOOL	I, Q, V, M, SM, S, T, C, L, поток сигнала

## Команды преобразования

### Стандартные команды преобразования

#### Команды преобразования чисел

Команды преобразования байта в целое (BTI), целого в байт (ITB), целого в двойное целое (ITD), двойного целого в целое (DTI), двойного целого в вещественное (DTR), двоично-десятичного в целое (BCDI) и целого в двоично-десятичное (IBCD) преобразуют входную величину IN в указанный формат и сохраняют выходное значение по адресу, указанному в OUT. Например, вы можете преобразовать двойное целое в вещественное число. Вы можете также осуществлять преобразования между целым двоично-десятичным (BCD) форматом.

#### Округление и округление отбрасыванием

Команда округления (ROUND) преобразует вещественное число IN в двойное целое число и помещает округленный результат в переменную, указанную в OUT.

Команда округления отбрасыванием дробной части (TRUNC) преобразует вещественное число IN в двойное целое число и помещает целую часть результата в переменную, указанную в OUT.

#### Генерирование комбинации двоичных разрядов для семисегментного индикатора

Команда генерирования комбинации двоичных разрядов для семисегментного индикатора (SEG) дает возможность создать битовую комбинацию, с помощью которой высвечиваются сегменты семисегментного индикатора.

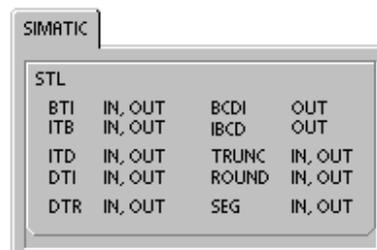
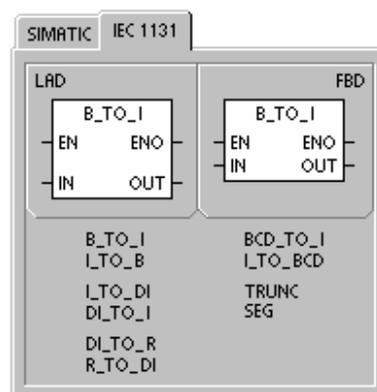
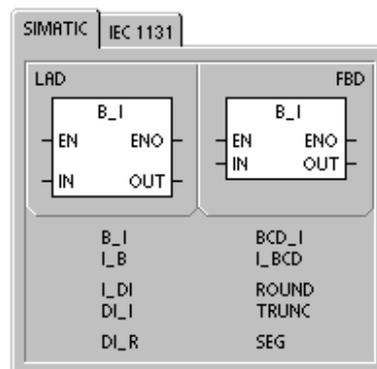


Таблица 6–17. Допустимые операнды для стандартных команд преобразования

Входы/выходы	Тип данных	Операнды
IN	BYTE WORD, INT  DINT REAL	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа IW, QW, VW, MW, SMW, SW, T, C, LW, AIW, AC, *VD, *LD, *AC, константа ID, QD, VD, MD, SMD, SD, LD, HC, AC, *VD, *LD, *AC, константа ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, константа
OUT	BYTE WORD, INT DINT, REAL	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

### Принцип действия команды преобразования BCD в целое и целого в BCD

<p>Команда преобразования BCD в целое число (BCDI) преобразует входное число, представленное в двоично-десятичном формате, IN в целое число и загружает результат в переменную, указанную в OUT. Допустимый диапазон для IN: от 0 до 9999 (BCD).</p> <p>Команда преобразования целого числа в BCD (IBCD) преобразует входное целое число IN в двоично-десятичный формат и загружает результат в переменную, указанную в OUT. Допустимый диапазон для IN: от 0 до 9999 (целое).</p>	<p><b>Сбойные состояния, устанавливающие ENO = 0</b></p> <ul style="list-style-type: none"> <li>■ SM1.6 (недопустимое значение BCD)</li> <li>■ 0006 (косвенный адрес)</li> </ul> <p><b>Биты специальной памяти (SM), на которые оказывается воздействие:</b></p> <ul style="list-style-type: none"> <li>■ SM1.6 (недопустимое значение BCD)</li> </ul>
--	--

### Принцип действия команды преобразования двойного целого в вещественное

<p>Команда преобразования двойного целого числа в вещественное (DTR) преобразует 32-битовое целое со знаком IN в 32-битовое вещественное число и помещает результат в переменную, указанную в OUT.</p>	<p><b>Сбойные состояния, устанавливающие ENO = 0</b></p> <ul style="list-style-type: none"> <li>■ 0006 (косвенный адрес)</li> </ul>
--	---

### Принцип действия команды преобразования двойного целого числа в целое

<p>Команда преобразования двойного целого числа в целое (DTI) преобразует двойное целое число IN в целое число и помещает результат в переменную, указанную в OUT.</p> <p>Если преобразованное вами значение слишком велико, чтобы быть представленным на выходе, то устанавливается бит переполнения, а выход не изменяется.</p>	<p><b>Сбойные состояния, устанавливающие ENO = 0</b></p> <ul style="list-style-type: none"> <li>■ SM1.1 (переполнение)</li> <li>■ 0006 (косвенный адрес)</li> </ul> <p><b>Биты специальной памяти, на которые действует команда:</b></p> <ul style="list-style-type: none"> <li>■ SM1.1 (переполнение)</li> </ul>
---	---

### Принцип действия команды преобразования целого числа в двойное целое

<p>Команда преобразования целого числа в двойное целое (ITD) преобразует целое число IN в двойное целое число и помещает результат в переменную, указанную в OUT. Знак распространяется.</p>	<p><b>Сбойные состояния, устанавливающие ENO = 0</b></p> <ul style="list-style-type: none"> <li>■ 0006 (косвенный адрес)</li> </ul>
--	---

### Принцип действия команды преобразования байта в целое число

<p>Команда преобразования байта в целое число (BTI) преобразует байт IN в целое число и помещает результат в переменную, указанную в OUT. Байт не имеет знака, поэтому распространение знака не происходит.</p>	<p><b>Сбойные состояния, устанавливающие ENO = 0</b></p> <ul style="list-style-type: none"> <li>■ 0006 (косвенный адрес)</li> </ul>
---	---

### Принцип действия команды преобразования целого числа в байт

<p>Команда преобразования целого числа в байт (ITB) преобразует слово IN в байт и помещает результат в переменную, указанную в OUT. Преобразуются значения от 0 до 255. Все остальные значения приводят к переполнению и не влияют на выход.</p>	<p><b>Сбойные состояния, устанавливающие ENO = 0</b></p> <ul style="list-style-type: none"> <li>■ SM1.1 (переполнение)</li> <li>■ 0006 (косвенный адрес)</li> </ul> <p><b>Биты специальной памяти, на которые действует команда:</b></p> <ul style="list-style-type: none"> <li>■ SM1.1 (переполнение)</li> </ul>
--	---



#### Совет

Для преобразования целого числа в вещественное используйте команду преобразования целого числа в двойное целое, а затем команду преобразования двойного целого числа в вещественное.

**Принцип действия команд округления и округления отбрасыванием**

<p>Команда округления (ROUND) преобразует вещественное число IN в двойное целое число и помещает результат в переменную, указанную в OUT. Если дробная часть равна 0,5 или больше, то число округляется в большую сторону.</p> <p>Команда округления отбрасыванием (TRUNC) преобразует вещественное число IN в двойное целое и помещает результат в переменную, указанную в OUT. Преобразуется только целая часть вещественного числа, а дробная часть отбрасывается.</p>	<p><b>Сбойные состояния, устанавливающие ENO = 0</b></p> <ul style="list-style-type: none"> <li>■ SM1.1 (переполнение)</li> <li>■ 0006 (косвенный адрес)</li> </ul> <p><b>Биты специальной памяти, на которые действует команда:</b></p> <ul style="list-style-type: none"> <li>■ SM1.1 (переполнение)</li> </ul>
---	---

Если преобразуемое вами значение не является допустимым вещественным числом или слишком велико, чтобы быть представленным на выходе, то устанавливается бит переполнения, а выход не меняется.

**Пример: Стандартные команды преобразования**

<p><b>Network 1</b></p> <p><b>Network 2</b></p>	<pre> Network 1 //Преобразовать дюймы в сантиметры: //1. Загрузить значение счетчика (дюймы) в // аккумулятор 1. //2. Преобразовать это значение в // вещественное число. //3. Умножить на 2,54 (преобразовать в // сантиметры). //4. Преобразовать значение обратно в целое // число. LD I0.0 ITD C10, AC1 DTR AC1, VD0 MOVR VD0, VD8 *R VD4, VD8 ROUND VD8, VD12  Network 2 //Преобразовать BCD в целое число LD I0.3 BCDI AC0     </pre>
---	---

Преобразование двойного целого в вещественное и округление		BCD в целое	
C10	101	Значение счетчика = 101 дюйм	AC0 1234
VD0	101.0	Значение счетчика (как вещественное число)	BCDI
VD4	2.54	Константа 2,54 (дюймы в сантиметры)	AC0 04D2
VD8	256.54	256,54 сантиметра как вещественное число	
VD12	257	257 сантиметров как двойное целое	

### Принцип действия команды генерирования комбинации двоичных разрядов для семисегментного индикатора

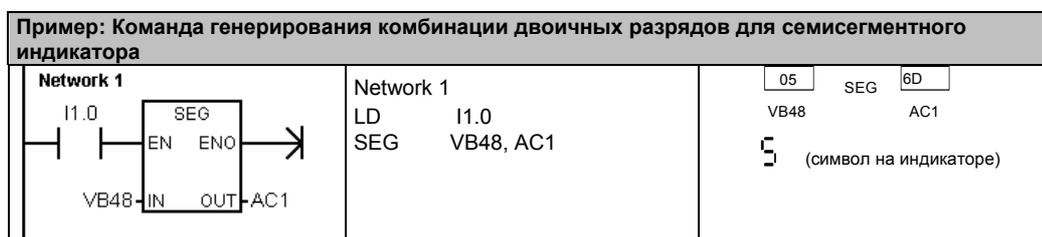
Для высвечивания сегментов семисегментного индикатора команда генерирования комбинации двоичных разрядов для семисегментного индикатора (SEG) преобразует символ (байт), указанный в IN, чтобы сгенерировать битовую комбинацию (байт) по адресу, указанному в OUT.

Высвечиваемые сегменты представляют собой символ в младшей цифре входного байта. На рис. 6–14 показано кодирование семисегментного дисплея, используемое командой SEG.	<b>Сбойные состояния, устанавливающие ENO = 0</b> ■ 0006 (косвенный адрес)
--	---

(IN) LSD	Отображение сегментов	(OUT)										
		-	g	f	e	d	c	b	a			
0	0	0	0	1	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	0	1	1	1	0
2	2	0	1	0	1	1	0	1	0	1	1	1
3	3	0	1	0	0	1	1	1	1	1	1	1
4	4	0	1	1	1	0	0	1	1	1	0	1
5	5	0	1	1	1	1	0	1	1	0	1	0
6	6	0	1	1	1	1	1	0	1	1	0	1
7	7	0	0	0	0	0	0	0	1	1	1	1

(IN) LSD	Отображение сегментов	(OUT)										
		-	g	f	e	d	c	b	a			
8	8	0	1	1	1	1	1	1	1	1	1	1
9	9	0	1	1	0	1	1	1	0	1	1	1
A	A	0	1	1	1	1	0	1	1	1	0	1
B	B	0	1	1	1	1	1	1	1	0	0	1
C	C	0	0	1	1	1	1	0	0	1	1	0
D	D	0	1	0	1	1	1	1	1	1	1	0
E	E	0	1	1	1	1	1	0	0	1	1	0
F	F	0	1	1	1	1	0	0	0	1	1	0

Рис. 6–14. Кодирование семисегментного индикатора



## Команды преобразования ASCII

Допустимые символы ASCII представляются шестнадцатеричными числами от 30 до 39 и от 41 до 46.

### Преобразования между ASCII и шестнадцатеричными числами

Команда преобразования строки символов ASCII в 16-ричное число (ATH) преобразует строку символов ASCII длиной LEN, начинающуюся с адреса IN, в шестнадцатеричные цифры, начиная с адреса OUT. Команда преобразования 16-ричного числа в строку символов ASCII (HTA) преобразует шестнадцатеричные цифры, начинающиеся с входного байта IN, в строку символов ASCII, которая начинается по адресу OUT. Количество шестнадцатеричных цифр, подлежащих преобразованию, задается длиной LEN.

Максимальное количество шестнадцатеричных цифр, которое может быть преобразовано, равно 255.

Допустимыми входными символами ASCII являются алфавитно-цифровые символы от 0 до 9 с шестнадцатеричными значениями от 30 до 39 и большие буквы от A до F с шестнадцатеричными значениями от 41 до 46.

**Сбойные состояния, устанавливающие ENO = 0**

- SM1.7 (недопустимый код ASCII) *только для ASCII в 16-ричный код*
- 0006 (косвенный адрес)
- 0091 (операнд вышел за пределы допустимого диапазона)

**Биты специальной памяти, на которые действует команда:**

- SM1.7 (недопустимый код ASCII)

### Преобразование числовых значений в ASCII

Команды преобразования целого числа в строку символов ASCII (ITA), преобразования двойного целого в строку символов ASCII (DTA) и преобразования вещественного числа в строку символов ASCII (RTA) преобразуют целое, двойное целое или вещественное число в символы ASCII.

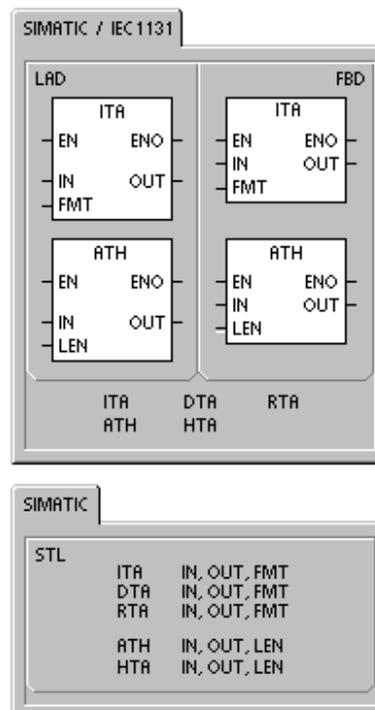
Таблица 6–18. Допустимые операнды для команд преобразования ASCII

Входы/выходы	Тип данных	Операнды
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, константа
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, константа
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, константа
LEN, FMT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC

### Принцип действия команды преобразования целого числа в строку символов ASCII

<p>Команда преобразования целого числа в строку символов ASCII (ITA) преобразует целое число, содержащееся в слове IN, в массив символов ASCII. Операнд FMT (формат) определяет точность преобразования справа от десятичной точки, а также форму представления разделителя целой и дробной части – в виде запятой или точки. Результат преобразования помещается в 8 последовательных байтов, начиная с адреса OUT.</p>	<p><b>Сбойные состояния, устанавливающие ENO = 0</b></p> <ul style="list-style-type: none"> <li>■ 0006 (косвенный адрес)</li> <li>■ Недопустимый формат</li> <li>■ <i>nnn</i> &gt; 5</li> </ul>
--	---

Массив символов ASCII всегда содержит 8 элементов.

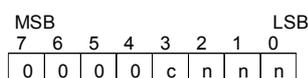


На рис. 6–15 описан операнд формата для команды преобразования целого числа в строку символов ASCII. Размер выходного буфера всегда равен 8 байтам. Количество цифр справа от десятичной точки в выходном буфере определяется полем *nnn*. Допустимый диапазон поля *nnn* составляет от 0 до 5. Задание 0 цифр справа от десятичной точки приводит к тому, что число отображается без десятичной точки. Для значений *nnn*, больших 5, выходной буфер заполняется пробелами ASCII. Бит *c* определяет использование запятой (*c*=1) или десятичной точки (*c*=0) в качестве разделителя целой и дробной части. Старшие 4 бита формата должны быть нулями.

На рис. 6–15 показаны примеры чисел, форматированных с помощью десятичной точки (*c* = 0), с тремя цифрами справа от десятичной точки (*nnn*=011). Выходной буфер форматирован в соответствии со следующими правилами:

- Положительные числа записываются в выходной буфер без знака.
- Отрицательные числа записываются в выходной буфер с ведущим знаком минус (-).
- Нули в старших разрядах слева от десятичной точки (кроме цифры, смежной с десятичной точкой) подавляются.
- Числа в выходном буфере выравниваются вправо.

FMT



*c* = запятая (1) или десятичная точка (0)  
*nnn* = цифры справа от десятичной точки

*in*=12

*in*=-123

*in*=1234

*in* = -12345

| Out |
|-----|-----|-----|-----|-----|-----|-----|-----|
| +1  | +2  | +3  | +4  | +5  | +6  | +7  |     |
|     |     | 0   | .   | 0   | 1   | 2   |     |
|     | -   | 0   | .   | 1   | 2   | 3   |     |
|     |     | 1   | .   | 2   | 3   | 4   |     |
| -   | 1   | 2   | .   | 3   | 4   | 5   |     |

Рис. 6–15. Операнд FMT для команды преобразования целого в строку символов ASCII (ITA)

### Принцип действия команды преобразования двойного целого в строку символов ASCII

Команда преобразования двойного целого в строку символов ASCII (DTA) преобразует двойное слово IN в массив символов ASCII. Операнд формат FMT определяет точность преобразования справа от десятичной точки. Результат преобразования помещается в 12 последовательных байтов, начиная с адреса OUT.	<b>Сбойные состояния, устанавливающие ENO = 0</b> <ul style="list-style-type: none"> <li>■ 0006 (косвенный адрес)</li> <li>■ Недопустимый формат</li> <li>■ <i>nnn</i> &gt; 5</li> </ul>
--	--

Размер выходного буфера всегда равен 12 байтам.

На рис. 6–16 описан операнд формата для команды преобразования двойного целого в строку символов ASCII. Количество цифр справа от десятичной точки в выходном буфере определяется полем *nnn*. Допустимый диапазон поля *nnn* составляет от 0 до 5. Задание 0 цифр справа от десятичной точки приводит к тому, что число отображается без десятичной точки. Для значений *nnn*, больших 5, выходной буфер заполняется символами пробела ASCII. Бит *c* определяет использование запятой (*c*=1) или десятичной точки (*c*=0) в качестве разделителя целой и дробной части. Старшие 4 бита формата должны быть нулями.

На рис. 6–16 показаны примеры чисел, форматированных с помощью десятичной точки (*c* = 0), с четырьмя цифрами справа от десятичной точки (*nnn*=100). Выходной буфер форматирован в соответствии со следующими правилами:

- Положительные числа записываются в выходной буфер без знака.
- Отрицательные числа записываются в выходной буфер с ведущим знаком минус (-).
- Нули в старших разрядах слева от десятичной точки (кроме цифры, смежной с десятичной точкой) подавляются.
- Числа в выходном буфере выравниваются вправо.

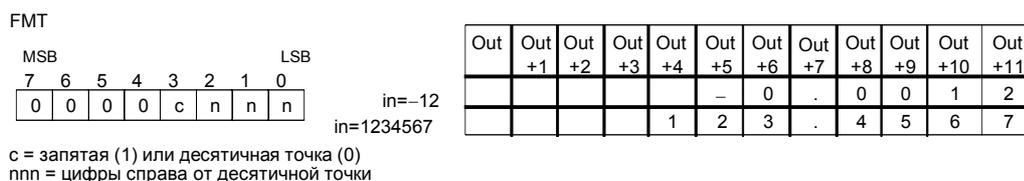


Рис. 6-16. Операнд FMT для команды преобразования целого числа в строку символов ASCII (DTA)

### Принцип действия команды преобразования вещественного числа в строку символов ASCII

<p>Команда преобразования вещественного числа в строку символов ASCII (RTA) преобразует число с плавающей точкой IN в символы ASCII. Формат FMT определяет точность преобразования справа от десятичной точки, а также форму представления разделителя целой и дробной части – в виде запятой или точки и размер выходного буфера.</p> <p>Результат преобразования помещается в выходном буфере, начиная с адреса OUT.</p>	<p><b>Сбойные состояния, устанавливающие ENO = 0</b></p> <ul style="list-style-type: none"> <li>■ 0006 (косвенный адрес)</li> <li>■ nnn &gt; 5</li> <li>■ ssss &lt; 3</li> <li>■ ssss &lt; число символов в OUT</li> </ul>
--	--

Длина (количество символов) результирующей строки символов ASCII соответствует размеру выходного буфера и может быть задана в диапазоне от 3 до 15 символов или байтов.

Формат чисел с плавающей точкой, используемый S7-200, поддерживает не более 7 значащих цифр. Попытка отобразить более 7 значащих цифр приводит к ошибке округления.

На рис. 6-17 описан операнд формата (FMT) для команды RTA. Размер выходного буфера определяется полем ssss. Размеры 0, 1 или 2 байта недопустимы. Количество цифр справа от десятичной точки в выходном буфере определяется полем nnn. Допустимый диапазон поля nnn составляет от 0 до 5. Задание 0 цифр справа от десятичной точки приводит к тому, что число отображается без десятичной точки. Выходной буфер заполняется символами пробела ASCII для значений nnn, больших 5, или если заданный выходной буфер слишком мал для хранения преобразованного значения. Бит c определяет использование запятой (c=1) или десятичной точки (c=0) в качестве разделителя целой и дробной части.

На рис. 6-17 также приведены примеры чисел, форматированных с помощью десятичной точки (c = 0), с одной цифрой справа от десятичной точки (nnn=001) и размером буфера, равным 6 байтам (ssss=0110). Выходной буфер форматруется в соответствии со следующими правилами:

- Положительные числа записываются в выходной буфер без знака.
- Отрицательные числа записываются в выходной буфер с ведущим знаком минус (-).
- Нули в старших разрядах слева от десятичной точки (кроме цифры, смежной с десятичной точкой) подавляются.
- Значение справа от десятичной точки округляется в соответствии с заданным количеством цифр справа от десятичной точки.
- Размер выходного буфера должен, по крайней мере, на три байта превышать количество цифр справа от десятичной точки.
- Числа в выходном буфере выравниваются вправо.

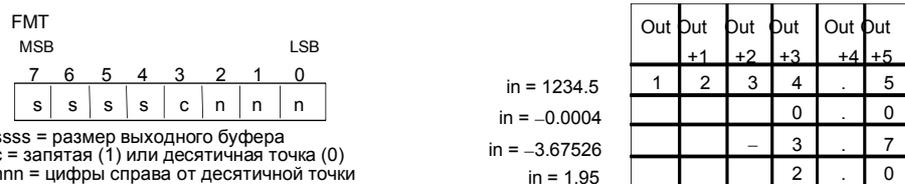


Рис. 6-17. Операнд FMT для команды преобразования вещественного числа в строку символов ASCII (RTA)

**Пример: Команда преобразования строки символов ASCII в 16-ричное число**

<p><b>Network 1</b></p>	<p><b>Network 1</b></p> <pre>LD I3.2 ATH VB30, VB40, 3</pre>																			
<table border="0"> <tr> <td>'3'</td> <td>'E'</td> <td>'A'</td> <td></td> <td></td> <td></td> </tr> <tr> <td>33</td> <td>45</td> <td>41</td> <td>ATH</td> <td>3E</td> <td>Ax</td> </tr> <tr> <td colspan="3">VB30</td> <td></td> <td colspan="3">VB40</td> </tr> </table>	'3'	'E'	'A'				33	45	41	ATH	3E	Ax	VB30				VB40			<p>Указание: X показывает, что половина байта не меняется</p>
'3'	'E'	'A'																		
33	45	41	ATH	3E	Ax															
VB30				VB40																

**Пример: Команда преобразования целого числа в строку символов ASCII**

<p><b>Network 1</b></p>	<p><b>Network 1</b></p> <pre>//Преобразовать целое число из VW2 //в 8 символов ASCII, начиная с VB10, //с использованием формата 16#0B //(запятая вместо десятичной точки, //за которой следуют 3 цифры).  LD I2.3 ITA VW2, VB10, 16#0B</pre>																											
<table border="0"> <tr> <td></td> <td></td> <td>'1'</td> <td>'2'</td> <td>'3'</td> <td>'4'</td> <td>'5'</td> </tr> <tr> <td>12345</td> <td>ITA</td> <td>20</td> <td>20</td> <td>31</td> <td>32</td> <td>2C</td> <td>33</td> <td>34</td> <td>35</td> </tr> <tr> <td>VW2</td> <td></td> <td>VB10</td> <td>VB11</td> <td>...</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </table>			'1'	'2'	'3'	'4'	'5'	12345	ITA	20	20	31	32	2C	33	34	35	VW2		VB10	VB11	...						
		'1'	'2'	'3'	'4'	'5'																						
12345	ITA	20	20	31	32	2C	33	34	35																			
VW2		VB10	VB11	...																								

**Пример: Команда преобразования вещественного числа в строку символов ASCII**

<p><b>Network 1</b></p>	<p><b>Network 1</b></p> <pre>//Преобразовать вещественное число из VD2 //в 10 символов ASCII, начиная с адреса // VB10, с помощью формата 16#A3 //(запятая вместо десятичной точки, // за которой следуют 3 цифры).  LD I2.3 RTA VD2, VB10, 16#A3</pre>																																
<table border="0"> <tr> <td></td> <td></td> <td>'1'</td> <td>'2'</td> <td>'3'</td> <td>'4'</td> <td>'5'</td> <td>'0'</td> </tr> <tr> <td>123.45</td> <td>RTA</td> <td>20</td> <td>20</td> <td>20</td> <td>31</td> <td>32</td> <td>33</td> <td>2E</td> <td>34</td> <td>35</td> <td>30</td> </tr> <tr> <td>VD2</td> <td></td> <td>VB10</td> <td>VB11</td> <td>...</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </table>			'1'	'2'	'3'	'4'	'5'	'0'	123.45	RTA	20	20	20	31	32	33	2E	34	35	30	VD2		VB10	VB11	...								
		'1'	'2'	'3'	'4'	'5'	'0'																										
123.45	RTA	20	20	20	31	32	33	2E	34	35	30																						
VD2		VB10	VB11	...																													

## Команды преобразования строк

### Преобразование числовых значений в строки

Команды преобразования целого числа в строку (ITS), преобразования двойного целого числа в строку (DTS), преобразования вещественного числа в строку (RTS) преобразуют целое, двойное целое или вещественное число (IN) в строку символов ASCII (OUT).

### Принцип действия команды преобразования целого числа в строку символов

Команда преобразования целого числа в строку символов (ITS) преобразует целое число IN в строку символов ASCII длиной 8 символов. Формат (FMT) определяет точность преобразования справа от десятичной точки, а также форму представления разделителя целой и дробной части – в виде запятой или точки. Результирующая строка записывается в 9 последовательных байтов, начиная с OUT. Подробную информацию о форматах строк символов вы найдете в главе 4.

#### Сбойные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)
- 0091 (операнд вышел за пределы допустимого диапазона)
- Недопустимый формат ( $nnn > 5$ )

На рис. 6–18 описан операнд формата для команды преобразования целого числа в строку. Длина выходной строки всегда равна 8 символам. Количество цифр справа от десятичной точки в выходном буфере определяется полем *nnn*. Допустимый диапазон поля *nnn* составляет от 0 до 5. Задание 0 цифр справа от десятичной точки приводит к тому, что число отображается без десятичной точки. Для значений *nnn*, больших 5, выходом является строка из 8 пробелов ASCII. Бит *c* определяет использование запятой ( $c=1$ ) или десятичной точки ( $c=0$ ) в качестве разделителя целой и дробной части. Старшие 4 бита формата должны быть нулями.

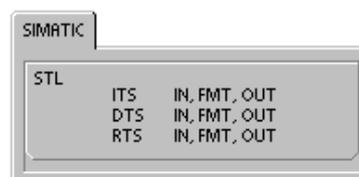
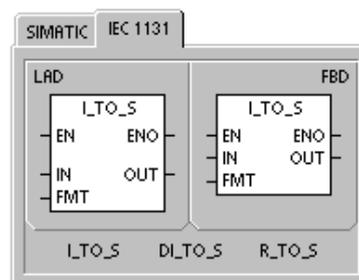
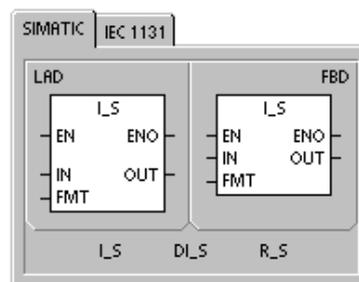
На рис. 6–18 показаны примеры чисел, форматированных с использованием десятичной точки ( $c = 0$ ), с тремя цифрами справа от десятичной точки ( $nnn = 011$ ). Значение OUT представляет собой длину строки.

Выходная строка форматируется в соответствии со следующими правилами:

- Положительные числа записываются в выходной буфер без знака.
- Отрицательные числа записываются в выходной буфер с ведущим знаком минус (-).
- Нули в старших разрядах слева от десятичной точки (кроме цифры, смежной с десятичной точкой) подавляются.
- Числа в выходной строке выравниваются вправо.

Таблица 6–19. Допустимые операнды для команд преобразования числовых значений в строки

Входы/выходы	Тип данных	Операнды
IN	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AIW, *VD, *LD, *AC, константа
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, константа
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, константа
FMT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа
OUT	STRING	VB, LB, *VD, *LD, *AC



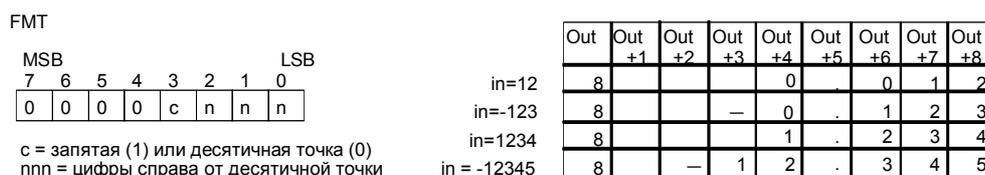


Рис. 6–18. Операнд FMT для команды преобразования целого числа в строку

### Принцип действия команды преобразования двойного целого числа в строку

<p>Команда преобразования двойного целого числа в строку (DTS) преобразует двойное целое число IN в строку ASCII длиной 12 символов. Формат (FMT) определяет точность преобразования справа от десятичной точки, а также форму представления разделителя целой и дробной части – в виде запятой или точки. Результирующая строка записывается в 13 последовательных байтов, начиная с адреса OUT. Дополнительную информацию вы найдете в главе 4, где описан формат строк.</p>	<p><b>Сбойные состояния, устанавливающие ENO = 0</b></p> <ul style="list-style-type: none"> <li>■ 0006 (косвенный адрес)</li> <li>■ 0091 (операнд вышел за пределы допустимого диапазона)</li> <li>■ Недопустимый формат (nnn &gt; 5)</li> </ul>
--	--

На рис. 6–19 описан операнд формата для команды преобразования двойного целого числа в строку. Длина выходной строки всегда равна 12 символам. Количество цифр справа от десятичной точки в выходном буфере определяется полем *nnn*. Допустимый диапазон поля *nnn* составляет от 0 до 5. Задание 0 цифр справа от десятичной точки приводит к тому, что число отображается без десятичной точки. Для значений *nnn*, больших 5, выходом является строка из 12 символов пробела ASCII. Бит *c* определяет использование запятой (*c*=1) или десятичной точки (*c*=0) в качестве разделителя целой и дробной части. Старшие 4 бита формата должны быть нулями.

На рис. 6–19 также показаны примеры чисел, форматированных с помощью десятичной точки (*c* = 0), с четырьмя цифрами справа от десятичной точки (*nnn* = 100). Значение OUT представляет собой длину строки. Выходная строка форматировается в соответствии со следующими правилами:

- Положительные числа записываются в выходной буфер без знака.
- Отрицательные числа записываются в выходной буфер с ведущим знаком минус (-).
- Нули в старших разрядах слева от десятичной точки (кроме цифры, смежной с десятичной точкой) подавляются.
- Числа в выходной строке выравниваются вправо.

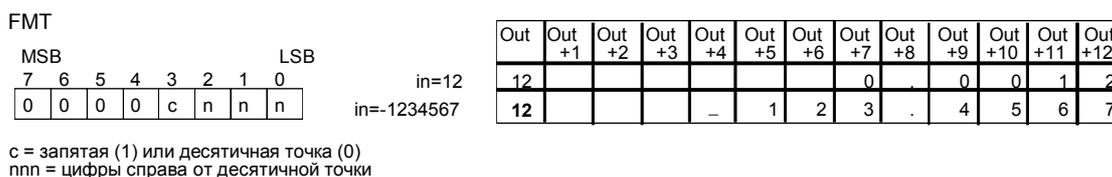


Рис. 6–19. Операнд FMT для команды преобразования двойного целого в строку

**Принцип действия команды преобразования вещественного числа в строку**

<p>Команда преобразования вещественного числа в строку (RTS) преобразует вещественное число IN в строку символов ASCII. Формат (FMT) определяет точность преобразования справа от десятичной точки, форму представления разделителя целой и дробной части – в виде запятой или точки, а также длину выходной строки.</p> <p>Результат преобразования помещается в строку, начинающуюся с OUT. Длина результирующей строки определяется в формате и может составлять от 3 до 15 символов. Дополнительную информацию вы найдете в главе 4, где описан формат строк.</p>	<p><b>Сбойные состояния, устанавливающие ENO = 0</b></p> <ul style="list-style-type: none"> <li>■ 0006 (косвенный адрес)</li> <li>■ 0091 (операнд вышел за пределы допустимого диапазона)</li> <li>■ Недопустимый формат:                     <ul style="list-style-type: none"> <li>nnn &gt; 5</li> <li>ssss &lt; 3</li> <li>ssss &lt; число требуемых символов</li> </ul> </li> </ul>
---	---

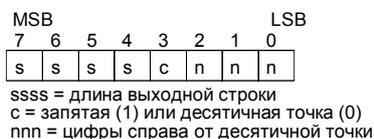
Формат вещественных чисел, используемый S7-200, поддерживает не более 7 значащих цифр. Попытка отобразить более 7 значащих цифр приводит к ошибке округления.

На рис. 6–20 описан операнд формата для команды преобразования вещественного числа в строку. Длина выходной строки определяется полем ssss. Размеры 0, 1 или 2 байта недопустимы. Количество цифр справа от десятичной точки в выходном буфере определяется полем ppp. Допустимый диапазон поля ppp составляет от 0 до 5. Задание 0 цифр справа от десятичной точки приводит к тому, что число отображается без десятичной точки. Выходная строка заполняется символами пробела ASCII для значений ppp, больших 5, или если указанная длина выходной строки слишком мала для хранения преобразованного значения. Бит c определяет использование запятой (c=1) или десятичной точки (c=0) в качестве разделителя целой и дробной части.

На рис. 6–20 приведены примеры чисел, форматированных с помощью десятичной точки (c = 0), с одной цифрой справа от десятичной точки (ppp = 001) и выходной строкой, имеющей длину 6 символов (ssss = 0110). Значение OUT представляет собой длину строки. Выходная строка форматируется в соответствии со следующими правилами:

- Положительные числа записываются в выходной буфер без знака.
- Отрицательные числа записываются в выходной буфер с ведущим знаком минус (-).
- Нули в старших разрядах слева от десятичной точки (кроме цифры, смежной с десятичной точкой) подавляются.
- Значение справа от десятичной точки округляется в соответствии с заданным количеством цифр справа от десятичной точки.
- Размер выходной строки должен, по крайней мере, на три байта превышать количество цифр справа от десятичной точки.
- Числа в выходной строке выравниваются вправо.

**FMT**



in=1234.5  
 in= -0.0004  
 in= -3.67526  
 in = 1.95

Out	Out +1	Out +2	Out +3	Out +4	Out +5	Out +6
6	1	2	3	4	.	5
6				0	.	0
6			-	3	.	7
6				2	.	0

Рис. 6–20. Операнд FMT для команды преобразования вещественного числа в строку

### Преобразование подстроки в числовые значения

Команды преобразования подстроки в целое число (STI), преобразования подстроки в двойное целое число (STD) и преобразования подстроки в вещественное число (STR) преобразуют строку символов IN, начинающуюся со смещения INDX, в целое, двойное целое или вещественное число OUT.

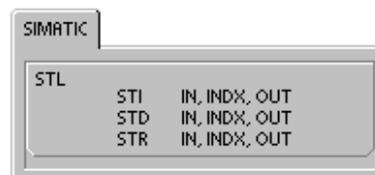
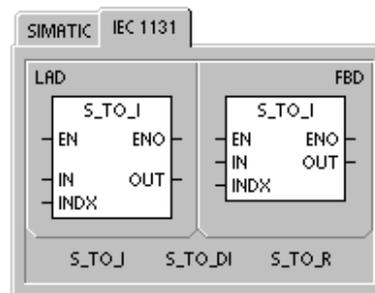
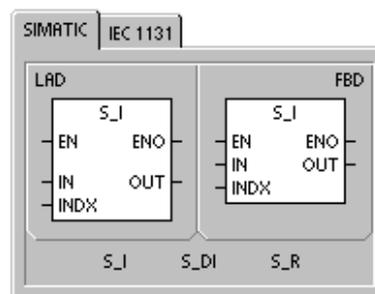
#### Сбойные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)
- 0091 (операнд вышел за пределы допустимого диапазона)
- 009B (индекс = 0)
- SM1.1 (переполнение)

Команды преобразования подстроки в целое и двойное целое число преобразуют строки, имеющие следующий формат: [пробелы] [+ или -] [цифры 0 - 9]

Команда преобразования подстроки в вещественное число преобразует строки, имеющие следующий формат: [пробелы] [+ или -] [цифры 0 - 9] [. или ,] [цифры 0 - 9]

Значение INDX обычно устанавливается в 1, так что преобразование начинается с первого символа строки. Значение INDX может быть и другим, чтобы начать преобразование с иной точки внутри строки. Этим можно воспользоваться, если входная строка содержит текст, не являющийся частью преобразуемого числа. Например, если строка имеет вид «Температура: 77.8», то установите INDX на значение 13, чтобы пропустить слово «Температура: » в начале строки.



Команда преобразования подстроки в вещественное число не преобразует строки, использующие научную нотацию или экспоненциальные формы вещественных чисел. Команда не генерирует ошибку переполнения (SM1.1), а преобразует строку в вещественное число до экспоненциальной функции, а затем завершает преобразование. Например, строка '1.234E6' преобразуется без ошибки в вещественное число 1.234.

Преобразование завершается при достижении конца строки или обнаружения первого недопустимого символа. Недопустимым является любой символ, отличный от цифры (0 - 9).

Ошибка переполнения (SM1.1) устанавливается, когда преобразование дает целое число, слишком большое для выходной величины. Например, команда преобразования подстроки в вещественное устанавливает ошибку переполнения, если входная строка дает значение, большее, чем 32767, или меньшее, чем -32768.

Ошибка переполнения (SM1.1) устанавливается также, если преобразование невозможно из-за отсутствия в строке допустимого значения. Например, если входная строка содержит 'A123', то команда преобразования устанавливает SM1.1 (переполнение), а выходная величина остается неизменной.

Таблица 6–20. Допустимые операнды для команд преобразования подстрок в числовые значения

Входы/выходы	Тип данных	Операнды
IN	STRING	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC, константа
INDX	BYTE	VB, IB, QB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа
OUT	INT DINT, REAL	VW, IW, QW, MW, SMW, SW, T, C, LW, AC, AQW, *VD, *LD, *AC VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Допустимые входные строки для целых и двойных целых чисел

Входная строка	Выходное целое
'123'	123
'-00456'	-456
'123.45'	123
'+2345'	2345
'000000123ABCD'	123

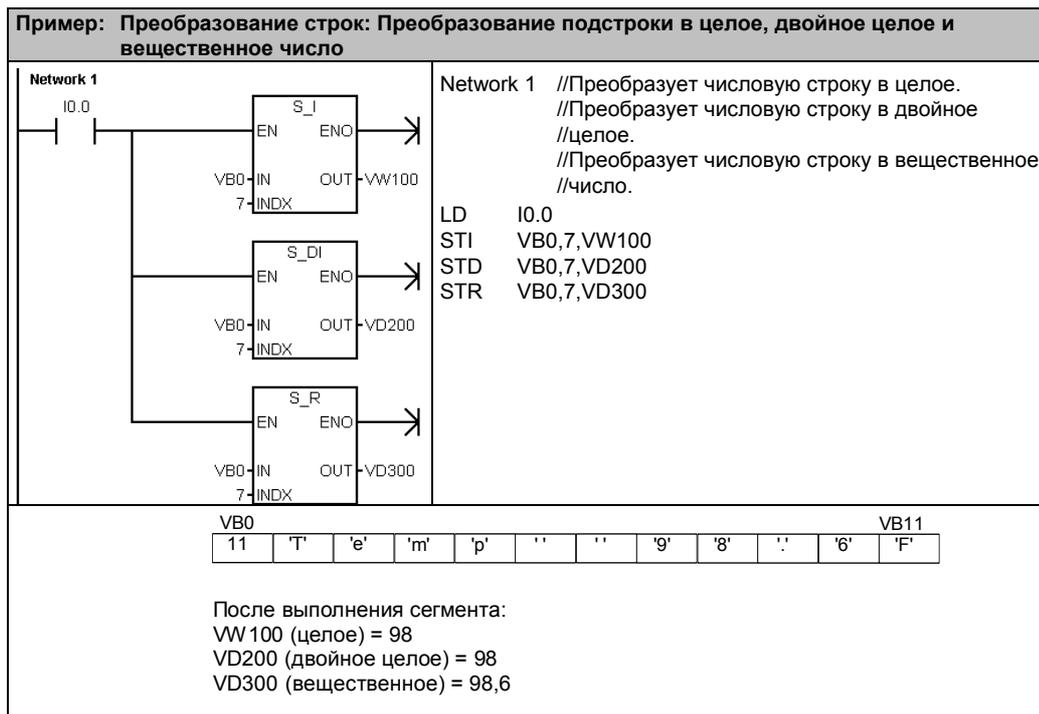
Допустимые входные строки для вещественных чисел

Входная строка	Выходное веществ. число
'123'	123.0
'-00456'	-456.0
'123.45'	123.45
'+2345'	2345.0
'00.000000123'	0.000000123

Недопустимые входные строки

Входная строка
'A123'
' '
'++123'
'+-123'
'+ 123'

Рис. 6-21. Примеры допустимых и недопустимых входных строк



## Команды кодирования и декодирования

### Кодирование

Команда кодирования (ENCO) записывает номер младшего установленного бита входного слова IN в младший полубайт (4 бита) выходного байта OUT.

### Декодирование

Команда декодирования (DECO) устанавливает в выходном слове OUT бит, соответствующий номеру бита, представленному младшим полубайтом (4 бита) входного байта IN. Все остальные биты выходного слова устанавливаются в 0.

### Биты специальной памяти и ENO

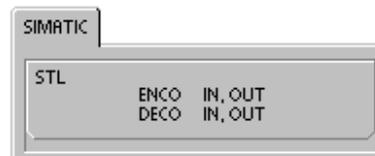
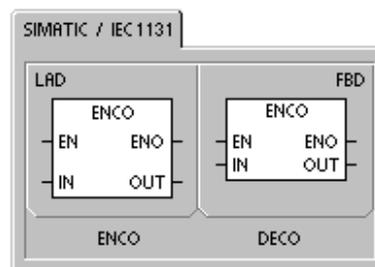
Для команд кодирования и декодирования следующие условия влияют на ENO.

**Сбойные состояния, устанавливающие ENO = 0**

- 0006 (косвенный адрес)

Таблица 6–21. Допустимые операнды для команд кодирования и декодирования

Входы/выходы	Типы данных	Операнды
IN	BYTE WORD	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, константа
OUT	BYTE WORD	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AQW, *VD, *LD, *AC



### Пример: Команды декодирования и кодирования

<p><b>Network 1</b></p>	<p><b>Network 1</b> //Аккумулятор 2 содержит биты ошибок.                  //1. Команда DECO устанавливает в VW40                  // бит, соответствующий этому коду                  // ошибки.                  //2. Команда ENCO преобразует младший                  // установленный бит в код ошибки,                  // который сохраняется в VB50.</p> <p>LD I3.1                  DECO AC2, VW40                  ENCO AC3, VB50</p>
<p>AC2 <span style="border: 1px solid black; padding: 2px;">3</span></p> <p style="text-align: center;">DECO <span style="border: 1px solid black; padding: 2px;">15 3 0</span></p> <p>VW40 <span style="border: 1px solid black; padding: 2px;">0000 0000 0000 1000</span></p>	<p>AC3 <span style="border: 1px solid black; padding: 2px;">15 9 0</span></p> <p style="text-align: center;">ENCO <span style="border: 1px solid black; padding: 2px;">1000 0010 0000 0000</span></p> <p>VB50 <span style="border: 1px solid black; padding: 2px;">9</span></p>

## Команды счета

### Счетчик SIMATIC

#### Прямой счет

Команда прямого счета (CTU) увеличивает текущее значение счетчика при появлении нарастающего фронта на входе (CU). Когда текущее значение Sxx больше или равно предустановленному значению PV, бит счетчика Sxx устанавливается. Счетчик сбрасывается, когда включается вход сброса (R), или когда выполняется команда сброса. Счетчик прекращает счет при достижении максимального значения (32767).

#### Принцип действия в STL:

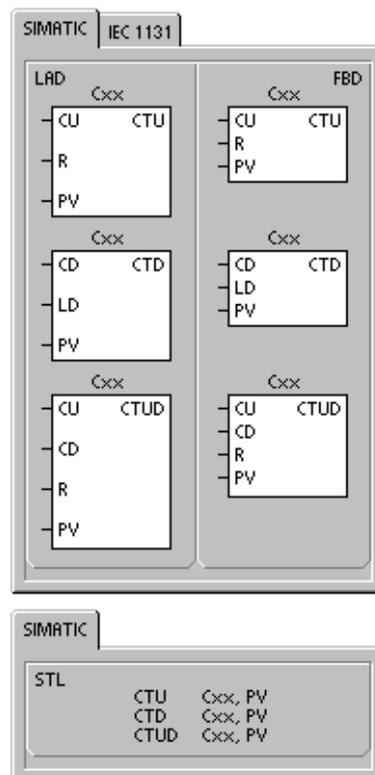
- Вход сброса: Вершина стека
- Вход прямого счета: Значение, загруженное на второй уровень стека

#### Обратный счет

Команда обратного счета (CTD) уменьшает текущее значение счетчика при появлении нарастающего фронта на входе (CD). Когда текущее значение Sxx равно нулю, бит счетчика Sxx включается. Счетчик сбрасывает свой бит Sxx и загружает текущее значение предустановленным значением PV, когда включается вход загрузки LD. Счетчик останавливается, когда он достигает нуля, и бит счетчика Sxx включается.

#### Принцип действия в STL:

- Вход загрузки: Вершина стека
- Вход обратного счета: Значение, загруженное на второй уровень стека.



## Реверсивный счетчик

Таблица 6–22. Допустимые операнды для счетчиков SIMATIC

Входы/выходы	Типы данных	Операнды
Sxx	WORD	константа (от C0 до C255)
CU, CD, LD, R	BOOL	I, Q, V, M, SM, S, T, C, L, поток сигнала
PV	INT	IW, QW, VW, MW, SMW, SW, LW, T, C, AC, AIW, *VD, *LD, *AC, константа



### Совет

Так как для каждого счетчика имеется только одно текущее значение, не назначайте один и тот же номер более чем одному счетчику (прямые, реверсивные и обратные счетчики с одним и тем же номером обращаются к одному и тому же текущему значению).

Если вы сбрасываете счетчик с помощью команды Сброс, то бит счетчика сбрасывается, а текущее значение устанавливается в ноль. Номер счетчика используется для обращения как к текущему значению, так и к биту счетчика.

Таблица 6–23. Принцип действия счетчиков

Тип	Режим	Бит счетчика	Выключение-включение питания/Первый цикл
CTU	CU увеличивает текущее значение. Текущее значение продолжает увеличиваться до достижения 32 767.	Бит счетчика устанавливается, когда: Текущее значение >= предустановленному	Бит счетчика сброшен. Текущее значение может быть сохранено. <sup>1</sup>
CTUD	CU увеличивает текущее значение. CD уменьшает текущее значение. Текущее значение продолжает увеличиваться или уменьшаться, пока счетчик не сброшен.	Бит счетчика устанавливается, когда: Текущее значение >= предустановленному	Бит счетчика сброшен. Текущее значение может быть сохранено. <sup>1</sup>
CTD	CD уменьшает текущее значение, пока текущее значение не достигнет 0.	Бит счетчика устанавливается, когда: Текущее значение = 0	Бит счетчика сброшен. Текущее значение может быть сохранено. <sup>1</sup>

<sup>1</sup> Вы можете определить текущее значение счетчика как сохраняемую величину. Информацию о сохраняемой памяти CPU S7–200 вы найдете в главе 4.

**Пример: Команда обратного счета SIMATIC**

<p><b>Network 1</b></p> <p><b>Network 2</b></p>	<p>Network 1 //Текущее значение счетчика обратного счета C1 //уменьшается с 3 до 0, когда I0.1 выключен. //Положительный фронт на I0.0 уменьшает текущее //значение C1 //Включение I0.1 загружает предустановленное //значение 3 для обратного счета</p> <p>LD I0.0 LD I0.1 CTD C1, +3</p> <p>Network 2 //Бит C1 установлен, когда текущее значение //счетчика C1 = 0</p> <p>LD C1 = Q0.0</p>
<p><b>Временная диаграмма</b></p>	

**Пример: Реверсивный счетчик SIMATIC**

<p><b>Network 1</b></p> <p><b>Network 2</b></p>	<p>Network 1 //I0.0 считает вперед //I0.1 считает назад //I0.2 сбрасывает текущее значение в 0</p> <p>LD I0.0 LD I0.1 LD I0.2 CTUD C48, +4</p> <p>Network 2 //Реверсивный счетчик C48 устанавливает //бит C48, когда текущее значение &gt;= 4</p> <p>LD C48 = Q0.0</p>
<p><b>Временная диаграмма</b></p>	

## Счетчик IEC

### Суммирующий счетчик

Суммирующий счетчик (CTU) ведет счет вверх от текущего значения к предустановленному значению (PV) при нарастающем фронте на входе прямого счета (CU). Когда текущее значение (CV) больше или равно предварительно установленному значению (PV), выходной бит счетчика (Q) включается. Счетчик сбрасывается, когда включается вход сброса (R). Суммирующий счетчик прекращает счет, когда он достигает предварительно установленного значения.

### Вычитающий счетчик

Вычитающий счетчик (CTD) ведет счет вниз от предустановленного значения (PV) при положительном фронте на входе обратного счета (CD). Когда текущее значение (CV) равно нулю, выходной бит счетчика (Q) включается. Счетчик сбрасывается и загружает в качестве текущего значения (CV) предустановленное значение (PV), когда включается вход загрузки (LD). Вычитающий счетчик прекращает счет, когда он достигает нуля.

### Реверсивный счетчик

Реверсивный счетчик (CTUD) ведет счет вверх или вниз от предустановленного значения (CV) при положительном фронте на входе прямого счета (CU) или обратного счета (CD). Когда текущее значение равно предварительно установленному значению, включается выход прямого счета (QU). Когда текущее значение равно нулю, включается выход обратного счета (QD). Счетчик загружает в качестве текущего значения (CV) предварительно установленное значение (PV), когда включается вход загрузки (LD). Аналогично, счетчик сбрасывается и загружает в качестве текущего значения (CV) нуль, когда включается сброс (R). Счетчик прекращает счет, когда он достигает предварительно установленного значения или нуля.

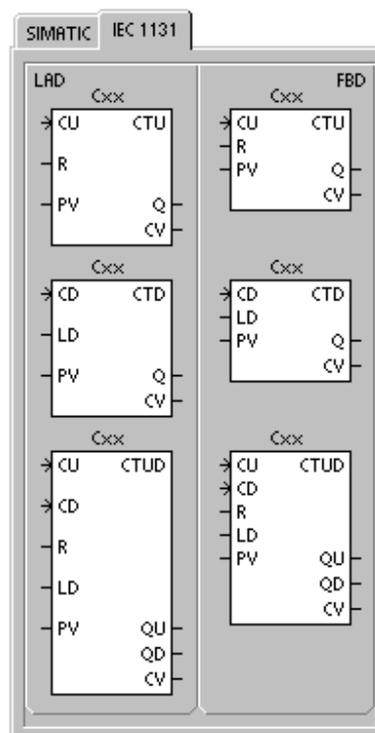


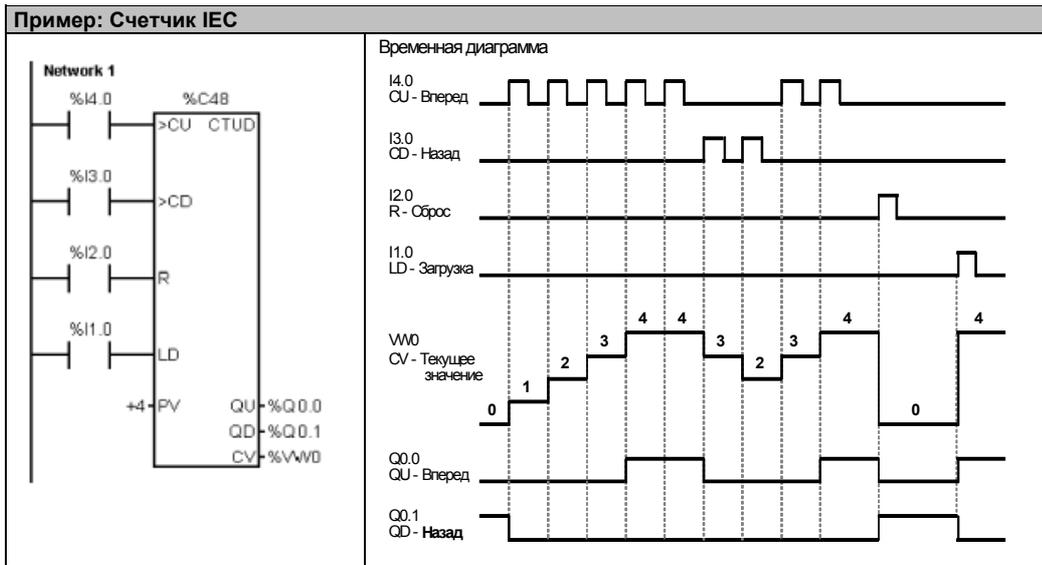
Таблица 6–24. Допустимые операнды для счетчиков IEC

Входы/выходы	Типы данных	Операнды
Cxx	CTU, CTD, CTUD	константа (от C0 до C255)
CU, CD, LD, R	BOOL	I, Q, V, M, SM, S, T, C, L, поток сигнала
PV	INT	IW, QW, VW, MW, SMW, SW, LW, AC, AIW, *VD, *LD, *AC, константа
Q, QU, QD	BOOL	I, Q, V, M, SM, S, L
CV	INT	IW, QW, VW, MW, SW, LW, AC, *VD, *LD, *AC



#### Совет

Так как для каждого счетчика имеется одно текущее значение, не назначайте один и тот же номер более чем одному счетчику. (Суммирующие, вычитающие и реверсивные счетчики обращаются к одному и тому же текущему значению.)



## Скоростные счетчики

### Определение режима работы скоростного счетчика

Команда определения режима работы скоростного счетчика (HDEF) устанавливает режим работы для определенного скоростного счетчика (HSCx). Выбором режима определяются датчик тактовых импульсов, направление и функции запуска и сброса скоростного счетчика.

Для каждого скоростного счетчика используется одна команда определения режима работы.

**Сбойные состояния, устанавливающие ENO = 0**

- 0003 (конфликт входов)
- 0004 (недопустимая команда в прерывании)
- 000A (повторное определение HSC)

### Скоростной счетчик

Команда активизации скоростного счетчика (HSC) конфигурирует и управляет режимом работы скоростного счетчика через сигнальные состояния битов специальной памяти HSC. Параметр N определяет номер скоростного счетчика.

Скоростные счетчики могут быть сконфигурированы на двенадцать различных режимов работы. См. таблицу 6–26.

Каждый счетчик имеет специализированные входы, которые поддерживают такие функции, как датчик тактовых импульсов, управление направлением, сброс и запуск. Для двухфазных счетчиков оба датчика тактовых импульсов могут работать со своей максимальной скоростью. В квадратурных режимах (A/B-счетчики) предоставляется возможность выбора однократной (1x) или четырехкратной (4x) скорости счета. Все счетчики работают с максимальной скоростью, не создавая помех друг другу.

**Сбойные состояния, устанавливающие ENO = 0**

- 0001 (HSC перед HDEF)
- 0005 (одновременно HSC/PLS)

Таблица 6–25. Допустимые операнды для скоростных счетчиков

Входы/выходы	Типы данных	Операнды
HSC, MODE	BYTE	константа
N	WORD	константа



Советы по программированию

За программами, использующими скоростные счетчики, обратитесь к советам по программированию на компакт-диске с документацией. См. советы 4 и 29.

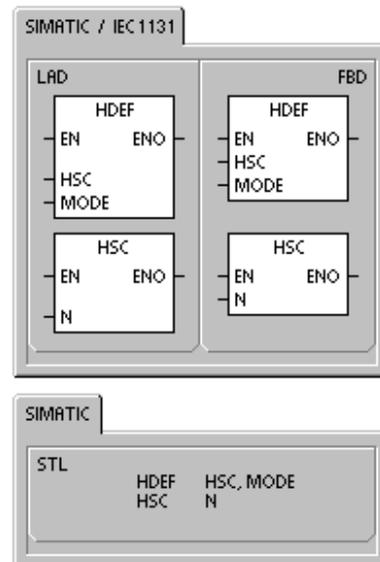
Скоростные счетчики ведут счет происходящих с высокой скоростью событий, которые не могут управляться при скоростях сканирования S7–200. Максимальная частота счета скоростного счетчика зависит от типа вашего CPU S7–200. Дальнейшую информацию вы найдете в Приложении А.



**Совет**

CPU 221 и CPU 222 поддерживают четыре скоростных счетчика: HSC0, HSC3, HSC4 и HSC5. Эти CPU не поддерживают HSC1 и HSC2.

CPU 224, CPU 224XP и CPU 226 поддерживают шесть скоростных счетчиков: от HSC0 до HSC5.



Обычно скоростные счетчики используются в качестве привода для счетных механизмов, в которых вал, вращающийся с постоянной скоростью, снабжен угловым шаговым датчиком. Угловой шаговый датчик дает определенное количество отсчетов на оборот, а также импульс сброса один раз за оборот. Датчик (датчики) тактовых импульсов и импульс сброса от углового шагового датчика обеспечивают входы для скоростного счетчика.

Скоростной счетчик загружается первым из нескольких предустановленных значений, и желаемые выходы активизируются на интервал времени, в течение которого текущее значение счетчика меньше текущего предустановленного значения. Счетчик настроен таким образом, что, когда текущее значение счетчика становится равным предустановленному значению, или при появлении сброса происходит прерывание.

Когда при равенстве текущего значения счетчика и предустановленного значения происходит прерывающее событие, загружается новое предустановленное значение, и устанавливается следующее состояние для выходов. Когда происходит событие, вызывающее прерывание по сбросу, то устанавливаются первое предустановленное значение и первые состояния выходов, и цикл повторяется.

Так как прерывания происходят со значительно меньшей частотой, чем считает скоростной счетчик, то может быть реализовано точное управление быстрыми операциями при относительно малом воздействии на общий цикл обработки программы ПЛК. Метод подключения прерываний позволяет выполнять каждую загрузку нового предустановленного значения в отдельной программе обработки прерывания, что упрощает управление состоянием. (В качестве альтернативы, все события, вызывающие прерывания, могут быть обработаны и в одной единственной программе обработки прерываний.)

### Описание различных скоростных счетчиков

Все счетчики в одном и том же режиме работают одинаково. Имеется четыре основных вида счетчиков: однофазный счетчик с внутренним управлением направлением, однофазный счетчик с внешним управлением направлением, двухфазный счетчик с 2 тактовыми входами и квадратурный счетчик с фазами А и В. Обратите внимание, что не каждый счетчик поддерживает все режимы. Каждый счетчик можно использовать: без входов сброса и пуска, со сбросом, но без пуска, или с входами пуска и сброса.

- Когда вы активизируете вход сброса, он сбрасывает текущее значение и сохраняет его сброшенным, пока вы не деактивизируете сброс.
- Когда вы активизируете вход пуска, он разрешает счетчику считать. Если вход пуска деактивизирован, текущее значение счетчика остается постоянным, а тактовые события игнорируются.
- Если сброс активизируется, когда пуск неактивен, то сброс игнорируется, а текущее значение не изменяется. Если вход пуска становится активным, когда активен вход сброса, текущее значение сбрасывается.

Перед использованием скоростного счетчика вы должны с помощью команды HDEF (High-Speed Counter Definition = Определение скоростного счетчика) выбрать его режим. С помощью бита памяти первого цикла SM0.1 (этот бит включен в течение первого цикла обработки программы, а затем выключается) вызовите подпрограмму, которая содержит команду HDEF.

## Программирование скоростного счетчика



Мастер команд

Для проектирования скоростного счетчика можно использовать мастер команд HSC. Этот мастер использует следующую информацию: тип и режим работы счетчика, предустановленное значение счетчика, текущее значение счетчика и начальное направление счета. Для вызова мастера команд HSC выберите команду меню **Tools > Instruction Wizard [Инструменты > Мастер команд]**, а затем из окна мастера команд выберите вариант HSC.

Для программирования скоростного счетчика вы должны выполнить следующие основные задачи:

- Определить счетчик и режим.
- Настроить управляющий байт.
- Установить текущее (начальное) значение.
- Задать предустановленное (целевое) значение.
- Назначить и разблокировать программу обработки прерываний.
- Активизировать скоростной счетчик.

### Определение режимов и входов счетчика

Для определения режимов и входов счетчика используется команда определения скоростного счетчика HDEF.

В таблице 6–26 показаны входы для таких функций скоростных счетчиков, как генератор тактовых импульсов, управление направлением, сброс и запуск. Один и тот же вход не может быть использован для двух разных функций, но любой вход, не используемый текущим режимом скоростного счетчика, может быть использован для другой цели. Например, если HSC0 используется в режиме 1, который использует I0.0 и I0.2, то I0.1 может быть использован для прерываний по фронту сигнала или для HSC3.

	<p><b>Совет</b></p> <p>Примите во внимание, что все режимы HSC0 (кроме режима 12) всегда используют I0.0, а все режимы HSC4 всегда используют I0.3, так что эти входы не бывают доступными для других целей, когда используются данные счетчики.</p>
---	--

Таблица 6–26. Входы для скоростных счетчиков

Режим	Описание	Входы			
	HSC0	I0.0	I0.1	I0.2	
	HSC1	I0.6	I0.7	I1.0	I1.1
	HSC2	I1.2	I1.3	I1.4	I1.5
	HSC3	I0.1			
	HSC4	I0.3	I0.4	I0.5	
	HSC5	I0.4			
0	Однофазный счетчик с внутренним управлением направлением	Датчик тактовых импульсов			
1		Датчик тактовых импульсов		Сброс	
2		Датчик тактовых импульсов		Сброс	Пуск
3	Однофазный счетчик с внешним управлением направлением	Датчик тактовых импульсов	Направление		
4		Датчик тактовых импульсов	Направление	Сброс	
5		Датчик тактовых импульсов	Направление	Сброс	Пуск
6	Двухфазный счетчик с 2 тактовыми входами	Датчик тактовых импульсов для прямого направления	Датчик тактовых импульсов для обратного направления		
7		Датчик тактовых импульсов для прямого направления	Датчик тактовых импульсов для обратного направления	Сброс	
8		Датчик тактовых импульсов для прямого направления	Датчик тактовых импульсов для обратного направления	Сброс	Пуск
9	Квадратурный счетчик с фазами А и В	Датчик тактовых импульсов А	Датчик тактовых импульсов В		
10		Датчик тактовых импульсов А	Датчик тактовых импульсов В	Сброс	
11		Датчик тактовых импульсов А	Датчик тактовых импульсов В	Сброс	Пуск
12	Режим счета 12 поддерживают только HSC0 и HSC3. HSC0 считает количество импульсов, выдаваемых Q0.0. HSC3 считает количество импульсов, выдаваемых Q0.1.				

### Примеры режимов HSC

Временные диаграммы на рисунках 6-22 - 6-26 показывают, как работает каждый счетчик в соответствии с режимом.

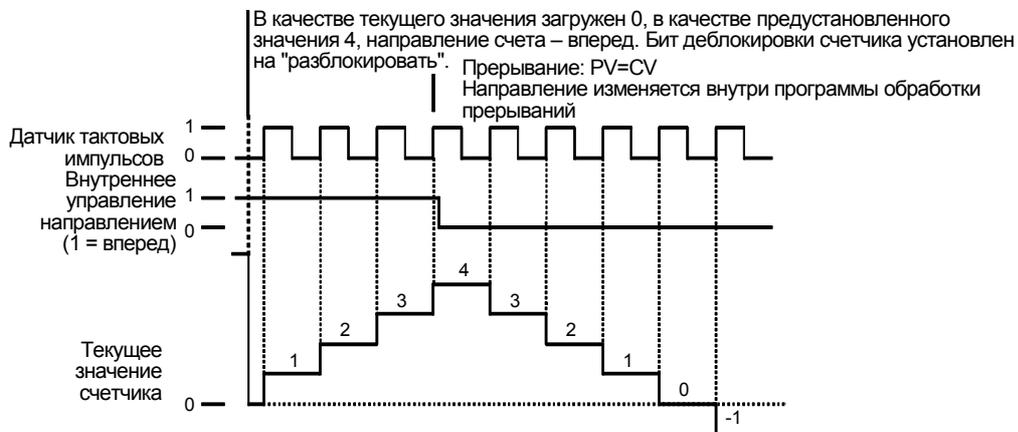


Рис. 6-22. Пример работы в режимах 0, 1 или 2



Рис. 6-23. Пример работы в режимах 3, 4 или 5

Когда используются режимы счета 6, 7 или 8, и в течение 0,3 микросекунды друг за другом появляется нарастающий фронт на тактовых входах счета вперед и счета назад, скоростной счетчик может рассматривать эти события как происходящие одновременно. Если это происходит, то текущее значение не меняется и не отображается изменение в направлении счета. Если между поступлениями нарастающих фронтов на тактовые входы счета вперед и счета назад проходит больше 0,3 микросекунды, то скоростной счетчик воспринимает эти события отдельно. В этом случае ошибки не происходит, и счетчик сохраняет правильное счетное значение.

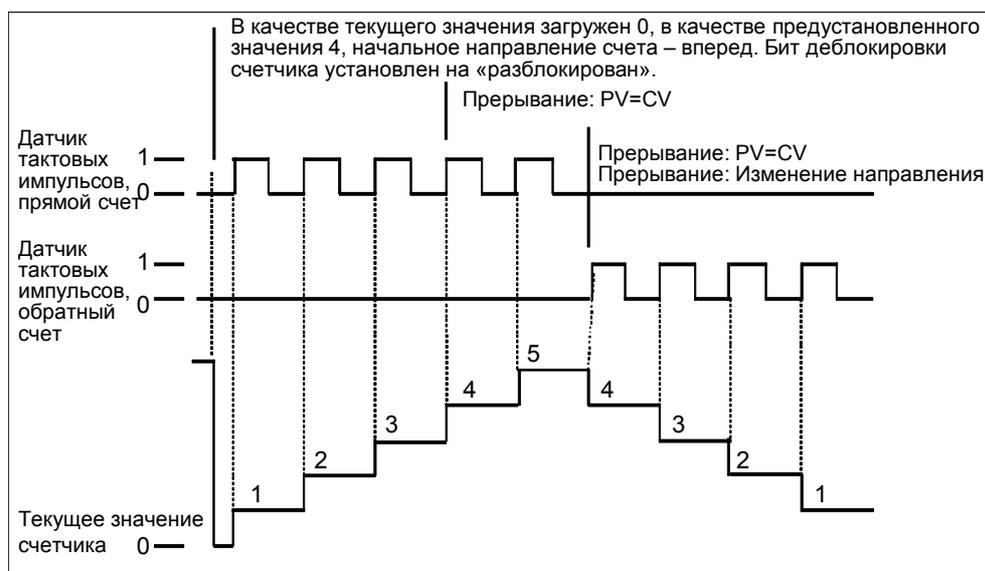


Рис. 6–24. Пример работы в режимах 6, 7 или 8

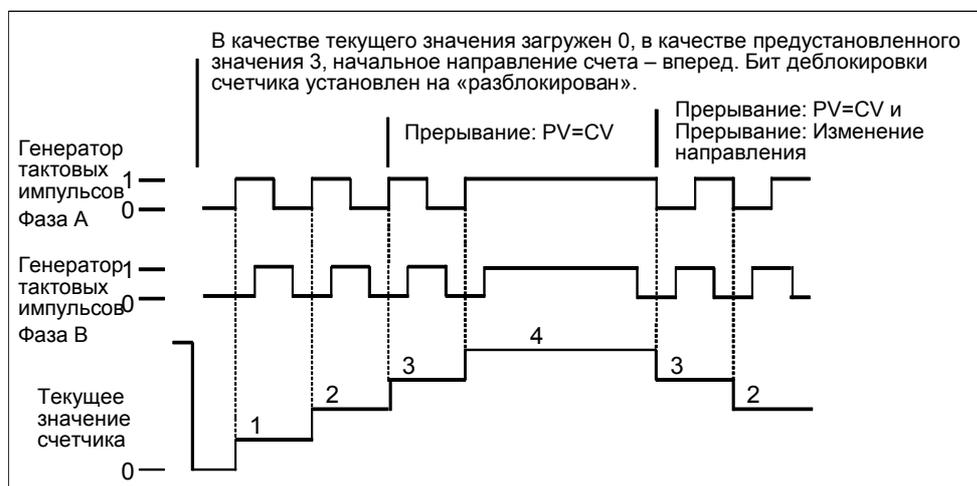


Рис. 6–25. Пример работы в режимах 9, 10 или 11 (квадратурный режим, однократная скорость)

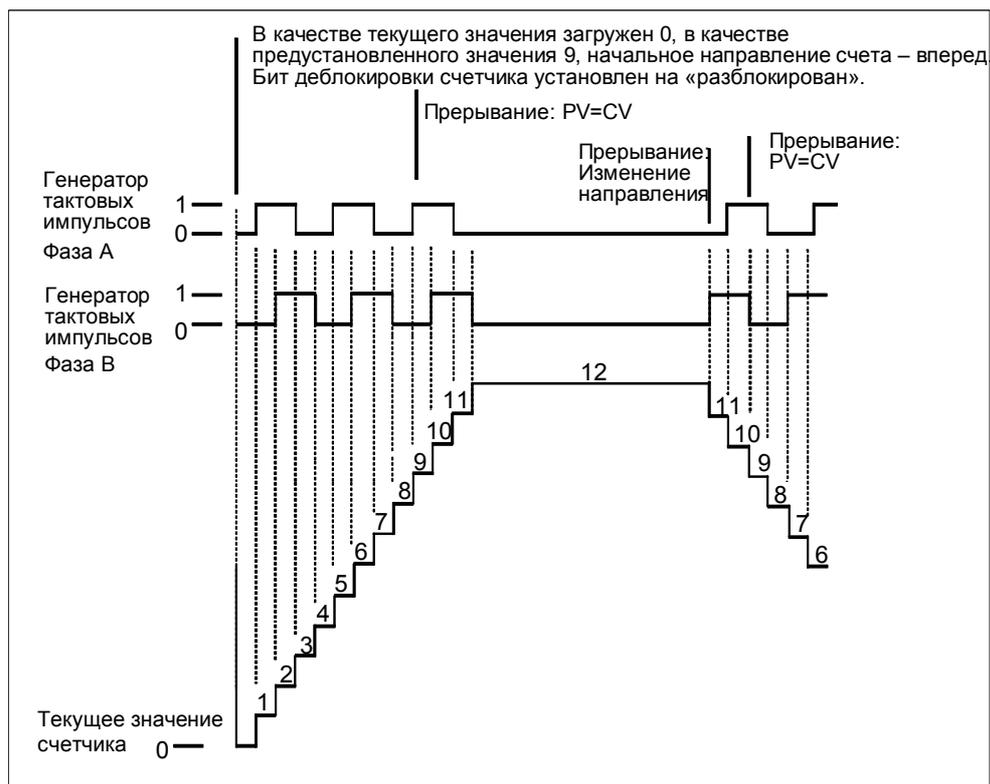


Рис. 6–26. Пример работы в режимах 9, 10 или 11 (квадратурный режим, четырехкратная скорость)

### Принцип действия входов сброса и пуска

Работа входов сброса и запуска показана на рис. 6–27 и действительна для всех режимов, использующих входы сброса и запуска. На диаграммах для входов сброса и запуска активность обоих входов запрограммирована для высокого уровня сигнала.

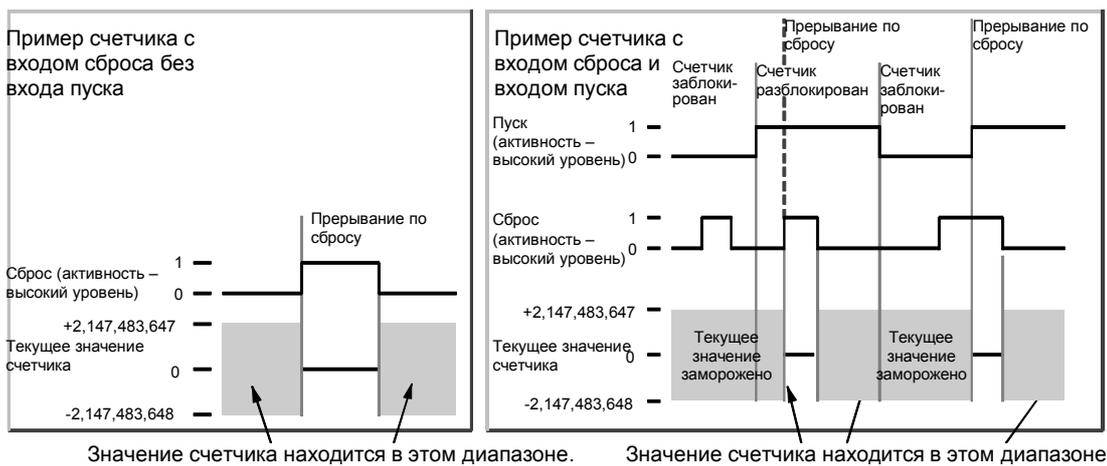


Рис. 6–27. Примеры работы счетчика, использующего вход сброса с входом и без входа пуска

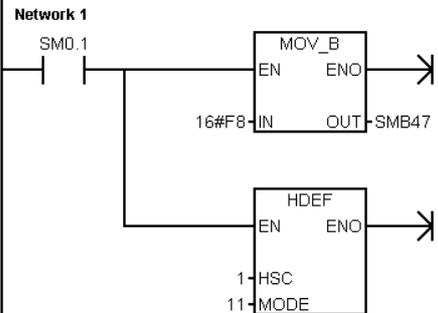
Четыре счетчика имеют три управляющих бита, которые используются для конфигурирования активного состояния входов сброса и пуска и для выбора односкоростного или четырехскоростного режима счета (только для квадратурных счетчиков). Эти биты находятся в управляющем байте соответствующего счетчика и используются только тогда, когда выполняется команда HDEF. Эти биты определены в таблице 6–27.

	<p><b>Совет</b></p> <p>Вы должны установить эти управляющие биты в соответствии с желаемым состоянием до исполнения команды HDEF. В противном случае счетчик принимает конфигурацию, определенную по умолчанию для выбранного режима работы счетчика.</p> <p>Если команда HDEF была выполнена, вы не можете изменить настройку счетчика, не переведя сначала S7–200 в состояние STOP.</p>
---	---

Таблица 6–27. Активный уровень для управляющих битов сброса, пуска и выбора однократной или четырехкратной скорости

HSC0	HSC1	HSC2	HSC4	Описание (используются только при исполнении HDEF)
SM37.0	SM47.0	SM57.0	SM147.0	Активный уровень управляющего бита для сброса <sup>1</sup> : 0 = сброс активен при высоком уровне; 1 = сброс активен при низком уровне
---	SM47.1	SM57.1	---	Активный уровень управляющего бита для пуска <sup>1</sup> : 0 = пуск активен при высоком уровне; 1 = пуск активен при низком уровне
SM37.2	SM47.2	SM57.2	SM147.2	Выбор скорости счета для квадратурных счетчиков: 0 = 4-кратная скорость      1 = 1-кратная скорость

<sup>1</sup> По умолчанию входы сброса и пуска активны при высоком уровне сигнала, а в квадратурных счетчиках скорость счета установлена четырехкратной (по отношению к частоте входного датчика тактовых импульсов).

Пример: Команда определения режима скоростного счетчика	
<p>M A I N</p>	

Network 1 //В первом цикле:  
//1. Выбрать для входов пуска и сброса в качестве активного высокого уровня и 4-кратную скорость счета.  
//2. Настроить HSC1 на квадратурный режим (A/B-счетчик) с входами сброса и пуска.

```
LD SM0.1
MOV_B 16#F8, SMB47
HDEF 1, 11
```

### Настройка управляющего бита

Определив счетчик и режим его работы, вы можете программировать динамические параметры счетчика. Каждый скоростной счетчик имеет управляющий байт, который позволяет выполнить следующие действия:

- разблокировать или заблокировать счетчик
- управлять направлением (только для режимов 0, 1 и 2) или устанавливать начальное направление счета для всех остальных режимов
- загружать текущее значение
- загружать предустановленное значение

Проверка управляющего байта и соответствующих текущего и предустановленного значений производится при выполнении команды HSC. В таблице 6–28 описан каждый из этих управляющих битов.

Таблица 6–28. Управляющие биты для HSC0, HSC1, HSC2, HSC3, HSC4 и HSC5

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	Описание
SM37.3	SM47.3	SM57.3	SM137.3	SM147.3	SM157.3	Бит управления направлением счета: 0 = обратный счет    1 = прямой счет
SM37.4	SM47.4	SM57.4	SM137.4	SM147.4	SM157.4	Записать направление счета в HSC: 0 = не актуализировать 1 = актуализировать направление
SM37.5	SM47.5	SM57.5	SM137.5	SM147.5	SM157.5	Записать новое предустановленное значение в HSC: 0 = не актуализировать; 1 = актуализировать предустановленное значение
SM37.6	SM47.6	SM57.6	SM137.6	SM147.6	SM157.6	Записать новое текущее значение в HSC: 0 = не актуализировать; 1 = актуализировать текущее значение
SM37.7	SM47.7	SM57.7	SM137.7	SM147.7	SM157.7	Разблокировка HSC: 0 = заблокировать HSC; 1 = разблокировать HSC

### Установка текущего и предустановленного значений

Каждый скоростной счетчик имеет 32–битное текущее значение и 32–битное предустановленное значение. Оба значения являются целыми числами со знаком. Чтобы загрузить новое текущее или предустановленное значение, вы должны настроить управляющий байт и байты специальной памяти, содержащие текущее и/или предустановленное значение, а также выполнить команду HSC, чтобы новые значения были переданы в скоростной счетчик. Таблица 6–29 описывает байты специальной памяти, используемые для хранения новых текущих и предустановленных значений.

В дополнение к управляющим байтам и байтам, содержащим новые текущие и предустановленные значения, текущее значение каждого скоростного счетчика может быть прочитано путем задания типа данных HC (текущее значение скоростного счетчика), за которым следует номер (0, 1, 2, 3, 4 или 5) счетчика, как показано в таблице 6–29. Текущее значение непосредственно доступно для операций чтения, но оно может быть записано только с помощью команды HSC.

Таблица 6–29. Новое текущее и новое предустановленное значение HSC0, HSC1, HSC2, HSC3, HSC4 и HSC5

Загружаемое значение	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5
Новое текущее значение	SMD38	SMD48	SMD58	SMD138	SMD148	SMD158
Новое предустановленное значение	SMD42	SMD52	SMD62	SMD142	SMD152	SMD162

Таблица 6–30. Текущие значения HSC0, HSC1, HSC2, HSC3, HSC4 и HSC5

Значение	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5
Текущее значение	HC0	HC1	HC2	HC3	HC4	HC5

### Адресация скоростных счетчиков (НС)

Для доступа к счетному значению скоростного счетчика указывается адрес этого счетчика с помощью типа памяти (НС) и номера счетчика (например, НС0). Текущее значение скоростного счетчика доступно только для чтения и может быть адресовано только как двойное слово (32 бита), как показано на рис. 6–28.



Рис. 6–28. Доступ к текущему значению скоростного счетчика

### Назначение прерываний

Все режимы счетчиков поддерживают прерывание по равенству текущего значения HSC заданному предустановленному значению. Режимы счетчиков, использующие вход внешнего сброса, поддерживают прерывание по активизации внешнего сброса. Все режимы счетчиков, кроме режимов 0, 1 и 2, поддерживают прерывание по изменению направления счета. Каждое из этих условий возникновения прерываний может быть заблокировано или разблокировано по отдельности. Полностью использование прерываний обсуждается в разделе о командах обмена данными и прерывания.

#### Замечание

Если вы попытаетесь загрузить новое текущее значение или заблокировать, а затем снова разблокировать скоростной счетчик из программы обработки прерываний для внешнего сброса, то может произойти фатальная ошибка.

### Байт состояния

Каждому скоростному счетчику поставлен в соответствие байт состояния, предоставляющий в распоряжение биты памяти, указывающие текущее направление счета, а также информацию о том, действительно ли текущее значение больше или равно предустановленному. Таблица 6–31 определяет эти биты состояния для каждого скоростного счетчика.



#### Совет

Биты состояния действительны только во время исполнения программы обработки прерывания скоростного счетчика. Цель контроля состояния скоростного счетчика состоит в том, чтобы разблокировать прерывания для событий, оказывающих воздействие на выполняемую операцию.

Таблица 6–31. Биты состояния для HSC0, HSC1, HSC2, HSC3, HSC4 и HSC5

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	Описание
SM36.0	SM46.0	SM56.0	SM136.0	SM146.0	SM156.0	Не используются
SM36.1	SM46.1	SM56.1	SM136.1	SM146.1	SM156.1	Не используются
SM36.2	SM46.2	SM56.2	SM136.2	SM146.2	SM156.2	Не используются
SM36.3	SM46.3	SM56.3	SM136.3	SM146.3	SM156.3	Не используются
SM36.4	SM46.4	SM56.4	SM136.4	SM146.4	SM156.4	Не используются
SM36.5	SM46.5	SM56.5	SM136.5	SM146.5	SM156.5	Бит состояния текущего направления счета: 0 = обратный счет 1 = прямой счет
SM36.6	SM46.6	SM56.6	SM136.6	SM146.6	SM156.6	Бит состояния, указывающий, равно ли текущее значение предустановленному: 0 = не равно 1 = равно
SM36.7	SM46.7	SM56.7	SM136.7	SM146.7	SM156.7	Бит состояния, указывающий, больше ли текущее значение, чем предустановленное: 0 = меньше или равно 1 = больше

## Примеры инициализирующих последовательностей для скоростных счетчиков

В следующих описаниях инициализации и последовательности обработки в качестве примера используется HSC1. При описании инициализаций предполагается, что S7-200 только что переведен в режим RUN, и поэтому бит памяти первого цикла установлен. Если это не так, помните, что команда HDEF может быть выполнена только один раз для каждого скоростного счетчика после вхождения в режим RUN. Выполнение HDEF для скоростного счетчика во второй раз приводит к ошибке выполнения и не изменяет настройку счетчика по сравнению с тем, как она была выполнена для данного счетчика при первом выполнении HDEF.



### Совет

Хотя приведенные далее последовательности показывают, как изменить направление, текущее и предустановленное значение по отдельности, вы можете изменить все эти настройки или любую их комбинацию в той же последовательности, устанавливая надлежащим образом SMB47, а затем выполняя команду HSC.

### Инициализация режимов 0, 1 и 2

Следующие шаги описывают, как инициализировать HSC1 в качестве однофазного реверсивного счетчика с внутренним управлением направлением счета (режим 0, 1 или 2).

1. Используйте бит памяти первого цикла для вызова подпрограммы, в которой будет выполняться операция по инициализации. Когда вы используете вызов подпрограммы, следующие циклы эту подпрограмму не вызывают, что сокращает время цикла и делает программу более структурированной.
2. В подпрограмме инициализации загрузите SMB47 в соответствии с желаемой операцией управления. Например:
 

SMB47 = 16#F8	<i>дает следующие результаты:</i>
	Разблокирует счетчик
	Записывает новое текущее значение
	Записывает новое предустановленное значение
	Устанавливает прямое направление счета
	Настраивает входы пуска и сброса на активность при высоком уровне сигнала
3. Выполните команду HDEF с входом HSC, установленным в 1, и входом MODE [режим], установленным в 0 при отсутствии внешнего сброса и пуска, 1 для внешнего сброса без пуска или 2 для внешнего сброса и пуска.
4. Загрузите SMD48 (двойное слово) желаемым текущим значением (загрузите 0, чтобы его очистить).
5. Загрузите SMD52 (двойное слово) желаемым предустановленным значением.
6. Чтобы распознавать равенство текущего и предустановленного значений, запрограммируйте прерывание, поставив в соответствии с программой обработки прерывания прерывающее событие CV = PV (событие 13). Подробную информацию об обработке прерываний вы найдете в разделе, посвященном командам прерывания.
7. Чтобы распознавать внешний сброс, запрограммируйте прерывание, поставив в соответствии с программой обработки прерывания прерывающее событие «внешний сброс» (external reset) (событие 15).
8. Для разблокировки прерываний выполните команду разрешения всех прерываний (ENI).
9. Выполните команду HSC, чтобы S7-200 запрограммировал HSC1.
10. Выйдите из подпрограммы.

### Инициализация режимов 3, 4 и 5

Следующие шаги описывают, как инициализировать HSC1 в качестве однофазного реверсивного счетчика с внешним управлением направлением счета (режим 3, 4 или 5):

1. Используйте бит памяти первого цикла для вызова подпрограммы, в которой будет выполняться операция по инициализации. Когда вы используете вызов подпрограммы, следующие циклы эту подпрограмму не вызывают, что сокращает время цикла и делает программу более структурированной.
2. В подпрограмме инициализации загрузите SMB47 в соответствии с желаемой операцией управления. Например:  
 SMB47 = 16#F8 *дает следующие результаты:*  
 Разблокирует счетчик  
 Записывает новое текущее значение  
 Записывает новое предустановленное значение  
 Устанавливает начальное направление счета HSC на прямой счет  
 Настраивает входы пуска и сброса на активность при высоком уровне сигнала
3. Выполните команду HDEF с входом HSC, установленным в 1, и входом MODE [режим], установленным одним из следующих способов: 3 при отсутствии внешнего сброса и пуска, 4 для внешнего сброса без пуска или 5 для внешнего сброса и пуска.
4. Загрузите SMD48 (двойное слово) желаемым текущим значением (загрузите 0, чтобы его очистить).
5. Загрузите SMD52 (двойное слово) желаемым предустановленным значением.
6. Чтобы распознать равенство текущего и предустановленного значений, запрограммируйте прерывание, поставив в соответствии с программой обработки прерывания прерывающее событие CV = PV (событие 13). Подробную информацию об обработке прерываний вы найдете в разделе, посвященном командам прерывания.
7. Чтобы распознавать изменения направления счета, запрограммируйте прерывание, поставив в соответствии с программой обработки прерывания прерывающее событие «изменение направления» (direction changed) (событие 14).
8. Чтобы распознавать внешний сброс, запрограммируйте прерывание, поставив в соответствии с программой обработки прерывания прерывающее событие «внешний сброс» (external reset) (событие 15).
9. Для разблокировки прерываний выполните команду разрешения всех прерываний (ENI).
10. Выполните команду HSC, чтобы S7-200 запрограммировал HSC1.
11. Выйдите из подпрограммы.

### Инициализация режимов 6, 7 и 8

Следующие шаги описывают, как инициализировать HSC1 в качестве двухфазного реверсивного счетчика с датчиками тактовых импульсов прямого и обратного направления счета (режим 6, 7 или 8):

1. Используйте бит памяти первого цикла для вызова подпрограммы, в которой будет выполняться операция по инициализации. Когда вы используете вызов подпрограммы, следующие циклы эту подпрограмму не вызывают, что сокращает время цикла и делает программу более структурированной.
2. В подпрограмме инициализации загрузите SMB47 в соответствии с желаемой операцией управления. Например:  
 SMB47 = 16#F8 *дает следующие результаты:*  
 Разблокирует счетчик  
 Записывает новое текущее значение  
 Записывает новое предустановленное значение  
 Устанавливает начальное направление счета HSC на прямой счет  
 Настраивает входы пуска и сброса на активность при высоком уровне сигнала
3. Выполните команду HDEF с входом HSC, установленным в 1, и входом MODE [режим], установленным в одно из следующих состояний: 6 при отсутствии внешнего сброса и пуска, 7 для внешнего сброса без пуска или 8 для внешнего сброса и пуска.
4. Загрузите SMD48 (двойное слово) желаемым текущим значением (загрузите 0, чтобы его очистить).

5. Загрузите SMD52 (двойное слово) желаемым предустановленным значением.
6. Чтобы распознать равенство текущего и предустановленного значений, запрограммируйте прерывание, поставив в соответствие программе обработки прерывания прерывающее событие CV = PV (событие 13). См. раздел о прерываниях.
7. Чтобы распознавать изменения направления счета, запрограммируйте прерывание, поставив в соответствие программе обработки прерывания прерывающее событие «изменение направления» (direction changed) (событие 14).
8. Чтобы распознавать внешний сброс, запрограммируйте прерывание, поставив в соответствие программе обработки прерывания прерывающее событие «внешний сброс» (external reset) (событие 15).
9. Для разблокировки прерываний выполните команду разрешения всех прерываний (ENI).
10. Выполните команду HSC, чтобы S7–200 запрограммировал HSC1.
11. Выйдите из подпрограммы.

### Инициализация режимов 9, 10 и 11

Следующие шаги описывают, как инициализировать HSC1 в качестве квадратурного счетчика с фазами A и B (режим 9, 10 или 11):

1. Используйте бит памяти первого цикла для вызова подпрограммы, в которой будет выполняться операция по инициализации. Когда вы используете вызов подпрограммы, следующие циклы эту подпрограмму не вызывают, что сокращает время цикла и делает программу более структурированной.
2. В подпрограмме инициализации загрузите SMB47 в соответствии с желаемой операцией управления.

Пример (однократная скорость счета):

SMB47 = 16#FC *даёт следующие результаты:*

Разблокирует счетчик  
 Записывает новое текущее значение  
 Записывает новое предустановленное значение  
 Устанавливает начальное направление счета HSC на прямой счет  
 Настраивает входы пуска и сброса на активность при высоком уровне сигнала

Пример (четырёхкратная скорость счета):

SMB47 = 16#F8 *даёт следующие результаты:*

Разблокирует счетчик  
 Записывает новое текущее значение  
 Записывает новое предустановленное значение  
 Устанавливает начальное направление счета HSC на прямой счет  
 Настраивает входы пуска и сброса на активность при высоком уровне сигнала

3. Выполните команду HDEF с входом HSC, установленным в 1, и входом MODE [режим], установленным одним из следующих способов: 9 при отсутствии внешнего сброса и пуска, 10 для внешнего сброса без пуска или 11 для внешнего сброса и пуска.
4. Загрузите SMD48 (двойное слово) желаемым текущим значением (загрузите 0, чтобы его очистить).
5. Загрузите SMD52 (двойное слово) желаемым предустановленным значением.
6. Чтобы распознать равенство текущего и предустановленного значений, запрограммируйте прерывание, поставив в соответствие программе обработки прерывания прерывающее событие CV = PV (событие 13). Подробную информацию о разблокировании прерываний (ENI) вы найдете в разделе о командах прерывания.
7. Чтобы распознавать изменения направления счета, запрограммируйте прерывание, поставив в соответствие программе обработки прерывания прерывающее событие «изменение направления» (direction changed) (событие 14).
8. Чтобы распознавать внешний сброс, запрограммируйте прерывание, поставив в соответствие программе обработки прерывания прерывающее событие «внешний сброс» (external reset) (событие 15).
9. Для разблокировки прерываний выполните команду разрешения всех прерываний (ENI).
10. Выполните команду HSC, чтобы S7–200 запрограммировал HSC1.
11. Выйдите из подпрограммы.

## Инициализация режима 12

Следующие шаги описывают, как инициализировать HSC0 для счета импульсов, сгенерированных PTO0 (режим 12).

1. Используйте бит памяти первого цикла для вызова подпрограммы, в которой будет выполняться операция по инициализации. Когда вы используете вызов подпрограммы, следующие циклы эту подпрограмму не вызывают, что сокращает время цикла и делает программу более структурированной.
2. В подпрограмме инициализации загрузите SMB47 в соответствии с желаемой операцией управления. Например:
 

SMB37 = 16#F8	<i>дает следующие результаты:</i>
	Разблокирует счетчик
	Записывает новое текущее значение
	Записывает новое предустановленное значение
	Устанавливает направление счета на прямой            счет
	Настраивает входы пуска и сброса на активность при высоком уровне сигнала
3. Выполните команду HDEF с входом HSC, установленным в 0, и входом MODE [режим], установленным в 12.
4. Загрузите SMD38 (двойное слово) желаемым текущим значением (загрузите 0, чтобы его очистить).
5. Загрузите SMD42 (двойное слово) желаемым предустановленным значением.
6. Чтобы распознать равенство текущего и предустановленного значений, запрограммируйте прерывание, поставив в соответствии программе обработки прерывания прерывающее событие CV = PV (событие 13). Подробную информацию об обработке прерываний вы найдете в разделе, посвященном командам прерывания.
7. Для разблокировки прерываний выполните команду разрешения всех прерываний (ENI).
8. Выполните команду HSC, чтобы S7-200 запрограммировал HSC1.
9. Выйдите из подпрограммы.

## Изменение направления в режиме 0, 1, 2 или 12

Следующие шаги описывают, как сконфигурировать изменение направления для HSC1 как однофазного счетчика с внутренним управлением направлением счета (режим 0, 1, 2 или 12):

1. Загрузите SMB47, чтобы записать желаемое направление:
 

SMB47 = 16#90	Разблокирует счетчик
	Устанавливает направление счета HSC на обратный счет
SMB47 = 16#98	Разблокирует счетчик
	Устанавливает направление счета HSC на прямой счет
2. Выполните команду HSC, чтобы S7-200 запрограммировал HSC1.



Пример: Скоростной счетчик		
M A I N	<p><b>Network 1</b></p>	<p>Network 1 //В первом цикле вызвать SBR_0.</p> <pre>LD SMD.1 CALL SBR_0</pre>
S B R  0	<p><b>Network 1</b></p>	<p>Network 1 // В первом цикле настроить HSC1:</p> <p>//1. Разблокировать счетчик.  // - Записать новое текущее значение.  // - Записать новое предустановленное значение.  // - Установить в качестве начального направления счета прямой счет.  // - Сделать входы пуска и сброса активными при высоком уровне сигнала.  //2. Настроить HSC1 на квадратурный режим с входами сброса и пуска.  //3. Сбросить текущее значение HSC1.  //4. Ввести предустановленное значение HSC1, равное 50.  //5. Программе обработки прерываний INT_0 поставить в соответствие событие 13 (текущее значение HSC1 = предустановленному)  //6. Разблокировать все прерывания.  //7. Запрограммировать HSC1.</p> <pre>LD SMD.1 MOVB 16#F8, SMB47 HDEF 1, 11 MOVD +0, SMD48 MOVD +50, SMD52 ATCH INT_0, 13 ENI HSC 1</pre>
I N T  0	<p><b>Network 1</b></p>	<p>Network 1 // Запрограммировать HSC1:</p> <p>//1. Сбросить текущее значение HSC1.  //2. Только записать новое текущее значение и оставить HSC1 активным.</p> <pre>LD SMD.0 MOVD +0, SMD48 MOVB 16#C0, SMB47 HSC 1</pre>

## Команда вывода импульсов



Управление позиционированием

Команда вывода импульсов (PLS) используется для управления функциями "Вывод последовательности импульсов" (PTO) и "Широтно-импульсная модуляция" (PWM), имеющимися в распоряжении скоростных выходов (Q0.0 и Q0.1).

Усовершенствованный Мастер управления позиционированием создает команды, настроенные на ваше приложение, что упрощает программирование ваших задач и позволяет использовать преимущества дополнительных функций CPU S7-200. Дополнительную информацию о мастере управления позиционированием вы найдете в главе 9.

Для создания своего собственного приложения для управления перемещением вы можете по-прежнему использовать старую команду PLS, однако линейное нарастание сигнала в PTO поддерживается только командами, создаваемыми усовершенствованным мастером управления позиционированием.

PTO предоставляет в распоряжение выходной сигнал в виде прямоугольных импульсов (с относительной длительностью импульсов 50%), период следования которых и количество определяется пользователем.

PWM предоставляет в распоряжение выходной импульсный сигнал с постоянным периодом следования и переменной относительной длительностью импульсов, причем период следования импульсов и их ширину определяет пользователь.

S7-200 имеет два генератора PTO/PWM, которые формируют или быструю последовательность импульсов, или широтно-модулированный импульсный сигнал. Один генератор поставлен в соответствие цифровому выходу Q0.0, другой генератор – цифровому выходу Q0.1. Определенный адрес специальной памяти (SM) хранит для каждого генератора следующие данные: управляющий байт (8-битовое значение), количество импульсов (32-битовое значение без знака), а также период следования импульсов и значение ширины импульса (16-битовое значение без знака).

Генераторы PTO и PWM и регистр образа процесса совместно используют Q0.0 и Q0.1. Когда функция PTO или PWM активна на Q0.0 или Q0.1, то выходом управляет генератор PTO или PWM, а нормальное использование выхода заблокировано. На форму выходного сигнала не влияет ни состояние регистра образа процесса, ни принудительное присваивание значений выходам, ни выполнение команд непосредственного вывода. Когда генератор PTO/PWM не активен, управление выходом возвращается регистру образа процесса. Регистр образа процесса определяет начальное и конечное состояние импульсного выходного сигнала, вызывая его начало и завершение на высоком или низком уровне.

Таблица 6-32. Допустимые операнды для команды вывода импульсов

Входы/выходы	Типы данных	Операнды
Q0.X	WORD	константа: 0 (= Q0.0) или 1 (= Q0.1)



### Совет

Перед активизацией функции PTO или PWM установите значение регистра образа процесса для Q0.0 и Q0.1 в 0.

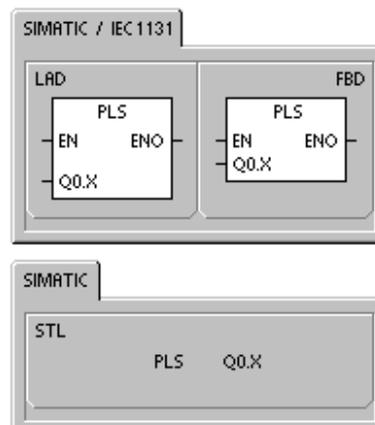
По умолчанию значения всех управляющих битов, периода следования импульсов, ширины и количества импульсов равны нулю.

**Выходы PTO/PWM должны иметь минимальную нагрузку не менее 10% от номинальной нагрузки, чтобы обеспечить четкий переход от выключенного состояния к включенному и наоборот.**



Советы по программированию

За программами, использующими команду PLS для функций PTO/PWM, обратитесь к советам по программированию на компакт-диске с документацией. См. советы 7, 22, 23, 30 и 50.



## Последовательность импульсов (PTO)

Функция PTO обеспечивает генерирование последовательности импульсов прямоугольной формы (с относительной длительностью 50%) с заданным количеством импульсов и заданным периодом следования импульсов. (См. рис. 6–29.) PTO может генерировать отдельную последовательность импульсов или несколько последовательностей импульсов (используя профиль импульсов). Вы задаете количество импульсов и период следования импульсов (в микро- или миллисекундах):

- Количество импульсов: от 1 до 4 294 967 295
- Период следования импульсов:
  - от 10 мкс до 65 535 мкс
  - или
  - от 2 мс до 65 535 мс.

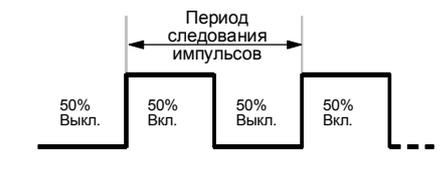


Рис. 6–29. Вывод последовательности импульсов (PTO)

Если при задании периода следования импульсов указывается нечетное число микросекунд или миллисекунд (напр., 75 мс), то это приводит к некоторому искажению относительной длительности импульсов.

Ограничения по количеству импульсов и периода следования импульсов приведены в таблице 6–33.

Таблица 6–33. Количество импульсов и период следования импульсов в функции PTO

Количество импульсов/Период следования импульсов	Реакция
Период следования импульсов < 2 единиц времени	Период следования импульсов устанавливается по умолчанию равным 2 единицам времени.
Количество импульсов = 0	Количество импульсов устанавливается по умолчанию равным 1 импульсу.

Функция PTO допускает сцепление, или конвейерную обработку последовательностей импульсов. Когда активная последовательность импульсов завершена, немедленно начинается вывод новой последовательности импульсов. Это обеспечивает непрерывность следующих друг за другом последовательностей импульсов.

### Использование мастера управления позиционированием

Мастер управления позиционированием автоматически обрабатывает односегментные и многосегментные конвейерные последовательности импульсов PTO, широтно-импульсную модуляцию, конфигурацию адресов SM и создание таблицы профилей. Эта информация приведена здесь для вашего сведения. Мы рекомендуем вам использовать мастер управления позиционированием. Дополнительную информацию о мастере управления позиционированием вы найдете в главе 9.

### Односегментная конвейерная обработка импульсов PTO

При односегментной конвейерной обработке вы несете ответственность за обновление ячеек области SM для следующей последовательности импульсов. Как только был запущен первый сегмент PTO, вы должны немедленно изменить ячейки SM в соответствии с требованиями второй последовательности и снова выполнить команду PLS. Атрибуты второй последовательности импульсов будут храниться в конвейере до завершения первой последовательности импульсов. В конвейере в каждый момент времени может храниться только одна запись. Как только завершится первая последовательность импульсов, начнется вывод второй последовательности, и конвейер становится доступным для задания характеристик новой последовательности импульсов. Вы можете затем повторить этот процесс, чтобы установить характеристики следующей последовательности импульсов.

Между последовательностями импульсов происходит плавный переход, если не меняется база времени и если активная последовательность импульсов не завершается раньше, чем настройка новой последовательности импульсов распознается при исполнении команды PLS.

### Многосегментная конвейерная обработка импульсов РТО

При многосегментной конвейерной обработке S7-200 автоматически считывает характеристики каждого сегмента последовательности импульсов из таблицы профиля, расположенной в V-памяти. Используемыми в этом режиме адресами специальной памяти являются управляющий байт, байт состояния и начальное смещение в V-памяти таблицы профиля (SMW168 или SMW178). В качестве базы времени могут быть заданы микросекунды или миллисекунды, но этот выбор применяется ко всем значениям периода следования импульсов в таблице профиля и не может быть изменен, когда профиль активен. Многосегментный режим работы запускается исполнением команды PLS.

Запись для каждого сегмента имеет длину 8 байтов и состоит из 16-битового значения периода следования импульсов, 16-битового значения приращения периода и 32-битового значения количества импульсов. Формат таблицы профиля показан в 6-34. Вы можете автоматически увеличивать или уменьшать длительность периода следования импульсов, программируя для каждого импульса определенное значение. Положительное значение в поле приращения периода следования импульсов увеличивает период, отрицательное значение уменьшает период, а нулевое значение не меняет периода следования импульсов.

При обработке профиля РТО номер активного в данный момент сегмента находится в SMB166 (или SMB176).

Таблица 6-34. Формат таблицы профиля для многосегментных последовательностей импульсов

Смещение в байтах	Сегмент	Описание записей таблицы
0		Количество сегментов: от 1 до 255 <sup>1</sup>
1	#1	Начальный период следования импульсов (от 2 до 65 535 единиц базы времени)
3		Приращение периода следования импульсов на импульс (величина со знаком) (от -32768 до 32767 единиц базы времени)
5		Количество импульсов (от 1 до 4 294 967 295)
9	#2	Начальный период следования импульсов (от 2 до 65 535 единиц базы времени)
11		Приращение периода следования импульсов на импульс (величина со знаком) (от -32768 до 32767 единиц базы времени)
13		Количество импульсов (от 1 до 4 294 967 295)
(продолжение)	#3	(продолжение)

<sup>1</sup> Ввод значения 0 для количества сегментов генерирует нефатальную ошибку. Вывод РТО не производится.

### Широтно-импульсная модуляция (PWM)

Функция PWM обеспечивает фиксированный период следования импульсов при переменной относительной длительности импульсов. (См. рис. 6-30.) Период следования и ширину импульсов можно задавать в микро- или миллисекундах:

- Период следования импульсов:  
от 10 мкс до 65 535 мкс  
или  
от 2 мс до 65 535 мс
- Ширина импульса: от 0 мкс до 65 535 мкс или  
от 0 мс до 65 535 мс



Рис. 6-30. Широтно-импульсная модуляция (PWM)

Как показано в таблице 6-35, при установке ширины импульса равной периоду следования импульсов (что делает относительную длительность импульсов равной 100 процентам) выход включен постоянно. При установке ширины импульса равной.

Таблица 6-35. Ширина и период следования импульсов и реакции функции PWM

Ширина импульса/период следования импульсов	Реакция
Ширина импульса $\geq$ периоду следования импульсов	Относительная длительность импульса равна 100%: выход включен постоянно.
Ширина импульса = 0	Относительная длительность импульса равна 0%: выход выключен.
Период следования импульсов $<$ 2 единиц времени	Период следования импульсов устанавливается по умолчанию равным двум единицам времени.

Имеется два способа изменения характеристик импульсов PWM:

- Синхронное обновление: Если не требуется изменения базы времени, то может быть выполнено синхронное обновление. При синхронном обновлении изменение характеристик импульсов происходит на границе периода следования импульсов, обеспечивая плавный переход.
- Асинхронное обновление: Обычно при работе PWM меняется ширина импульсов при постоянном периоде следования, так что изменение базы времени не требуется. Однако, если требуется изменение базы времени генератора PTO/PWM, то используется асинхронное обновление. Асинхронное обновление вызывает на мгновение блокирование генератора PTO/PWM асинхронно со следованием импульсов PWM. Это может вызвать нежелательную неустойчивость работы управляемого устройства. Поэтому рекомендуется синхронное обновление PWM. Выбирайте базу времени, которая будет работать со всеми ожидаемыми вами значениями периода следования импульсов.

	<p><b>Совет</b></p> <p>Бит метода обновления PWM (SM67.4 или SM77.4) в управляющем байте определяет тип обновления, используемый при исполнении команды PLS для реализации изменений. Если изменяется база времени, то произойдет асинхронное обновление независимо от состояния бита метода обновления PWM.</p>
---	--

### Конфигурирование и управление функциями PTO/PWM с помощью битов специальной памяти

Команда PLS считывает данные, хранящиеся в указанных ячейках специальной памяти (SM), и соответствующим образом программирует генератор PTO/PWM. SMB67 управляет PTO 0 или PWM 0, а SMB77 управляет PTO 1 или PWM 1. Таблица 6–36 описывает регистры, используемые для управления функционированием PTO/PWM. Вы можете использовать таблицу 6–37 в качестве быстрой справки, чтобы определить значение, которое следует поместить в управляющий регистр PTO/PWM, чтобы вызвать желаемую операцию.

Вы можете изменить характеристики сигнала PTO или PWM, изменяя биты памяти в области SM (включая управляющий байт), а затем исполняя команду PLS. Вы можете в любое время заблокировать генерирование импульсов PTO или PWM, записав 0 в бит деблокировки PTO/PWM управляющего бита (SM67.7 или SM77.7), а затем исполняя команду PLS.

Бит бездействия PTO в байте состояния (SM66.7 или SM76.7) предназначен для указания на завершение запрограммированной последовательности импульсов. Кроме того, по окончании последовательности импульсов может быть вызвана программа обработки прерываний. (По этому вопросу читайте описания команд прерывания и обмена данными.) Если вы используете многосегментный режим, то команда обработки прерываний вызывается после завершения таблицы профиля.

Следующие условия устанавливают SM66.4 (или SM76.4) и SM66.5 (или SM76.5):

- Если вы задаете приращение периода следования импульсов, которое через некоторое количество импульсов приведет к недопустимой длительности периода, то генерируется условие арифметического переполнения, которое завершает функцию PTO и устанавливает бит ошибки вычисления приращения (SM66.4 или SM76.4) в 1. Выход снова управляется через регистр образа процесса.
- Если вы обрабатываемый профиль PTO завершаете (деактивизируете) вручную, то в 1 устанавливается бит завершения по инициативе пользователя (SM66.5 или SM76.5).
- Если вы пытаетесь загрузить конвейерную обработку, когда конвейер полон, то в 1 устанавливается бит переполнения PTO (SM66.6 или SM76.6). Если вы хотите обнаруживать последующие переполнения, то при обнаружении переполнения вы должны сбросить этот бит вручную. Этот бит устанавливается в 0 при переходе в режим RUN.

	<p><b>Совет</b></p> <p>Если вы собираетесь загрузить новое количество импульсов (SMD72 или SMD82), ширину импульсов (SMW70 или SMW80) или период следования импульсов (SMW68 или SMW78), то установите также соответствующие биты обновления в управляющем регистре перед выполнением команды PLS. Если вы используете многосегментную последовательность импульсов, то перед выполнением команды PLS вы также должны загрузить начальное смещение (SMW168 или SMW178) таблицы профиля и значения таблицы профиля.</p>
---	--

Таблица 6–36. Биты специальной памяти для управляющих регистров PTO / PWM

Q0.0	Q0.1	Биты состояния
SM66.4	SM76.4	Профиль PTO прерван (ошибка расчета приращения): 0 = нет ошибки                      1 = прерван
SM66.5	SM76.5	Профиль PTO прерван по команде пользователя: 0 = нет завершения                1 = прерван
SM66.6	SM76.6	Переполнение/потеря значимости в конвейере PTO: 0 = нет переполнения            1 = переполнение/потеря значимости
SM66.7	SM76.7	PTO не действует: 0 = действует    1 = PTO не действует
Q0.0	Q0.1	Управляющие биты
SM67.0	SM77.0	PTO/PWM: обновить период следования импульсов: 0 = не обновлять                1 = обновить период следования импульсов
SM67.1	SM77.1	PWM: обновить значение ширины импульсов: 0 = не обновлять                1 = обновить ширину импульсов
SM67.2	SM77.2	PTO: обновить количество импульсов: 0 = не обновлять                1 = обновить количество импульсов
SM67.3	SM77.3	PTO/PWM: выбрать базу времени:    0 = 1 мкс/такт            1 = 1 мс/такт
SM67.4	SM77.4	PWM: метод обновления: 0 = асинхронное обновление    1 = синхронное обновление
SM67.5	SM77.5	PTO: односегментный/многосегментный режим: 0 = односегментный            1 = многосегментный
SM67.6	SM77.6	Выбор PTO или PWM:                0 = PTO            1 = PWM
SM67.7	SM77.7	Активизация PTO/PWM:    0 = заблокировать    1 = активизировать
Q0.0	Q0.1	Другие регистры PTO/PWM
SMW68	SMW78	Значение периода следования импульсов PTO/PWM    диапазон: от 2 до 65535
SMW70	SMW80	Значение ширины импульса PWM                            диапазон: от 0 до 65535
SMD72	SMD82	Значение количества импульсов PTO                    диапазон: от 1 до 4 294 967 295
SMB166	SMB176	Номер активного сегмента PTO                            только в многосегментном режиме
SMW168	SMW178	Начальный адрес таблицы профиля PTO (байтовое смещение от V0)    только в многосегментном режиме
SMB170	SMB180	Байт состояния линейного профиля
SMB171	SMB181	Результирующий регистр линейного профиля
SMD172	SMD182	Регистр частот, ручной режим

Таблица 6–37. Справочные данные об управляющем байте PTO/PWM

Управляющий регистр (16-ричное значение)	Деблокировка	Результат выполнения операции PLS						
		Функция	Сегментирование PTO	Метод обновления PWM	База времени	Кол-во импульсов	Ширина импульса	Период следования импульсов
16#81	Да	PTO	Односегм.		1 мкс/цикл			Загружен
16#84	Да	PTO	Односегм.		1 мкс/цикл	Загружено		
16#85	Да	PTO	Односегм.		1 мкс/цикл	Загружено		Загружен
16#89	Да	PTO	Односегм.		1 мс/цикл			Загружен
16#8C	Да	PTO	Односегм.		1 мс/цикл	Загружено		
16#8D	Да	PTO	Односегм.		1 мс/цикл	Загружено		Загружен
16#A0	Да	PTO	Многосегм.		1 мкс/цикл			
16#A8	Да	PTO	Многосегм.		1 мс/цикл			
16#D1	Да	PWM		Синхрон.	1 мкс/цикл			Загружен
16#D2	Да	PWM		Синхрон.	1 мкс/цикл		Загруж.	
16#D3	Да	PWM		Синхрон.	1 мкс/цикл		Загруж.	Загружен
16#D9	Да	PWM		Синхрон.	1 мс/цикл			Загружен
16#DA	Да	PWM		Синхрон.	1 мс/цикл		Загруж.	
16#DB	Да	PWM		Синхрон.	1 мс/цикл		Загруж.	Загружен

## Расчет значений таблицы профилей

Возможность многосегментного режима конвейерной обработки генераторов РТО/PWM может быть полезной во многих приложениях, в частности, при управлении шаговыми двигателями.

Например, вы можете использовать РТО с профилем импульсов для управления шаговым двигателем с помощью простой последовательности, состоящей из разгона, рабочего режима и торможения, или значительно более сложных последовательностей. Для этого вы определяете профиль импульсов, который может включать в себя до 255 сегментов, причем каждый сегмент соответствует некоторой функции, напр., разгон, рабочий режим и торможение.

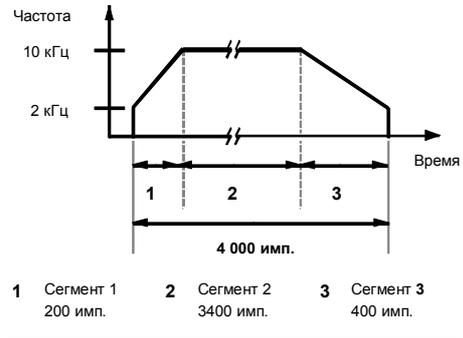


Рис. 6–31. Частотно-временная диаграмма

На рис. 6–31 показан пример значений таблицы профиля, необходимых для генерирования последовательности выходных импульсов, которая разгоняет шаговый двигатель (сегмент 1), обеспечивает его работу с постоянной скоростью (сегмент 2), а затем тормозит двигатель (сегмент 3).

В этом примере: Начальная и конечная частота следования импульсов равна 2 кГц, максимальная частота следования импульсов равна 10 кГц, и для достижения желаемого количества оборотов двигателя требуется 4 000 импульсов. Так как величины в таблице профиля выражаются в терминах периода следования импульсов, а не в частоте, то вы должны преобразовать заданные значения частоты в значения периода следования импульсов. Тогда начальное и конечное значение периода следования импульсов составит 500 мкс, а период следования импульсов, соответствующий максимальной частоте, равен 100 мкс. На интервале ускорения выходного профиля максимальная частота следования импульсов должна быть достигнута примерно через 200 импульсов. Интервал замедления профиля должен быть завершен примерно за 400 импульсов.

Для данного сегмента может быть использована следующая формула для определения приращения периода следования импульсов, которую генератор РТО/PWM использует для настройки периода каждого импульса:

Приращение периода следования импульсов для сегмента =

$$\left| \text{End\_CT}_{\text{seg}} - \text{Init\_CT}_{\text{seg}} \right| / \text{Quantity}_{\text{seg}}$$

где: End\_CT<sub>seg</sub> = конечное значение периода следования импульсов для данного сегмента

Init\_CT<sub>seg</sub> = начальное значение периода следования импульсов для данного сегмента

Quantity<sub>seg</sub> = количество импульсов в данном сегменте

С помощью этой формулы можно рассчитать значения приращений периода следования импульсов для данного примера применения:

Сегмент 1 (ускорение):

Приращение периода следования импульсов = -2

Сегмент 2 (постоянная скорость):

Приращение периода следования импульсов = 0

Сегмент 3 (замедление):

Приращение периода следования импульсов = 1

В таблице 6–38 приведены значения для генерирования последовательности импульсов из примера (предполагается, что таблица профиля находится в памяти переменных, начиная с V500). Вы можете включить в свою программу команды для загрузки этих значений в память переменных, или вы можете определить значения профиля в блоке данных.

Таблица 6–38. Значения таблицы профиля

Адрес	Значение	Описание	
VB500	3	Общее количество сегментов	
VW501	500	Начальный период следования импульсов	Сегмент 1
VW503	-2	Приращение начального периода следования импульсов	
VD505	200	Количество импульсов	Сегмент 2
VW509	100	Начальный период следования импульсов	
VW511	0	Приращение периода следования импульсов	Сегмент 3
VD513	3400	Количество импульсов	
VW517	100	Начальный период следования импульсов	Сегмент 3
VW519	1	Приращение периода следования импульсов	
VD521	400	Количество импульсов	

Чтобы определить, приемлемы ли переходы между сегментами последовательности импульсов, вам нужно знать период для последнего импульса сегмента. Если приращение периода следования импульсов не равно 0, вы должны рассчитать период для последнего импульса сегмента, так как это значение не указано в профиле. Для расчета периода для последнего импульса пользуйтесь следующей формулой:

$$\text{Период последнего импульса сегмента} = \text{Init\_CT}_{\text{seg}} + ( \text{Delta}_{\text{seg}} * ( \text{Quantity}_{\text{seg}} - 1 ) )$$

где:  $\text{Init\_CT}_{\text{seg}}$  = начальный период следования импульсов для этого сегмента

$\text{Delta}_{\text{seg}}$  = приращение периода следования импульсов для этого сегмента

$\text{Quantity}_{\text{seg}}$  = количество импульсов в этом сегменте

В то время как приведенный выше упрощенный пример полезен в качестве введения, реальные приложения могут потребовать более сложных профилей последовательностей импульсов. Помните, что приращение периода следования импульсов может быть задано только как целое количество микросекунд или миллисекунд, и изменение периода выполняется на каждом импульсе.

Влияние этих двух пунктов состоит в том, что расчет приращения периода следования импульсов для данного сегмента может потребовать итеративного подхода. Может потребоваться некоторая гибкость в значении конечного периода следования импульсов или количества импульсов для данного сегмента.

В процессе определения правильных значений таблицы профиля может быть полезна длительность данного сегмента профиля. Время, необходимое для завершения данного сегмента профиля, может быть рассчитано с помощью следующей формулы:

$$\text{Длительность сегмента} = \text{Quantity}_{\text{seg}} * ( \text{Init\_CT} + ( ( \text{Delta}_{\text{seg}}/2 ) * ( \text{Quantity}_{\text{seg}} - 1 ) ) )$$

где:  $\text{Quantity}_{\text{seg}}$  = количество импульсов в данном сегменте

$\text{Init\_CT}_{\text{seg}}$  = начальный период следования импульсов для данного сегмента

$\text{Delta}_{\text{seg}}$  = приращение периода следования импульсов для данного сегмента

## Арифметические операции

### Операции сложения, вычитания, умножения и деления

#### Сложение

$IN1 + IN2 = OUT$   
 $IN1 + OUT = OUT$

#### Вычитание

$IN1 - IN2 = OUT$      *LAD u FBD*  
 $OUT - IN1 = OUT$      *STL*

Команды сложения целых чисел (+I) и вычитания целых чисел (-I) складывают или вычитают два 16-битовых целых числа и дают 16-битовый результат. Команды сложения двойных целых чисел (+D) и вычитания двойных целых чисел (-D) складывают или вычитают два 32-битовых целых числа и дают 32-битовый результат. Команды сложения вещественных чисел (+R) и вычитания вещественных чисел (-R) складывают или вычитают два 32-битовых вещественных числа и дают результат в виде 32-битового вещественного числа.

#### Умножение

$IN1 * IN2 = OUT$   
 $IN1 * OUT = OUT$

#### Деление

$IN1 / IN2 = OUT$      *LAD u FBD*  
 $OUT / IN1 = OUT$      *STL*

Команды умножения целых чисел (\*I) и деления целых чисел (/I) перемножают или делят два 16-битовых целых числа и дают 16-битовый результат. (Для деления остаток не сохраняется.) Команды умножения двойных целых чисел (\*D) и деления двойных целых чисел (/D) перемножают или делят два 32-битовых целых числа и дают 32-битовый результат. (Для деления остаток не сохраняется.) Команды умножения вещественных чисел (\*R) и деления вещественных чисел (/R) перемножают или делят два 32-битовых вещественных числа и дают в результате 32-битовое вещественное число.

#### Биты специальной памяти и ENO

SM1.1 указывает на ошибки переполнения и недопустимые значения. Если бит SM1.1 установлен, то состояние SM1.0 и SM1.2 не имеет значения, и первоначальные входные операнды не меняются. Если SM1.1 и SM1.3 не установлены, то арифметическая операция завершилась с допустимым результатом, и SM1.0 и SM1.2 содержат допустимый статус. Если во время операции деления устанавливается SM1.3, то все остальные биты состояния арифметической операции остаются неизменными.

#### Сбойные состояния, устанавливающие ENO = 0

- SM1.1 (переполнение)
- SM1.3 (деление на ноль)
- 0006 (косвенный адрес)

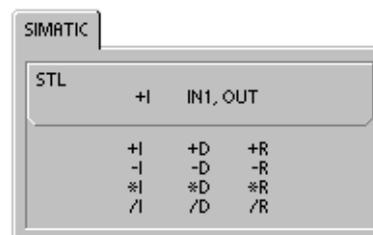
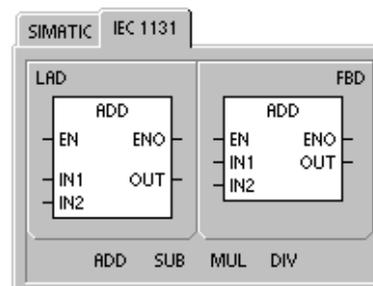
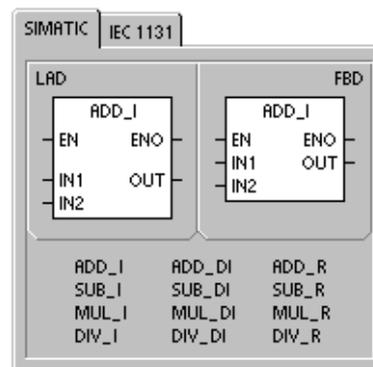
#### Биты специальной памяти, на которые действует команда

- SM1.0 (ноль)
- SM1.1 (переполнение, во время операции получено недопустимое значение или обнаружен недопустимый входной параметр)
- SM1.2 (отрицательное число)
- SM1.3 (деление на ноль)

Таблица 6–39. Допустимые операнды для команд сложения, вычитания, умножения и деления

Входы/выходы	Типы данных	Операнды
IN1, IN2	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, константа
	DINT REAL	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, константа ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, константа
OUT	INT DINT, REAL	IW, QW, VW, MW, SMW, SW, LW, T, C, AC, *VD, *AC, *LD ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Вещественные числа (или числа с плавающей точкой) представляются в формате, описанном в стандарте ANSI/IEEE 754-1985 (одинарная точность). За дополнительной информацией обратитесь к этому стандарту.



**Пример: Арифметические операции с целыми числами**

Network 1

Network 1

```

LD I0.0
+I AC1, AC0
*I AC1, VW100
/I VW10, VW200
                    
```

Сложение

40	+	60	=	100
AC1		AC0		AC0

Умножение

40	*	20	=	800
AC1		VW100		VW100

Деление

4000	/	40	=	100
VW200		VW10		VW200

**Пример: Арифметические операции с вещественными числами**

Network 1

Network 1

```

LD I0.0
+R AC1, AC0
*R AC1, VD100
/R VD10, VD200
                    
```

Сложение

4000.0	+	6000.0	=	10000.0
AC1		AC0		AC0

Умножение

400.0	*	200.0	=	80000.0
AC1		VD100		VD100

Деление

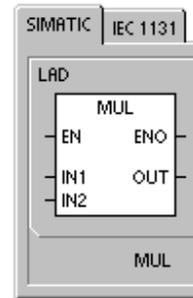
4000.0	/	41.0	=	97.5609
VD200		VD10		VD200

## Умножение целых чисел с представлением результата в виде двойного целого числа и деление целых чисел с остатком

### Умножение целых чисел с представлением результата в виде двойного целого числа

IN1 \* IN2 = OUT LAD и FBD  
 IN1 \* OUT = OUT STL

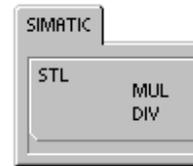
Команда умножения целых чисел с представлением результата в виде двойного целого числа (MUL) перемножает два 16-битовых целых числа и дает 32-битовое произведение. В команде умножения STL младшее слово (16 битов) 32-битового выхода OUT используется как один из сомножителей.



### Деление целых чисел с остатком

IN1 / IN2 = OUT LAD и FBD  
 OUT / IN1 = OUT STL

Команда деления целых чисел с остатком (DIV) делит два 16-битовых целых числа и дает 32-битовый результат, состоящий из 16-битового остатка (старшее слово) и 16-битового частного (младшее слово).



В команде деления STL младшее слово (16 битов) 32-битового выхода OUT используется как делимое.

### Биты специальной памяти и ENO

Для обеих команд, представленных на этой странице, биты специальной памяти (SM) указывают на ошибки и недопустимые значения. Если во время операции деления устанавливается SM1.3 (деление на ноль), то все остальные биты состояния арифметической операции остаются неизменными. В противном случае все поддерживаемые биты состояния арифметической операции содержат после завершения арифметической операции допустимый статус.

#### Сбойные состояния, устанавливающие ENO = 0

- SM1.1 (переполнение)
- SM1.3 (деление на ноль)
- 0006 (косвенный адрес)

#### Биты специальной памяти, на которые действует команда

- SM1.0 (ноль)
- SM1.1 (переполнение)
- SM1.2 (отрицательное число)
- SM1.3 (деление на ноль)

Таблица 6–40. Допустимые операнды для умножения целых чисел с представлением результата в виде двойного целого числа и деления целых чисел с остатком

Входы/выходы	Типы данных	Операнды
IN1, IN2	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, константа
OUT	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

**Пример: Умножение целых чисел с представлением результата в виде двойного целого числа и деление целых чисел с остатком**

Network 1

Network 1

```
LD I0.0
MUL AC1, VD100
DIV VW10, VD200
```

Умножение целых чисел с представлением результата в виде двойного целого числа			
400	*	200	= 80000
AC1		VW102	VD100

Деление целых чисел с остатком			
4000	/	41	= 23 97
VW202		VW10	остат. частн. VW200 VW202 VD200

Примечание: VD100 содержит VW100 и VW102, а VD200 содержит VW200 и VW202.

## Числовые функции

### Синус, косинус и тангенс

Команды вычисления синуса (SIN), косинуса (COS) и тангенса (TAN) вычисляют тригонометрическую функцию угловой величины IN и помещают результат в OUT. Входной угол задается в радианах.

$$\text{SIN (IN) = OUT} \quad \text{COS (IN) = OUT} \quad \text{TAN (IN) = OUT}$$

Для преобразования угла из градусов в радианы: Используйте MUL\_R (\*R) для умножения угла в градусах на 1.745329E-2 (примерно на  $\pi/180$ ).

### Натуральный логарифм и натуральная экспонента

Команда вычисления натурального логарифма находит натуральный логарифм числа в IN и помещает результат в OUT.

Команда вычисления натуральной экспоненты выполняет экспоненциальную операцию по возведению  $e$  в степень, заданную значением в IN, и помещает результат в OUT.

$$\text{LN (IN) = OUT} \quad \text{EXP (IN) = OUT}$$

Для получения десятичного логарифма из натурального: Разделите натуральный логарифм на 2,302585 (примерно натуральный логарифм 10).

Для возведения любого вещественного числа в степень, заданную другим вещественным числом, включая дробные: Объедините команду вычисления натуральной экспоненты с командой вычисления натурального логарифма. Например, для возведения X в степень Y введите следующую команду: EXP (Y \* LN (X)).

### Квадратный корень

Команда вычисления квадратного корня (SQRT) извлекает квадратный корень из вещественного числа (IN) и дает результат в виде вещественного числа OUT.

$$\text{SQRT (IN) = OUT}$$

Для получения других корней:

$$5 \text{ в кубе} = 5^3 = \text{EXP}(3 * \text{LN}(5)) = 125$$

$$\text{Кубический корень из } 125 = 125^{(1/3)} = \text{EXP}((1/3) * \text{LN}(125)) = 5$$

$$\text{Квадратный корень из } 5 \text{ в кубе} = 5^{(3/2)} = \text{EXP}(3/2 * \text{LN}(5)) = 11.18034$$

### Биты специальной памяти и ENO для числовых операций

Для всех команд, описанных на этой странице, SM1.1 используется для указания на ошибки переполнения и недопустимые значения. Если SM1.1 установлен, то состояние SM1.0 и SM1.2 не имеет значения, первоначальные входные операнды не изменяются. Если SM1.1 не установлен, то арифметическая операция завершилась с допустимым результатом, а SM1.0 и SM1.2 содержат действительное состояние.

**Сбойные состояния, устанавливающие ENO = 0**

- SM1.1 (переполнение)
- 0006 (косвенный адрес)

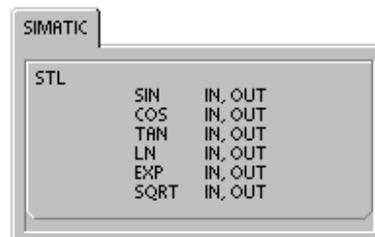
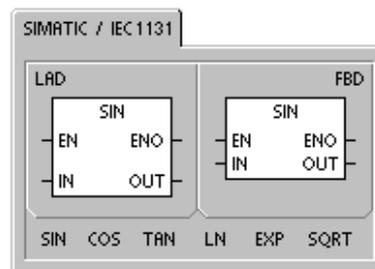
**Биты специальной памяти, на которые действует команда**

- SM1.0 (ноль)
- SM1.1 (переполнение)
- SM1.2 (отрицательное число)

Таблица 6–41. Допустимые операнды числовых функций

Входы/выходы	Типы данных	Операнды
IN	REAL	ID. QD. VD. MD. SMD. SD. LD. AC. *VD. *LD. *AC. константа
OUT	REAL	ID. QD. VD. MD. SMD. SD. LD. AC. *VD. *LD. *AC

Вещественные числа (или числа с плавающей точкой) представляются в формате, описанном в стандарте ANSI/IEEE 754-1985 (одинарная точность). За дополнительной информацией обратитесь к этому стандарту.



## Команды инкрементирования и декрементирования Marker 22

### Увеличение на 1

IN + 1 = OUT      LAD u FBD  
 OUT + 1 = OUT    STL

### Уменьшение на 1

IN - 1 = OUT      LAD u FBD  
 OUT - 1 = OUT    STL

Команды инкрементирования и декрементирования прибавляют 1 к входу IN или вычитают из него 1 и помещают результат в переменную OUT.

Операции увеличения байта на 1 (INCB) и уменьшения байта на 1 (DECB) являются беззнаковыми.

Операции увеличения слова на 1 (INCW) и уменьшения слова на 1 (DECW) учитывают знак.

Операции увеличения двойного слова на 1 (INCD) и уменьшения двойного слова на 1 (DECD) учитывают знак.

### Сбойные состояния, устанавливающие ENO = 0 :

- SM1.1 (переполнение)
- 0006 (косвенный адрес)

### Биты специальной памяти, на которые действует команда:

- SM1.0 (ноль)
- SM1.1 (переполнение)
- SM1.2 (отрицательное число) для операций со словами и двойными словами

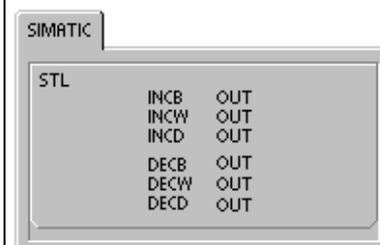
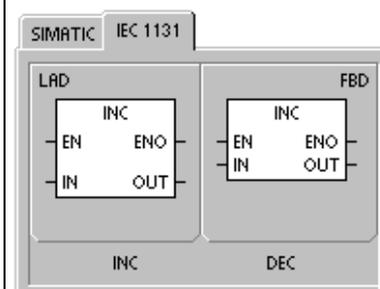
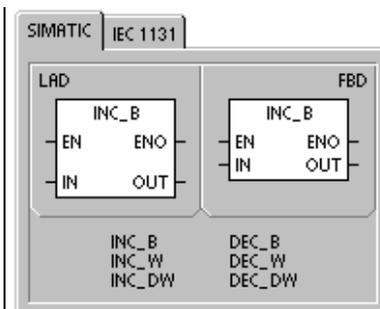


Таблица 6–42. Допустимые операнды для команд инкрементирования и декрементирования

Входы/выходы	Типы данных	Операнды
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, константа
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, константа
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

**Пример: Команды инкрементирования и декрементирования**

Network 1

Network 1

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

Увеличение слова на 1	125	+ 1 =	126
	AC0		AC0
Уменьшение двойного слова на 1	128000	- 1 =	127999
	VD100		VD100

## Пропорционально-интегрально-дифференциальный (PID) регулятор

Команда "PID-регулятор" выполняет расчет контура PID-регулятора для заданного контура регулирования LOOP с помощью данных о входных величинах и конфигурации в таблице (TBL).

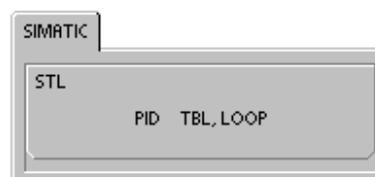
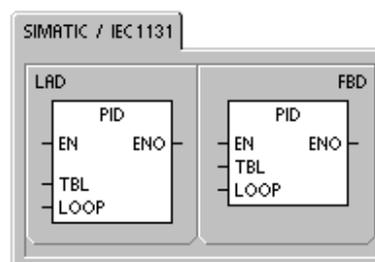
**Сбойные состояния, устанавливающие ENO = 0 :**

- SM1.1 (переполнение)
- 0006 (косвенный адрес)

**Биты специальной памяти, на которые действует команда:**

- SM1.1 (переполнение)

Команда PID-регулятор (пропорционально-интегрально-дифференциальный регулятор) предназначена для расчета PID-регуляторов. Чтобы эти расчеты можно было выполнять, вершина логического стека (TOS) должна быть активизирована (поток сигнала). Команда имеет два операнда: TABLE, являющийся начальным адресом таблицы с данными контура регулирования, и LOOP – номер контура регулирования, являющийся константой от 0 до 7.



В программе можно использовать до восьми команд PID. Если две или более команд PID используются с одним и тем же номером контура регулирования (даже если у них разные номера таблиц), то расчеты PID-регуляторов будут влиять друг на друга, и выход будет непредсказуемым.

Таблица контура регулирования хранит девять параметров, используемых для управления и контроля за работой контура регулирования. Сюда входят текущее и предыдущее значение регулируемой переменной (фактическое значение), заданное значение, регулирующее воздействие (выход), коэффициент усиления, период квантования, постоянная времени интегрирования (или время интегрирования), постоянная времени воздействия по производной (скорость) и интегральная сумма (смещение).

Для выполнения расчета PID-регулятора с желаемым временем квантования команда PID-регулятор должна выполняться или из программы обработки прерывания по времени, или из главной программы со скоростью, управляемой таймером. Время квантования должно подаваться как вход в команду PID-регулятор через таблицу контура регулирования.

В команду PID-регулятор встроена возможность автоматической настройки. В главе 15 вы найдете подробное описание автонастройки. Панель управления настройкой PID работает только с PID-регуляторами, созданными с помощью мастера PID.

Таблица 6–43. Допустимые операнды для PID-регулятора

Входы/выходы	Типы данных	Операнды
TBL	BYTE	VB
LOOP	BYTE	константа (от 0 до 7)



Мастер команд

STEP 7–Micro/WIN предоставляет в распоряжение мастер PID для руководства вами при определении PID-алгоритма для процесса с замкнутым контуром управления. Выберите команду меню **Tools** → **Instruction Wizard** [Инструментальные средства → **Мастер команд**], а затем выберите **PID** из окна Instruction Wizard [Мастер команд].



**Совет**

Заданное значение нижней границы и заданное значение верхней границы должны соответствовать верхней и нижней границе регулируемой переменной.

## PID-алгоритм

В установленном режиме PID-регулятор управляет своим выходом (регулирующим воздействием) таким образом, чтобы свести ошибку регулирования ( $e$ ) к нулю. Мерой ошибки является разность между заданным значением (setpoint, SP) и значением регулируемой переменной (process variable, PV) (фактическое значение). Принцип PID-регулятора основан на следующем уравнении, которое выражает регулирующее воздействие  $M(t)$  как функцию пропорциональной составляющей, интегральной составляющей и дифференциальной составляющей:

Выход (регулирующее воздействие)	=	Пропорциональная составляющая	+	Интегральная составляющая	+	Дифференциальная составляющая
$M(t)$	=	$K_C * e$	+	$K_C \int_0^t e dt + M_{нач}$	+	$K_C * de/dt$
где:	$M(t)$	- регулирующее воздействие (выход регулятора) как функция времени				
	$K_C$	- коэффициент усиления контура регулирования				
	$e$	- ошибка регулирования (разность между заданным значением и регулируемой переменной)				
	$M_{нач}$	- начальное значение регулирующего воздействия				

Чтобы реализовать эту функцию управления в цифровой вычислительной машине, должно быть выполнено квантование непрерывной функции в соответствии с периодическими замерами значения ошибки с последующим расчетом регулирующего воздействия. Соответствующее уравнение, являющееся основой для решения на цифровой вычислительной машине, имеет вид:

$M_n$	=	$K_C * e_n$	+	$K_I * \sum_{x=1}^n e_x + M_{нач}$	+	$K_D * (e_n - e_{n-1})$
Выход (регулирующее воздействие)	=	Пропорциональная составляющая	+	Интегральная составляющая	+	Дифференциальная составляющая
где:	$M_n$	- расчетное значение регулирующего воздействия в момент квантования $n$				
	$K_C$	- коэффициент усиления контура регулирования				
	$e_n$	- значение ошибки регулирования в момент квантования $n$				
	$e_{n-1}$	- предыдущее значение ошибки регулирования (в момент квантования $n - 1$ )				
	$K_I$	- коэффициент пропорциональности интегральной составляющей				
	$M_{нач}$	- начальное значение регулирующего воздействия				
	$K_D$	- коэффициент пропорциональности дифференциальной составляющей				

Из этого уравнения следует, что интегральная составляющая является функцией всех составляющих ошибки от первого до текущего отсчета. Дифференциальная составляющая является функцией текущего и предыдущего отсчета, тогда как пропорциональная составляющая является функцией только текущего отсчета. В цифровой вычислительной машине нецелесообразно хранить все отсчеты ошибки регулирования, да в этом и нет необходимости.

Так как компьютер должен вычислять регулирующее воздействие каждый раз, как опрашивается значение ошибки, начиная с первого отсчета, то необходимо сохранять только предыдущее значение ошибки и предыдущее значение интегральной составляющей. Как результат повторяющейся природы компьютерного решения, может быть получено упрощение уравнения, подлежащего решению в каждый момент квантования. Упрощенное уравнение имеет вид:

$M_n$	=	$K_C * e_n$	+	$K_I * e_n + MX$	+	$K_D * (e_n - e_{n-1})$
Выход (регулирующее воздействие)	=	Пропорциональная составляющая	+	Интегральная составляющая	+	Дифференциальная составляющая
где:	$M_n$	- расчетное значение регулирующего воздействия в момент квантования $n$				
	$K_C$	- коэффициент усиления контура регулирования				
	$e_n$	- значение ошибки регулирования в момент квантования $n$				
	$e_{n-1}$	- предыдущее значение ошибки регулирования (в момент квантования $n - 1$ )				
	$K_I$	- коэффициент пропорциональности интегральной составляющей				
	$MX$	- предыдущее значение интегральной составляющей (в момент квантования $n - 1$ )				
	$K_D$	- коэффициент пропорциональности дифференциальной составляющей				

S7-200 использует модифицированную форму упрощенного выше уравнения при расчете регулирующего воздействия. Это модифицированное уравнение имеет вид:

<b>M<sub>n</sub></b>	<b>=</b>	<b>MP<sub>n</sub></b>	<b>+</b>	<b>MI<sub>n</sub></b>	<b>+</b>	<b>MD<sub>n</sub></b>
Выход (регулирующее воздействие)	=	Пропорциональная составляющая	+	Интегральная составляющая	+	Дифференциальная составляющая

где: M<sub>n</sub> - расчетное значение регулирующего воздействия в момент квантования n  
 MP<sub>n</sub> - значение пропорциональной составляющей регулирующего воздействия в момент квантования n  
 MI<sub>n</sub> - значение интегральной составляющей регулирующего воздействия в момент квантования n  
 MD<sub>n</sub> - значение дифференциальной составляющей регулирующего воздействия в момент квантования n

### Пропорциональная составляющая в уравнении PID-регулятора

Пропорциональная составляющая MP – это произведение коэффициента усиления (K<sub>C</sub>), определяющего точность расчета регулирующего воздействия, и ошибки регулирования (e), представляющей собой разность между заданным значением (SP) и регулируемой переменной (PV) в данный момент квантования. Уравнение для пропорциональной составляющей, решаемое S7-200, имеет вид:

<b>MP<sub>n</sub></b>	<b>=</b>	<b>K<sub>C</sub></b>	<b>*</b>	<b>(SP<sub>n</sub> - PV<sub>n</sub>)</b>	
где: MP <sub>n</sub>	- значение пропорциональной составляющей регулирующего воздействия в момент квантования n	K <sub>C</sub>	- коэффициент усиления контура регулирования	SP <sub>n</sub>	- заданное значение регулируемой величины в момент квантования n
		PV <sub>n</sub>	- значение регулируемой переменной в момент квантования n		

### Интегральная составляющая в уравнении PID-регулятора

Интегральная составляющая MI пропорциональна сумме ошибок за все время управления. Уравнение для интегральной составляющей, решаемое S7-200, имеет вид:

<b>MI<sub>n</sub></b>	<b>=</b>	<b>K<sub>C</sub></b>	<b>*</b>	<b>T<sub>S</sub></b>	<b>/</b>	<b>T<sub>I</sub></b>	<b>*</b>	<b>(SP<sub>n</sub> - PV<sub>n</sub>)</b>	<b>+</b>	<b>MX</b>	
где: MI <sub>n</sub>	- значение интегральной составляющей регулирующего воздействия в момент квантования n	K <sub>C</sub>	- коэффициент усиления контура регулирования	T <sub>S</sub>	- период квантования контура регулирования	T <sub>I</sub>	- постоянная времени интегрирования контура регулирования (называемая также временем интегрирования)	SP <sub>n</sub>	- заданное значение регулируемой величины в момент квантования n	PV <sub>n</sub>	- значение регулируемой переменной в момент квантования n
		MX	- значение интегральной составляющей в момент квантования n – 1 (называемое также интегральной суммой или смещением)								

Интегральная сумма или смещение (MX) – это текущая сумма всех предыдущих значений интегральной составляющей. После каждого расчета MI<sub>n</sub> смещение обновляется значением MI<sub>n</sub>, которое может быть согласовано или ограничено (подробности см. в разделе «Переменные и диапазоны»). Начальное значение смещения обычно устанавливается равным значению регулирующего воздействия (M<sub>нач</sub>) сразу перед его первым расчетом для контура регулирования. Частью интегральной составляющей являются также несколько констант: коэффициент усиления (K<sub>C</sub>), период квантования (T<sub>S</sub>), представляющий собой время цикла, с которым PID-регулятор пересчитывает регулирующее воздействие, и постоянная времени интегрирования (или сброс) (T<sub>I</sub>), которая используется для управления влиянием интегральной составляющей на расчет регулирующего воздействия.

### Дифференциальная составляющая в уравнении PID-регулятора

Дифференциальная составляющая MD пропорциональна изменению ошибки регулирования. S7-200 использует следующее уравнение для расчета дифференциальной составляющей:

$$MD_n = K_C * \left( \frac{T_D}{T_S} * ((SP_n - PV_n) - (SP_{n-1} - PV_{n-1})) \right)$$

Во избежание ступенчатых изменений или скачков регулирующего воздействия при изменениях заданного значения это уравнение модифицировано в предположении, что заданное значение постоянно ( $SP_n = SP_{n-1}$ ). В результате рассчитывается изменение регулируемой переменной, а не изменение ошибки регулирования. Это показывает следующее уравнение:

$MD_n =$	$K_C$	*	$\frac{T_D}{T_S}$	*	$(SP_n - PV_n - SP_n + PV_{n-1})$
или:					
$MD_n =$	$K_C$	*	$\frac{T_D}{T_S}$	*	$(PV_{n-1} - PV_n)$
где:	$MD_n$	- значение дифференциальной составляющей регулирующего воздействия в момент квантования n			
	$K_C$	- коэффициент усиления контура регулирования			
	$T_S$	- период квантования контура регулирования			
	$T_D$	- постоянная времени воздействия по производной контура регулирования (называемая также временем упреждения)			
	$SP_n$	- заданное значение регулируемой величины в момент квантования n			
	$SP_{n-1}$	- заданное значение регулируемой величины в момент квантования n-1			
	$PV_n$	- значение регулируемой переменной в момент квантования n			
	$PV_{n-1}$	- значение регулируемой переменной в момент квантования n-1			

Для использования в следующих расчетах дифференциальной составляющей должна сохраняться регулируемая переменная, а не ошибка регулирования. При первом отсчете значение  $PV_{n-1}$  инициализируется равным  $PV_n$ .

### Выбор типа регулятора

Во многих системах управления может оказаться необходимым использовать только один или два метода регулирования. Например, может потребоваться только пропорциональное или пропорционально-интегральное управление. Выбор желаемого типа регулятора выполняется установкой значений постоянных параметров.

Если вам не нужно интегральное воздействие (нет составляющей "I" в расчете PID-регулятора), то постоянная времени интегрирования должна быть задана равной бесконечности. Даже при отсутствии интегрального воздействия значение этой составляющей не может быть равно нулю из-за начального значения интегральной суммы MX.

Если вам не нужно дифференцирующее воздействие (нет составляющей "D" в расчете PID-регулятора), то постоянная времени воздействия по производной (упреждение) должна быть задана равной 0.0.

Если вам не нужно пропорциональное воздействие (нет составляющей "P" в расчете PID-регулятора), а вы хотите иметь интегральный или интегрально-дифференциальный регулятор, то значение 0.0 должно быть задано для коэффициента усиления. Так как усиление является коэффициентом в уравнениях для расчета интегральной и дифференциальной составляющей, установка значения 0.0 для коэффициента усиления контура регулирования приведет к тому, что при расчете интегральной и дифференциальной составляющей для коэффициента усиления будет использовано значение 1.0.

### Преобразование и нормализация входов контура регулирования

Контур регулирования имеет две входных переменных – заданное значение и регулируемую переменную. Заданное значение – это обычно фиксированная величина, например, уставка скорости для регулятора скорости движения в вашем автомобиле. Регулируемая переменная – это величина, связанная с регулирующим воздействием, поэтому она измеряет влияние, оказываемое регулирующим воздействием на управляемую систему. В примере с регулятором скорости движения автомобиля регулируемой переменной является вход тахометра, измеряющего скорость вращения колес.

Заданное значение и регулируемая переменная – это реальные физические величины, диапазон значений которых и единицы измерения могут быть различными. Прежде чем эти физические величины могут быть использованы командой PID-регулятор, они должны быть преобразованы в нормализованные представления с плавающей точкой.

Первый шаг состоит в преобразовании физической величины, представленной в виде 16-битового целого числа в вещественное число, или число с плавающей точкой. Следующая последовательность команд показывает, как можно преобразовать целое число в вещественное.

ITD	AIW0, AC0	// Преобразовать входную величину в двойное слово
DTR	AC0, AC0	// Преобразовать 32-битовое целое в вещественное число

Следующий шаг состоит в преобразовании вещественного представления реальной физической величины в нормализованное значение между 0.0 и 1.0. Для нормализации заданного значения или регулируемой переменной используется следующее уравнение:

<b>R<sub>Norm</sub></b>	<b>=</b>	<b>((R<sub>Raw</sub> / Span) + Offset)</b>
где:	R <sub>Norm</sub>	- нормализованное вещественное число, представляющее реальную физическую величину
	R <sub>Raw</sub>	- ненормализованное, или необработанное (raw), вещественное представление реальной физической величины
	Offset	- смещение, равное 0.0 для униполярных величин, и 0,5 для биполярных величин
	Span	- диапазон, равный разности между максимально возможным и минимально возможным значением: = 32 000 для униполярных величин (обычно) = 64 000 для биполярных величин (обычно)

Следующая последовательность команд, являющаяся продолжением предыдущей последовательности, показывает, как нормализовать биполярную величину в AC0 (с диапазоном 64 000):

/R	64000.0, AC0	//Нормализовать значение в аккумуляторе
+R	0.5, AC0	//Сместить значение в диапазон от 0.0 до 1.0
MOVR	AC0, VD100	//Сохранить нормализованное значение в таблице контура регулирования
		//TABLE

### Преобразование регулирующего воздействия в масштабированную целую величину

Регулирующее воздействие – это управляющая переменная, например, установка дроссельной заслонки в примере с регулятором скорости движения автомобиля. Регулирующее воздействие – это нормализованное вещественное значение между 0.0 и 1.0. Прежде чем регулирующее воздействие может быть использовано для управления аналоговым выходом, оно должно быть преобразовано в 16-битовую масштабированную целую величину. Этот процесс противоположен преобразованию PV и SP в нормализованную величину. Первый шаг состоит в преобразовании регулирующего воздействия в масштабированное вещественное число с помощью следующей формулы:

<b>R<sub>Scal</sub></b>	<b>=</b>	<b>(M<sub>n</sub> – Offset) * Span</b>
где:	R <sub>Scal</sub>	- масштабированное (scaled) вещественное значение регулирующего воздействия
	M <sub>n</sub>	- нормализованное вещественное значение регулирующего воздействия
	Offset	- смещение, равное 0.0 для униполярных величин, и 0,5 для биполярных величин
	Span	- диапазон, равный разности между максимально возможным и минимально возможным значением = 32 000 для униполярных величин (обычно) = 64 000 для биполярных величин (обычно)

Следующая последовательность команд показывает, как масштабировать регулирующее воздействие:

```
MOVW  VD108, AC0      //Передать регулирующее воздействие в аккумулятор
-R     0.5, AC0        //Этот оператор включить только для биполярного значения
*R     64000.0, AC0    //Масштабировать значение в аккумуляторе
```

Затем масштабированное вещественное представление регулирующего воздействия должно быть преобразовано в 16-битовое целое. Как выполнить это преобразование, показывает следующая последовательность команд:

```
ROUND  AC0, AC0       //Преобразовать вещественное число в 32-битовое целое
DTI    AC0, LW0       //Преобразовать эту величину в 16-битовое целое число
MOVW   LW0, AQW0      //Записать это значение в аналоговый выход
```

## Контуры регулирования с положительной и отрицательной обратной связью

Контур регулирования имеет положительную обратную связь, если его коэффициент усиления положителен, и отрицательную обратную связь, если его коэффициент усиления отрицателен. (Для интегрального и интегрально-дифференциального регулятора, где значение коэффициента усиления равно 0.0, задание положительных значений для постоянных времени интегрирования и воздействия по производной приведет к положительной обратной связи, а задание отрицательных значений – к отрицательной обратной связи.)

### Переменные и диапазоны

Регулируемая переменная и заданное значение являются входами для расчета PID-регулятора. Поэтому поля таблицы контура регулирования для этих переменных могут считываться, но не могут быть изменены командой PID-регулятор.

Регулирующее воздействие генерируется как результат расчетов PID-регулятора, так что поле регулирующего воздействия в таблице контура регулирования обновляется после каждого расчета PID-регулятора. Регулирующее воздействие заключено между 0.0 и 1.0. Поле регулирующего воздействия может быть использовано пользователем для задания начального регулирующего воздействия при организации перехода от ручного управления к автоматическому с помощью команды PID-регулятор (см. обсуждение этого вопроса в нижеследующем разделе «Режимы»).

Если используется интегральный регулятор, то значение смещения обновляется в результате расчета PID-регулятора, и обновленное значение используется как вход в следующем расчете PID-регулятора. Если рассчитанное регулирующее воздействие выходит за пределы допустимого диапазона (меньше 0.0 или больше 1.0), то смещение корректируется в соответствии со следующими формулами:

<b>MX</b>	<b>=</b>	<b>1.0</b>	<b>-</b>	<b>(MP<sub>n</sub> + MD<sub>n</sub>)</b>	<i>если расчетное значение M<sub>n</sub> &gt; 1.0</i>
или					
<b>MX</b>	<b>=</b>	<b>-</b>	<b>(MP<sub>n</sub> + MD<sub>n</sub>)</b>		<i>если расчетное значение M<sub>n</sub> &lt; 0.0</i>
где:	MX	- значение скорректированного смещения			
	MP <sub>n</sub>	- значение пропорциональной составляющей регулирующего воздействия в момент квантования p			
	MD <sub>n</sub>	- значение дифференциальной составляющей регулирующего воздействия в момент квантования p			
	M <sub>n</sub>	- значение регулирующего воздействия в момент квантования p			

При корректировке смещения описанным выше способом достигается улучшение чувствительности системы, когда рассчитанное регулирующее воздействие возвращается в надлежащий диапазон. Рассчитанное смещение также устанавливается в диапазоне между 0.0 и 1.0, а затем записывается в поле смещения таблицы контура регулирования при завершении каждого расчета PID-регулятора. Значение, сохраняемое в таблице контура регулирования, используется в следующем цикле расчетов PID-регулятора.

Значение смещения в таблице контура регулирования может быть изменено пользователем перед выполнением команды PID-регулятор для оказания воздействия на определенные ситуации в некоторых приложениях. При ручной корректировке смещения необходимо учитывать, что любое смещение, записываемое в таблицу контура регулирования, должно быть вещественным числом в диапазоне между 0.0 и 1.0.

Значение регулируемой величины, используемое для сравнения при расчете дифференциальной составляющей PID-регулятора, сохраняется в таблице контура регулирования. Это значение не может быть изменено пользователем.

## Режимы

Для PID-регуляторов S7-200 нет встроенного управления режимом работы. Расчет PID-регулятора выполняется только тогда, когда поток сигнала достигает блока PID. Поэтому «автоматический» режим имеет место, когда расчеты PID-регулятора выполняются циклически. «Ручной» режим имеет место, когда расчеты PID-регулятора не выполняются.

Команда PID-регулятор имеет бит истории потока сигнала, аналогичный биту, используемому в команде счета. Команда использует этот бит для обнаружения изменения потока сигнала с 0 на 1. Когда изменение потока сигнала обнаруживается, это заставляет команду выполнить ряд действий, обеспечивающих плавный переход от ручного управления к автоматическому. Для плавного перехода к автоматическому режиму управления значение регулирующего воздействия, установленное при ручном управлении, должно быть передано в качестве входа команде PID-регулятор (записано в таблицу контура регулирования для  $M_n$ ) перед переключением в автоматический режим управления. Команда PID-регулятор выполняет следующие действия с величинами в таблице контура регулирования, чтобы обеспечить плавный переход от ручного режима управления к автоматическому при обнаружении перехода потока сигнала из 0 в 1:

- устанавливает заданное значение ( $SP_n$ ) = регулируемой переменной ( $PV_n$ )
- устанавливает старое значение регулируемой переменной ( $PV_{n-1}$ ) = регулируемой переменной ( $PV_n$ )
- устанавливает смещение ( $MX$ ) = регулирующему воздействию ( $M_n$ )

По умолчанию бит истории PID-регулятора установлен, и это состояние формируется при запуске и при каждом переходе контроллера из STOP в RUN. Если сигнал достигает блока PID при первом его исполнении после перехода в режим RUN, то переход потока сигнала из 0 в 1 не обнаруживается, и действия, обеспечивающие плавный переход, не выполняются.

## Аварийный контроль и специальные операции

PID-регулятор – это простая, но мощная команда, выполняющая расчеты, необходимые для PID-регулирования. Если требуются другие функции, например, контроль аварийных ситуаций или выполнение специальных расчетов с переменными контура регулирования, то они должны быть реализованы с помощью основных команд, поддерживаемых S7-200.

## Сбойные ситуации

Если начальный адрес таблицы контура регулирования или операнды контура регулирования, указанные в команде, находятся вне допустимого диапазона, то при компиляции CPU выдаст ошибку компиляции (ошибка диапазона), и компиляция потерпит неудачу.

Некоторые входные значения таблицы контура регулирования не проверяются на соответствие допустимому диапазону командой PID-регулятор. Вы должны позаботиться о том, чтобы регулируемая переменная и заданное значение (а также смещение и предыдущее значение регулируемой переменной, если они используются в качестве входов) были вещественными числами в диапазоне между 0.0 и 1.0.

Если при выполнении арифметических операций в расчетах PID-регулятора встретится любая ошибка, то будет установлен бит SM1.1 (переполнение или недопустимое значение), и выполнение команды PID-регулятор будет завершено. (Обновление выходных значений в таблице контура регулирования может быть неполным, так что вы не должны принимать эти значения во внимание, а исправить входное значение, вызвавшее арифметическую ошибку перед следующим выполнением команды PID-регулятор.)

### Таблица контура регулирования

Таблица контура регулирования имеет длину 80 байтов и формат, показанный в таблице 6–44.

Таблица 6–44. Таблица контура регулирования

Смещение	Поле	Формат	Тип	Описание
0	Регулируемая переменная ( $PV_n$ )	REAL	In	Содержит регулируемую переменную, которая должна быть масштабирована между 1.0.
4	Заданное значение ( $SP_n$ )	REAL	In	Содержит заданное значение, которое должно быть масштабировано между 0.0 и 1.0.
8	Регулирующее воздействие ( $M_n$ )	REAL	In/Out	Содержит расчетное регулирующее воздействие, масштабированное между 0.0 и 1.0.
12	Коэффициент усиления ( $K_C$ )	REAL	In	Содержит усиление, являющееся коэффициентом пропорциональности. Может быть положительным или отрицательным числом.
16	Период квантования ( $T_S$ )	REAL	In	Содержит период квантования в секундах. Должен быть положительным числом.
20	Постоянная времени интегрирования ( $T_I$ )	REAL	In	Содержит постоянную времени интегрирования в минутах. Должна быть положительным числом.
24	Постоянная времени воздействия по производной ( $T_D$ )	REAL	In	Содержит постоянную времени воздействия по производной в минутах. Должна быть положительным числом.
28	Смещение ( $MX$ )	REAL	In/Out	Содержит значение смещения или интегральной суммы в пределах от 0.0 до 1.0.
32	Предыдущее значение регулируемой переменной ( $PV_{n-1}$ )	REAL	In/Out	Содержит предыдущее значение регулируемой переменной, сохраняемое после последнего выполнения команды PID-регулятор.
36 – 79	Зарезервировано для переменных автонастройки. Подробности см. в таблице 15–1.			

## Команды прерывания

### Разблокирование и блокирование прерываний

Команда разблокирования прерываний (ENI) разблокирует обработку всех назначенных прерывающих событий. Команда блокирования прерываний (DISI) блокирует обработку всех прерывающих событий.

Когда вы переходите в режим RUN, прерывания первоначально заблокированы. Находясь в режиме RUN, вы можете разблокировать все прерывания, выполнив команду разблокирования прерываний. Выполнение команды блокирования прерываний запрещает обработку прерываний, однако активные прерывающие события и далее будут ставиться в очередь.

**Сбойные состояния, устанавливающие ENO = 0 :**

- 0004 (попытка исполнения команд ENI, DISI или HDEF в программе обработки прерываний)

### Условный возврат из программы обработки прерываний

Команда условного возврата из программы обработки прерываний (CRETI) может быть использована для возврата из программы обработки прерываний в зависимости от условия, задаваемого предшествующей логикой.

### Назначение прерывания

Команда назначения прерывания (ATCH) связывает прерывающее событие EVNT с номером программы обработки прерываний INT и разблокирует прерывающее событие.

**Сбойные состояния, устанавливающие ENO = 0 :**

- 0002 (конфликт при назначении входов для HSC)

### Отсоединение прерывания

Команда отсоединения прерывания (DTCH) разрывает связь прерывающего события EVNT со всеми программами обработки прерываний и блокирует прерывающее событие.

### Очистка прерывающих событий

Команда очистки прерывающих событий удаляет все прерывающие события типа EVNT из очереди прерываний. Эта команда используется для очистки очереди прерываний от нежелательных прерывающих событий. Если эта команда используется для удаления ложных прерывающих событий, вы должны отсоединить это событие перед удалением событий из очереди. Иначе после выполнения команды очистки прерывающих событий к очереди будут добавлены новые события.

В примере показан скоростной счетчик в квадратном режиме, использующий команду CLR\_EVNT для удаления прерываний. Если шаговый датчик оптического устройства остановился на границе перехода от света к тьме, то небольшие колебания машины могут вызвать нежелательные прерывания перед тем, как сможет быть загружена новое значение PV.

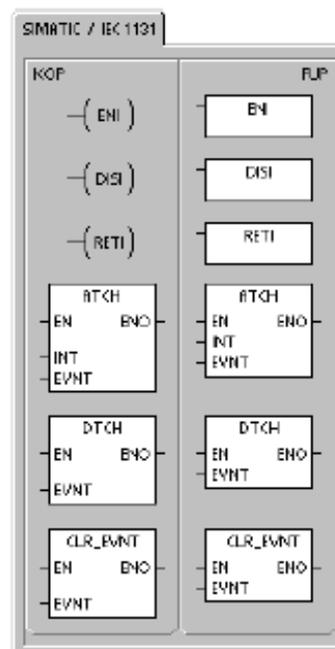


Таблица 6–45. Допустимые операнды для команд прерывания

Входы/выходы	Типы данных	Операнды
INT	BYTE	константа (от 0 до 127)
EVNT	BYTE	константа CPU 221 и CPU 222: от 0 до 12, от 19 до 23 и от 27 до 33 CPU 224: от 0 до 23 и от 27 до 33 CPU 224XP и CPU 226: от 0 до 33

## Принцип действия команд назначения и отсоединения прерываний

Прежде чем программа обработки прерывания может быть вызвана, должно быть установлено соответствие между прерывающим событием и сегментом программы, который вы хотите выполнить, когда это событие происходит. Для организации связи между прерывающим событием (задаваемым номером прерывающего события) и сегментом программы (задаваемым номером программы обработки прерывания) используйте команду назначения прерывания (ATCH). Одной программе обработки прерываний можно поставить в соответствие несколько прерывающих событий, но одно событие не может быть одновременно поставлено в соответствие нескольким программам обработки прерываний.

Когда вы назначаете прерывающее событие программе обработки прерываний, это прерывание автоматически разблокируется. Если вы заблокировали все прерывания с помощью команды блокирования прерываний, то каждое возникновение прерывающего события ставится в очередь, пока прерывания не будут снова разблокированы с помощью команды разблокирования прерываний или не произойдет переполнение очереди прерываний.

Отдельные прерывающие события можно заблокировать разрывом связи между этим прерывающим событием и программой обработки прерывания с помощью команды отсоединения прерывания. Команда отсоединения возвращает прерывание в неактивное или игнорируемое состояние. Таблица 6–46 перечисляет различные типы прерывающих событий.

Таблица 6–46. Прерывающие события

Событие	Описание	CPU 221 CPU 222	CPU 224	CPU 224XP CPU 226
0	I0.0 Нарастающий фронт	да	да	да
1	I0.0 Падающий фронт	да	да	да
2	I0.1 Нарастающий фронт	да	да	да
3	I0.1 Падающий фронт	да	да	да
4	I0.2 Нарастающий фронт	да	да	да
5	I0.2 Падающий фронт	да	да	да
6	I0.3 Нарастающий фронт	да	да	да
7	I0.3 Падающий фронт	да	да	да
8	Порт 0 Символ принят	да	да	да
9	Порт 0 Передача завершена	да	да	да
10	Управляемое временем прерывание 0 SMB34	да	да	да
11	Управляемое временем прерывание 1 SMB35	да	да	да
12	HSC0 CV=PВ (текущее значение = предустановленному)	да	да	да
13	HSC1 CV=PВ (текущее значение = предустановленному)		да	да
14	HSC1 Изменение направления		да	да
15	HSC1 Внешний сброс		да	да
16	HSC2 CV=PВ (текущее значение = предустановленному)		да	да
17	HSC2 Изменение направления		да	да
18	HSC2 Внешний сброс		да	да
19	PLS0 Прерывание по завершению отсчета количества импульсов PTO	да	да	да
20	PLS1 Прерывание по завершению отсчета количества импульсов PTO	да	да	да
21	Таймер T32 Прерывание СТ=PT	да	да	да
22	Таймер T96 Прерывание СТ=PT	да	да	да
23	Порт 0 Прием сообщения завершен	да	да	да
24	Порт 1 Прием сообщения завершен			да
25	Порт 1 Символ принят			да
26	Порт 1 Передача завершена			да
27	HSC0 Изменение направления	да	да	да

Таблица 6–46. Прерывающие события, продолжение

28	HSC0	Внешний сброс	да	да	да
29	HSC4	CV=PV (текущее значение = предустановленному)	да	да	да
30	HSC4	Изменение направления	да	да	да
31	HSC4	Внешний сброс	да	да	да
32	HSC3	CV=PV (текущее значение = предустановленному)	да	да	да
33	HSC5	CV=PV (текущее значение = предустановленному)	да	да	да

## Исполнение программ обработки прерываний в S7-200

Программа обработки прерывания выполняется в ответ на соответствующее внутреннее или внешнее событие. После выполнения последней команды программы обработки прерывания управление возвращается в главную программу. Вы можете покинуть эту программу, выполнив команду условного возврата из программы обработки прерываний (CRETI). В таблице 6–47 приведены некоторые указания и ограничения по использованию программ обработки прерываний в вашей программе.

Таблица 6–47. Рекомендации и ограничения по использованию программ обработки прерываний

<b>Указания</b>
Обработка прерываний обеспечивает быструю реакцию на определенные внутренние или внешние события. Вам следует оптимизировать программы обработки прерываний, чтобы выполнить конкретную задачу, а затем вернуть управление главной программе. Если программа обработки прерывания спроектирована короткой с точными спецификациями, то она будет быстро выполняться и не будет задерживать другие процессы на длительные промежутки времени. Если этого не сделать, то неожиданные условия могут вызвать ненормальную работу оборудования, управляемого главной программой. Для прерываний безусловно верна аксиома «чем короче, тем лучше».
<b>Ограничения</b>
В программе обработке прерывания нельзя использовать команды блокирования прерываний (DISI), разблокирования прерываний (ENI), определения режима работы скоростного счетчика (HDEF) и завершения обработки (END).

### Системная поддержка прерываний

Так как прерывания могут оказывать влияние на контакты, катушки и аккумуляторы, то система сохраняет и перезагружает логический стек, аккумуляторные регистры и биты специальной памяти (SM), которые отображают состояние аккумуляторов и команд. Это позволяет избежать искажения главной программы пользователя из-за перехода в программу обработки прерывания и возвращения из нее.

### Совместное использование данных главной программой и программами обработки прерываний

Данные могут совместно использоваться главной программой и одной или несколькими программами обработки прерываний. Так как невозможно предсказать, когда S7-200 может сгенерировать прерывание, то желательно ограничить количество переменных, которые применяются как в программе обработки прерываний, так и в других местах программы. В результате действий программы обработки прерываний могут возникнуть проблемы непротиворечивости совместно используемых данных, когда выполнение команд вашей главной программы прерывается событиями, вызывающими прерывания. Чтобы гарантировать, что ваша программа обработки прерываний будет использовать только временную память и не перезапишет данные, используемые еще в каком-либо месте вашей программы, пользуйтесь таблицей локальных переменных программы обработки прерываний.

Существует ряд методов программирования, которые вы можете использовать, чтобы обеспечить корректное разделение данных между вашей главной программой и программами обработки прерываний. Эти методы или ограничивают способ доступа к совместно используемым ячейкам памяти, или препятствуют прерыванию последовательностей команд, использующих разделяемые ячейки памяти.

- Для программы на STL, совместно использующей только одну переменную: Если совместно используемые данные представляют собой одну переменную в виде байта, слова или двойного слова, и ваша программа написана на STL, то корректный доступ к совместно используемым данным может быть обеспечен сохранением промежуточных результатов операций над совместно используемыми данными только в тех адресах памяти или аккумуляторах, которые совместно не используются.
- Для программы на LAD, которая совместно использует единственную переменную: Если разделяемые данные представляют собой единственную переменную в виде байта, слова или двойного слова и ваша программа написана на LAD, то корректный совместный доступ может быть обеспечен установлением соглашения, что доступ к разделяемым ячейкам памяти может осуществляться только с помощью команд пересылки (MOVB, MOVW, MOVD, MOVR). В то время как многие команды LAD составлены из непрерываемых последовательностей команд STL, команды пересылки состоят из единственной команды STL, на исполнение которой не могут влиять прерывающие события.
- Для программы на STL или LAD, совместно использующей несколько переменных: Если разделяемые данные составлены из ряда связанных байтов, слов или двойных слов, то для управления исполнением программ обработки прерываний могут быть использованы команды блокировки/разблокировки прерываний (DISI и ENI). В той точке вашей программы, где должны начаться операции с разделяемыми ячейками памяти, заблокируйте прерывания. Как только все действия, влияющие на совместно используемые ячейки памяти, завершены, вновь разблокируйте прерывания. В течение времени, когда прерывания заблокированы, программы обработки прерываний не могут выполняться и, следовательно, не имеют доступа к разделяемым ячейкам памяти; однако такой подход может привести к запаздыванию реакции на прерывающие события.

### **Вызов подпрограмм из программ обработки прерываний**

Из программы обработки прерывания можно вызвать только один уровень вложенности подпрограмм. Аккумуляторы и логический стек совместно используются программой обработки прерывания и вызываемой подпрограммой.

## **Виды прерываний, поддерживаемых S7–200**

S7–200 поддерживает следующие виды программ обработки прерываний:

- Прерывания коммуникационных портов: S7–200 генерирует события, которые позволяют вашей программе управлять коммуникационным портом.
- Прерывания по вводу/выводу: S7–200 генерирует события для различных изменений состояния различных входов-выходов. Эти события позволяют вашей программе реагировать на скоростные счетчики, вывод импульсов и на нарастающие или падающие фронты на входах.
- Прерывания, управляемые временем: S7–200 генерирует события, которые позволяют вашей программе реагировать через определенные интервалы времени.

### **Прерывания коммуникационных портов**

Ваша программа может управлять последовательным коммуникационным портом S7–200. Этот режим работы коммуникационного порта называется свободно программируемым обменом данными (Freerport). В этом режиме ваша программа определяет скорость передачи, количество битов на символ, способ контроля четности и протокол. Для облегчения обмена данными, управляемого вашей программой, в вашем распоряжении имеются прерывания приема и передачи. За дополнительной информацией обратитесь к командам приема и передачи.

### **Прерывания по вводу/выводу**

К прерываниям по вводу/выводу относятся прерывания при нарастающем/падающем фронте, прерывания от скоростных счетчиков и прерывания от последовательности импульсов. S7–200 может генерировать прерывание при нарастающем и/или падающем фронте на входе (I0.0, I0.1, I0.2 или I0.3). Появления нарастающего и падающего фронта могут быть распознаны для любой из этих точек ввода. Эти события могут использоваться для отображения условия, которое немедленно должно быть принято во внимание, когда это событие происходит.

Прерывания от скоростных счетчиков дают возможность реагировать на такие условия, как достижение текущим значением предустановленного значения, изменение направления счета, которое может соответствовать реверсированию направления, в котором вращается вал, или внешний сброс счетчика. Каждое из этих прерываний дает возможность предпринимать в реальном времени действия в ответ на быстрые события, которыми нельзя управлять при скоростях, определяемых временем цикла программируемого логического контроллера.

Прерывания от последовательности импульсов немедленно извещают о завершении вывода предписанного количества импульсов. Импульсные последовательности часто используются для управления шаговыми двигателями.

Каждое из вышеописанных прерываний может быть разблокировано назначением программы обработки прерывания соответствующему событию ввода/вывода.

### Прерывания, управляемые временем

К прерываниям, управляемым временем, относятся циклические прерывания и прерывания, вызываемые таймерами T32 и T96. С помощью циклических прерываний вы можете задать действия, которые должны выполняться циклически. Время цикла устанавливается в пределах от 1 до 255 мс шагами по 1 мс. Вы должны записать время цикла в SMB34 для циклического прерывания 0 и в SMB35 для циклического прерывания 1. Событие, вызывающее циклические прерывания, передает управление соответствующей программе обработки прерываний каждый раз, как истекает время работы таймера. Обычно циклические прерывания используются для управления опросом аналоговых входов или для вызова PID-регулятора через регулярные интервалы времени.

Циклическое прерывание разблокируется, и начинается отсчет времени, когда вы назначаете программу обработки прерывания событию, вызывающему циклическое прерывание. При этом система воспринимает значение времени цикла, и последующие изменения в SMB34 и SMB35 на это время цикла влияния не оказывают. Чтобы изменить время цикла, вы должны задать для него новое значение, а затем снова назначить программу обработки прерывания событию, вызывающему циклическое прерывание. Когда происходит повторное назначение, функция циклического прерывания сбрасывает все накопленное время от предыдущего назначения и начинает отсчет времени с новым значением.

Будучи разблокированным, циклическое прерывание работает постоянно, выполняя назначенную программу обработки прерывания при каждом истечении заданного временного интервала. Если вы выйдете из режима RUN или отсоедините циклическое прерывание, то циклическое прерывание блокируется. Если выполняется глобальная команда блокирования прерываний, то циклические прерывания продолжают происходить. Каждое появление циклического прерывания ставится в очередь (пока прерывания не будут разблокированы или не заполнятся очередь).

Прерывания от таймера T32 или T96 позволяют своевременно реагировать на завершение заданного интервала времени. Эти прерывания поддерживаются только для таймеров T32 и T96 с задержкой включения (TON) и с задержкой выключения (TOF) с разрешением 1 мс. В противном случае таймеры T32 и T96 обладают обычными свойствами. Когда прерывание разблокировано, назначенная программа обработки прерывания выполняется, когда текущее значение активного таймера становится равным предустановленному значению во время нормального обновления 1-миллисекундного таймера, выполняемого в S7-200. Эти прерывания разблокируются назначением программы обработки прерывания событиям, вызывающим прерывание от таймера T32/T96.

### Приоритет прерываний и постановки их в очередь

Прерывания обслуживаются S7-200 в порядке их возникновения с учетом соответствующей группы приоритета. В любой момент времени исполняется только одна программа обработки прерывания. Когда исполнение программы обработки прерывания начинается, программа исполняется до своего завершения. Она не может быть прервана другой программой обработки прерывания, даже если последняя имеет более высокий приоритет. Прерывания, возникающие во время обработки другого прерывания, ставятся в очередь для последующей обработки.

Три очереди прерываний и максимальное количество прерываний, которое они могут хранить, показаны в таблице 6-48.

Таблица 6-48. Максимальное количество записей на очередь прерываний

Очередь	CPU 221, CPU 222, CPU 224	CPU 224XP и CPU 226
Очередь коммуникационных прерываний	4	8
Очередь прерываний по вводу/выводу	16	16
Очередь прерываний, управляемых временем	8	8

В принципе может возникнуть больше прерываний, чем может содержать очередь. Поэтому системой поддерживаются биты переполнения очереди (обозначающие тип прерывающих событий, которые были потеряны). Биты переполнения очереди прерываний показаны в таблице 6-49. Вам следует использовать эти биты только в программе обработки прерывания, так как они сбрасываются, когда очередь опустошается и управление возвращается главной программе.

В табл. 6–50 представлены все прерывающие события, их приоритеты и соответствующие номера событий.

Таблица 6–49. Биты переполнения очереди прерываний

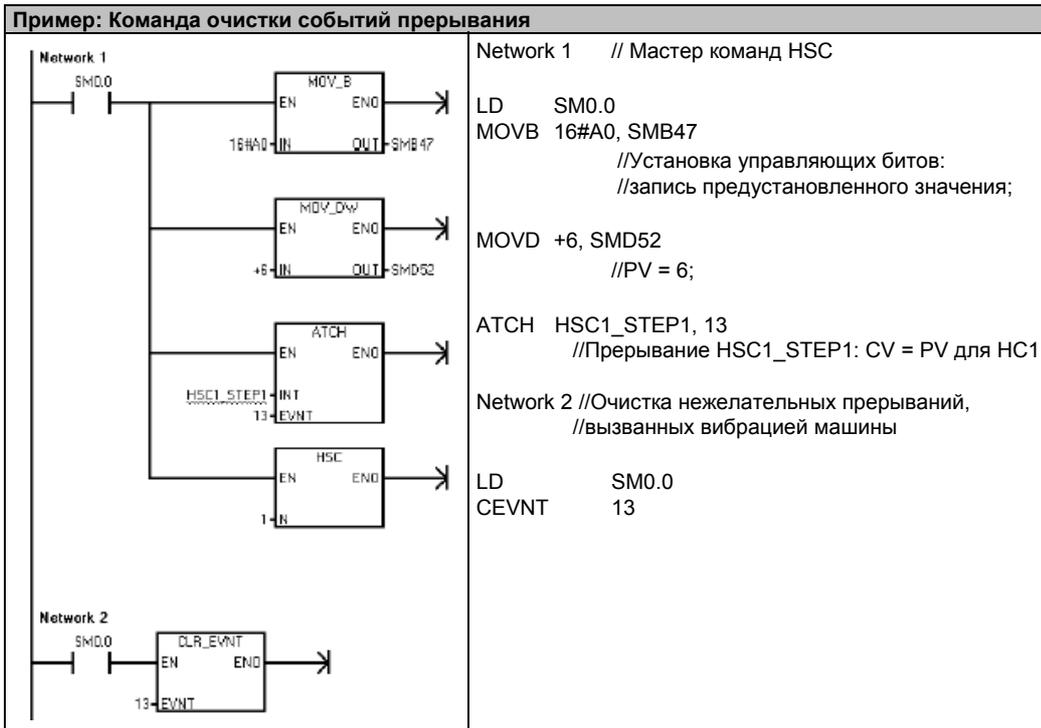
Описание (0 = нет переполнения, 1 = переполнение)	Бит специальной памяти
Очередь коммуникационных прерываний	SM4.0
Очередь прерываний по вводу/выводу	SM4.1
Очередь прерываний, управляемых временем	SM4.2

Таблица 6–50. Порядок приоритетов для прерывающих событий

Событие	Описание		Группа приоритета	Приоритет в группе
8	Порт 0	Символ принят	Коммуникации <i>Наивысший приоритет</i>	0
9	Порт 0	Передача завершена		0
23	Порт 0	Прием сообщения завершен		0
24	Порт 1	Прием сообщения завершен		1
25	Порт 1	Символ принят		1
26	Порт 1	Передача завершена		1
19	PLS0	Прерывание по завершению отсчета количества импульсов РТО	Дискретные прерывания <i>Средний приоритет</i>	0
20	PLS1	Прерывание по завершению отсчета количества импульсов РТО		1
0	I0.0	Нарастающий фронт		2
2	I0.1	Нарастающий фронт		3
4	I0.2	Нарастающий фронт		4
6	I0.3	Нарастающий фронт		5
1	I0.0	Падающий фронт		6
3	I0.1	Падающий фронт		7
5	I0.2	Падающий фронт		8
7	I0.3	Падающий фронт		9
12	HSC0	CV=PV (текущее значение = предустановленному)		10
27	HSC0	Изменение направления		11
28	HSC0	Внешний сброс		12
13	HSC1	CV=PV (текущее значение = предустановленному)		13
14	HSC1	Изменение направления		14
15	HSC1	Внешний сброс		15
16	HSC2	CV=PV (текущее значение = предустановленному)		16
17	HSC2	Изменение направления		17
18	HSC2	Внешний сброс		18
32	HSC3	CV=PV (текущее значение = предустановленному)		19
29	HSC4	CV=PV (текущее значение = предустановленному)		20
30	HSC4	Изменение направления		21
31	HSC4	Внешний сброс		22
33	HSC5	CV=PV (текущее значение = предустановленному)	23	
10	Управляемое временем прерывание 0	SMB34	Прерывания, управляемые временем <i>Низший</i>	0
11	Управляемое временем прерывание 1	SMB35		1
21	Таймер T32	Прерывание CT=PT		2
22	Таймер T96	Прерывание CT=PT		3

Пример: Команды прерывания		
M A I N	<p><b>Network 1</b></p> <p><b>Network 2</b></p> <p><b>Network 3</b></p>	<p>Network 1 //В первом цикле:                  //1. Определить программу обработки прерываний INT_0                  // как программу обработки прерыванию по падающему                  // фронту на I0.0                  //2. Разблокировать все прерывающие события.</p> <pre>LD SM0.1 ATCH INT_0, 1 ENI</pre> <p>Network 2 //Если обнаружена ошибка ввода/вывода,                  //заблокировать прерывание по падающему фронту на                  // I0.0. Этот сегмент не обязателен.</p> <pre>LD SM5.0 DTCH 1</pre> <p>Network 3 //Если M5.0 установлен,                  //заблокировать все прерывания.</p> <pre>LD M5.0 DISI</pre>
I N T 0	<p><b>Network 1</b></p>	<p>Network 1 //Программа обработки прерываний по падающему                  фронту //на I0.0:                  //Условный возврат при ошибке ввода/вывода.</p> <pre>LD SM5.0 CRETI</pre>

Пример: Прерывание, управляемое временем, для чтения значения аналогового входа		
M A I N	<p><b>Network 1</b></p>	<p>Network 1 // В первом цикле вызвать подпрограмму 0.</p> <pre>LD SM0.1 CALL SBR_0</pre>
S B R 0	<p><b>Network 1</b></p>	<p>Network 1 //1. Установить интервал для прерывания,                  // управляемого временем, от 0 до 100 мс.                  //2. Управляемое временем прерывание 0 поставить в                  // соответствие INT_0 (событие 10).                  //3. Разблокировать все прерывающие события.</p> <pre>LD SM0.0 MOV_B 100, SMB34 ATCH INT_0, 10 ENI</pre>
I N T 0	<p><b>Network 1</b></p>	<p>Network 1 //Считывать значение AIW4 каждые 100 мс</p> <pre>LD SM0.0 MOVW AIW4, VW100</pre>



## Логические операции

### Операции инвертирования

#### Инвертирование байта, слова и двойного слова

Команды инвертирования байта (INVB), слова (INW) и двойного слова (INVD) образуют дополнение входа IN до единицы и загружают результат по адресу OUT.

Сбросные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)

Биты специальной памяти, на которые действует команда:

- SM1.0 (ноль)

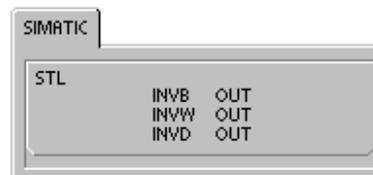
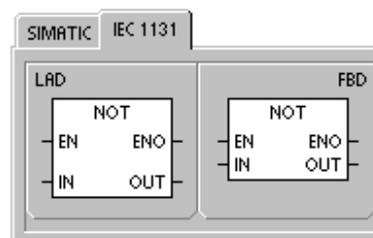
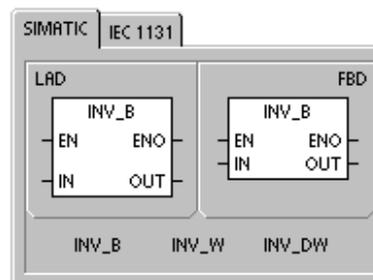


Таблица 6–51 Допустимые операнды для команд инвертирования

Входы/выходы	Типы данных	Операнды
IN	BYTE WORD DWORD	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, константа ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, константа
OUT	BYTE WORD DWORD	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

#### Пример: Операция инвертирования

<p>Network 1</p>	<p>Network 1</p> <pre>LD I4.0 INW AC0</pre> <p>Инвертирование слова</p> <p>ACO 1101 0111 1001 0101</p> <p>Формирование дополнения до единицы</p> <p>ACO 0010 1000 0110 1010</p>
------------------	---

## Поразрядные логические операции И, ИЛИ и исключающее ИЛИ

### Поразрядное логическое И с байтами, словами и двойными словами

Поразрядные логические операции И с байтами (ANDB), словами (ANDW) и двойными словами (ANDD) логически сопрягают соответствующие биты двух входных величин IN1 и IN2 в соответствии с таблицей истинности логической операции И и загружают результат по адресу OUT.

### Поразрядное логическое ИЛИ с байтами, словами и двойными словами

Поразрядные логические операции ИЛИ с байтами (ORB), словами (ORW) и двойными словами (ORD) логически сопрягают соответствующие биты двух входных величин IN1 и IN2 в соответствии с таблицей истинности логической операции ИЛИ и загружают результат по адресу OUT.

### Поразрядное логическое исключающее ИЛИ с байтами, словами и двойными словами

Поразрядные логические операции Исключающее ИЛИ с байтами (XORB), словами (XORW) и двойными словами (XORD) логически сопрягают соответствующие биты двух входных величин IN1 и IN2 в соответствии с таблицей истинности логической операции Исключающее ИЛИ и загружают результат по адресу OUT.

### Биты специальной памяти и ENO

Для всех команд, описанных на этой странице, следующие условия влияют на биты специальной памяти и ENO.

#### Сбойные состояния, устанавливающие ENO = 0

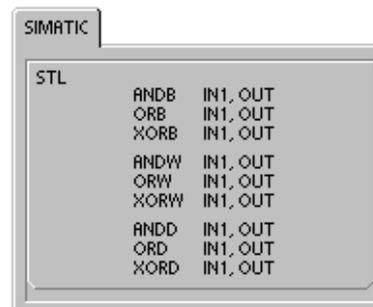
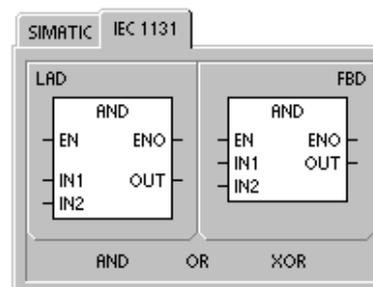
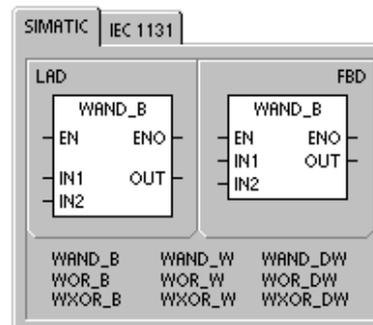
- 0006 (косвенный адрес)

#### Биты специальной памяти, на которые действует команда:

- SM1.0 (ноль)

Таблица 6–52. Допустимые операнды для операций И, ИЛИ и Исключающее ИЛИ

Входы/выходы	Типы данных	Операнды
IN1, IN2	BYTE WORD DWORD	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, константа ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, константа
OUT	BYTE WORD DWORD	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *AC, *LD ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD





## Команды пересылки

### Пересылка байта, слова, двойного слова или вещественного числа

Команды пересылки байта (MOVB), слова (MOVW), двойного слова (MOVD) и вещественного числа (MOVR) пересылают значение из адреса IN в адрес OUT, не изменяя исходной величины.

Используйте команду пересылки двойного слова для создания указателя. Подробную информацию вы найдете в разделе об указателях и косвенной адресации в главе 4.

У команды пересылки IEC на входе и выходе могут применяться различные типы данных, но они должны быть одного размера.

**Сбойные состояния, устанавливающие ENO = 0**

- 0006 (косвенный адрес)

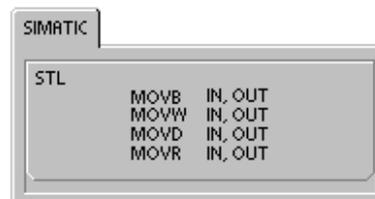
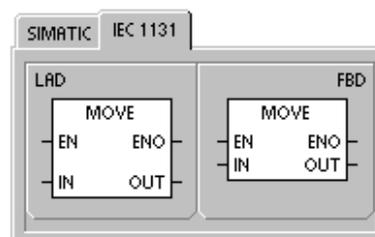
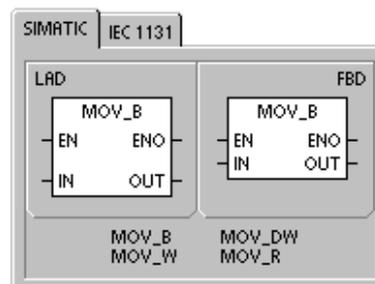


Таблица 6–53. Допустимые операнды для команд пересылки

Входы/ выходы	Типы данных	Операнды
IN	BYTE WORD, INT  DWORD, DINT  REAL	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, константа ID, QD, VD, MD, SMD, SD, LD, HC, &VB, &IB, &QB, &MB, &SB, &T, &C, &SMB, &AIW, &AQW, AC, *VD, *LD, *AC, константа, ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, константа
OUT	BYTE WORD, INT  DWORD, DINT, REAL	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AQW, *VD, *LD, *AC ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

## Непосредственное чтение или запись и пересылка байта

Команды непосредственной пересылки байта позволяют пересылать байт непосредственно между физическим входом или выходом и адресом в памяти.

Команда пересылки непосредственно считанного байта (BIR) считывает физический вход (IN) и записывает результат по адресу в памяти (OUT), не обновляя регистр образа процесса.

Команда пересылки байта для непосредственной записи (BIW) считывает данные из адреса в памяти (IN) и записывает их в физический выход (OUT) и в соответствующую ячейку образа процесса.

### Сбойные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)
- Нет доступа к модулю расширения

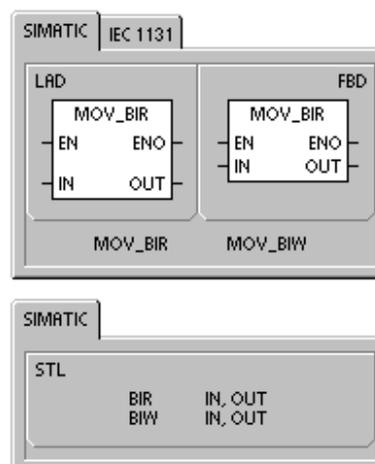


Таблица 6–54. Допустимые операнды для команды пересылки непосредственно считанного байта

Входы/выходы	Типы данных	Операнды
IN	BYTE	IB, *VD, *LD, *AC
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC

Таблица 6–55. Допустимые операнды для команды пересылки байта для непосредственной записи

Входы/выходы	Типы данных	Операнды
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа
OUT	BYTE	QB, *VD, *LD, *AC

## Команды групповой пересылки

### Групповая пересылка байтов, слов, двойных слов

Команды групповой пересылки байтов (BMB), слов (BMW) и двойных слов (BMD) пересылают указанное количество (N) байтов, слов или двойных слов, начиная с входного адреса IN, в новую область, начинающуюся с выходного адреса OUT.

N находится в диапазоне от 1 до 255.

#### Сбойные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)
- 0091 (операнд вышел за пределы допустимого диапазона)

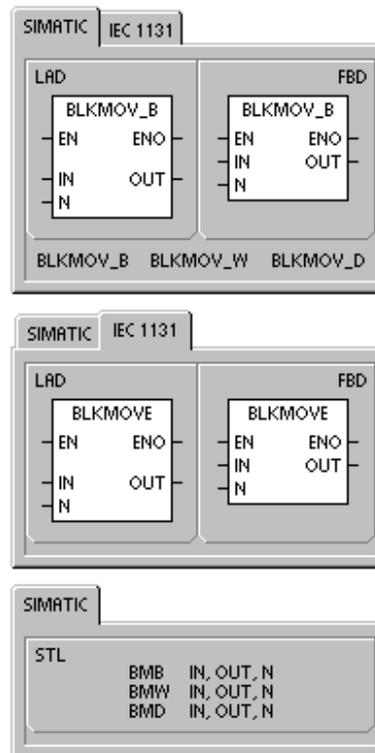


Таблица 6–56. Допустимые операнды для команд групповой пересылки

Входы/выходы	Типы данных	Операнды
IN	BYTE WORD, INT DWORD, DINT	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC IW, QW, VW, MW, SMW, SW, T, C, LW, AIW, *VD, *LD, *AC ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
OUT	BYTE WORD, INT DWORD, DINT	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC IW, QW, VW, MW, SMW, SW, T, C, LW, AQW, *VD, *LD, *AC ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, константа, *VD, *LD, *AC

**Пример: Команда групповой пересылки**

Network 1

Network 1 //Переслать массив 1 (от VB20 до VB23)  
//в массив 2 (от VB100 до VB103)

```
LD I2.1
BMB VB20, VB100, 4
```

Массив 1	VB20	VB21	VB22	VB23
	30	31	32	33
↓				
Массив 2	VB100	VB101	VB102	VB103
	30	31	32	33

## Команды управления программой

### Условное завершение

Команда условного завершения (END) завершает текущий цикл в зависимости от результата предшествующей логической операции. Вы можете использовать команду условного завершения в главной программе, но ее нельзя использовать в подпрограммах и программах обработки прерываний.

### Останов

Команда останова (STOP) завершает выполнение программы, вызывая переход CPU S7-200 из RUN в STOP.

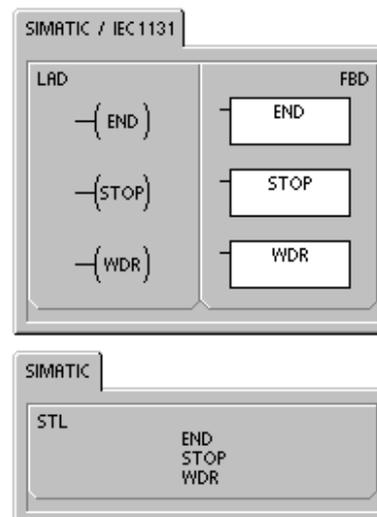
Если команда STOP выполняется в программе обработки прерывания, то эта программа завершается немедленно, а все прерывания, стоящие в очереди, игнорируются. Оставшиеся действия в текущем цикле обработки программы завершаются, включая выполнение главной программы пользователя, а переход из RUN в STOP производится в конце текущего цикла.

### Сброс контроля времени

Команда сброса контроля времени (WDR) перезапускает системный таймер контроля времени CPU S7-200, увеличивая время, которое может занимать цикл обработки программы, не вызывая ошибки контроля времени.

Команду сброса контроля времени следует использовать с осторожностью. Если вы с помощью программных циклов препятствуете завершению цикла обработки программы или существенно задерживаете его завершение, то следует иметь в виду, что до завершения цикла обработки программы запрещены следующие процессы:

- связь (за исключением режима свободно программируемого обмена данными Freepoint)
- обновление входов и выходов (кроме входов и выходов с непосредственным доступом)
- обновление принудительно задаваемых значений
- обновление битов специальной памяти (не обновляются биты SM0, SM5 – SM29)
- диагностика в режиме реального времени
- 10-миллисекундные и 100-миллисекундные таймеры не накапливают время должным образом для циклов обработки программы, превышающих 25 секунд
- команда STOP при использовании в программе обработки прерывания
- модули расширения с цифровыми выходами также имеют таймер контроля времени, который выключает выходы, если S7-200 не производит запись в этот модуль. Чтобы эти выходы оставались включенными во время продленного цикла, пользуйтесь командой непосредственной записи для каждого модуля расширения. См. также пример, следующий за этим описанием.





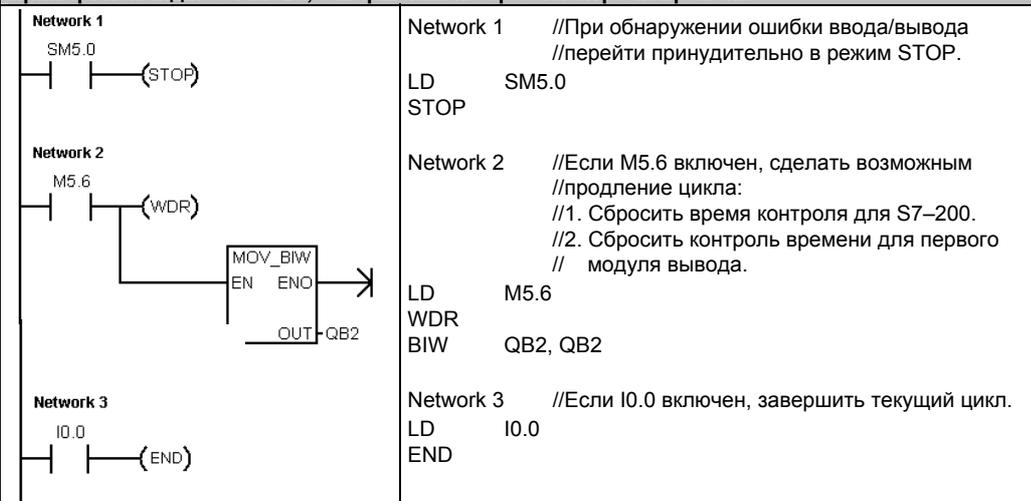
**Совет**

Команду сброса контроля времени следует использовать для перезапуска таймера контроля времени, если вы ожидаете, что время цикла обработки программы превысит 500 мс, или вы ожидаете увеличения активности прерываний, что может воспрепятствовать возвращению в главный цикл более чем на 500 мс.

Каждый раз, когда вы применяете команду сброса контроля времени, вы должны также использовать команду непосредственной записи для одного выходного байта (QB) на каждый цифровой модуль вывода, чтобы сбросить контроль времени модуля расширения.

Если вы используете команду сброса контроля времени, чтобы можно было выполнить программу с большим временем цикла, то перевод переключателя режимов работы в положение STOP вызывает переход S7-200 в состояние STOP в течение 1,4 секунды.

**Пример: Команды останова, завершения и сброса контроля времени**



## Команды формирования программного цикла For-Next

С помощью команд FOR и NEXT вы можете управлять программными циклами, которые повторяются определенное количество раз. Каждая команда FOR требует наличия команды NEXT. Вы можете вкладывать циклы FOR/NEXT друг в друга (помещать цикл FOR-NEXT внутри другого цикла FOR-NEXT). Глубина вложения не может превышать восьми.

Команда FOR выполняет команды, расположенные между операторами FOR и NEXT. Вы должны задать значение индекса или счетчик цикла INDX, начальное значение INIT и конечное значение FINAL.

Команда NEXT отмечает конец цикла FOR.

### Сбойные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)

Если вы активизируете цикл FOR-NEXT, то процесс циклического выполнения продолжается, пока не закончатся итерации, если только вы не измените конечное значение изнутри самого цикла. Вы можете изменять эти значения, пока цикл FOR-NEXT выполняет циклическую обработку. Когда цикл активизируется снова, он копирует начальное значение в индекс (счетчик цикла).

Команда FOR-NEXT сбрасывает себя каждый раз, когда она активизируется.

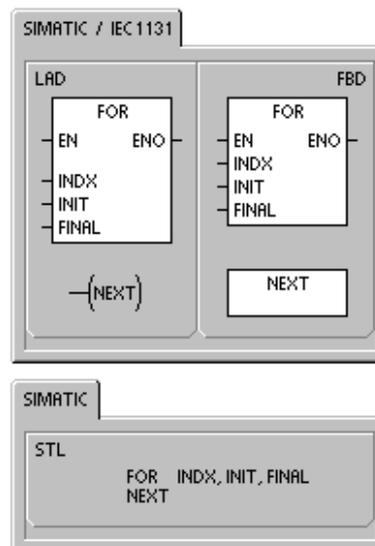
Например, если значение INIT равно 1, значение FINAL равно 10, то команды между FOR и NEXT исполняются 10 раз, причем значение INDX каждый раз увеличивается на единицу: 1, 2, 3, ...10.

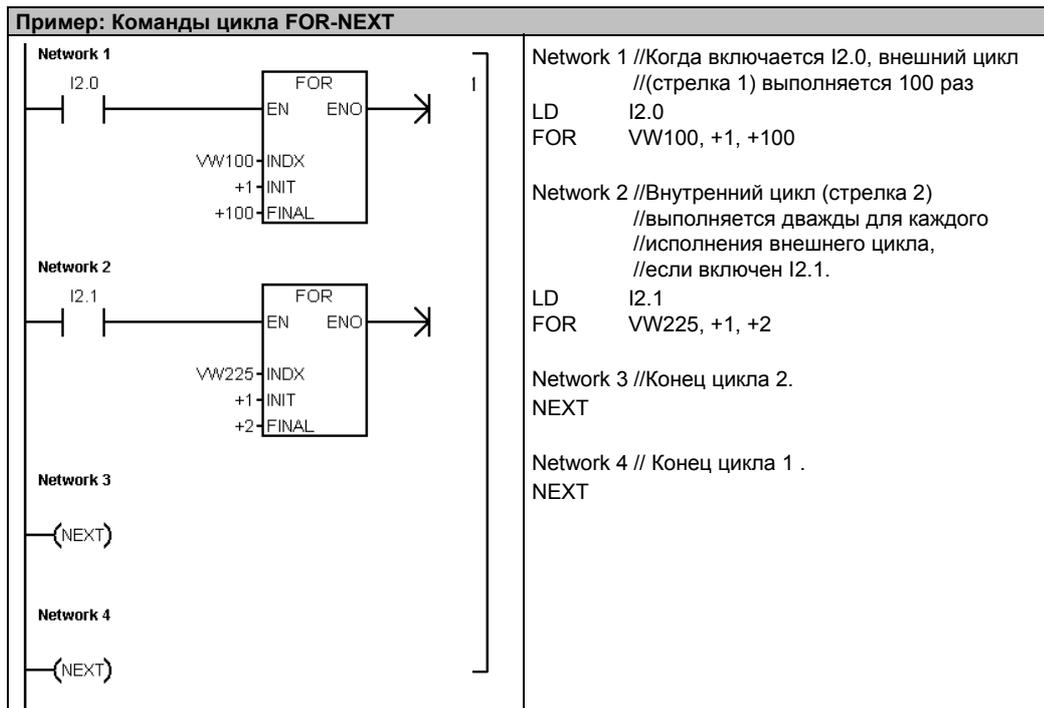
Если начальное значение больше конечного, то цикл не выполняется. После каждого исполнения команд между FOR и NEXT значение INDX увеличивается, а результат сравнивается с конечным значением. Если INDX больше конечного значения, то цикл завершается.

Если вершина стека равна 1, когда ваша программа входит в цикл FOR-NEXT, то вершина стека останется равной 1, когда ваша программа покинет цикл FOR-NEXT.

Таблица 6–57. Допустимые операнды для команд FOR и NEXT

Входы/выходы	Типы данных	Операнды
INDX	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
INIT, FINAL	INT	VW, IW, QW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, константа





## Команды перехода

Команда перехода на метку (JMP) осуществляет переход к указанной метке N внутри программы.

Команда "Метка" (LBL) отмечает положение цели перехода N.

Команду перехода на метку можно использовать в основной программе, в подпрограммах и в программах обработки прерываний. Команда перехода и соответствующая метка всегда должны находиться внутри одного и того же сегмента кода (в основной программе, подпрограмме или программе обработки прерываний).

Вы не можете перейти из главной программы на метку в подпрограмме или в программе обработки прерывания. Аналогично, вы не можете перейти из подпрограммы или программы обработки прерывания на метку вне этой подпрограммы или программы обработки прерывания.

Команду перехода на метку можно использовать внутри сегмента SCR, но соответствующая метка должна находиться внутри того же сегмента SCR.

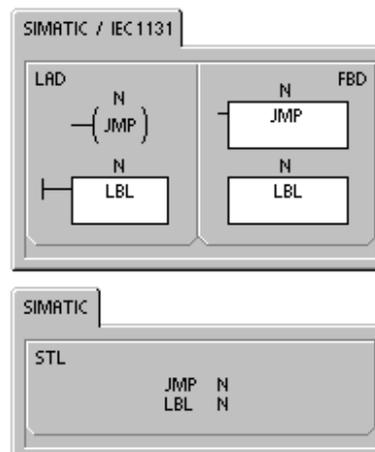


Таблица 6–58. Допустимые операнды для команд перехода

Входы/выходы	Типы данных	Операнды
N	WORD	константа (от 0 до 255)

Пример: Команда перехода на метку	
<p><b>Network 1</b></p> <p><b>Network 2</b></p>	<p>Network 1 //Если сохраняемые данные не потеряны, перейти на //метку LBL 4</p> <p>LDN SM0.2</p> <p>JMP 4</p> <p>Network 2</p> <p>LBL 4</p>

## Команды для реле управления очередностью (SCR)

Команды управления очередностью (SCR) предоставляют в ваше распоряжение простое и вместе с тем мощное средство для программирования пошагового управления, которое естественно встраивается в программу LAD, FBD или STL.

Если ваше приложение состоит из последовательности функций, которые должны неоднократно выполняться, то с помощью реле управления очередностью, вы можете так структурировать свою программу, что она будет непосредственно соответствовать вашему приложению. В результате вам будет проще и быстрее программировать и тестировать ваше приложение.

Команда загрузки реле управления очередностью (LSCR) загружает стек SCR и логический стек значением S-бита, на который указывает команда N.

Сегмент SCR активизируется или деактивируется результирующим значением стека SCRT. Значение стека SCR копируется в вершину логического стека, так что блоки или выходные катушки могут быть непосредственно связаны с левой силовой шиной без промежуточных контактов.

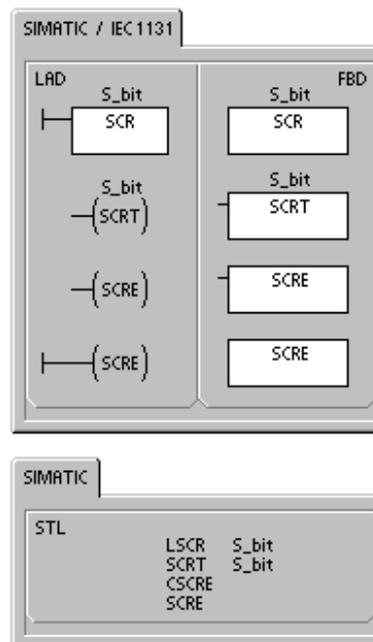
### Ограничения

Используя SCR, примите во внимание следующие ограничения:

- Нельзя использовать один и тот же S-бит более чем в одной программе. Например, если вы используете S0.1 в главной программе, не используйте его в подпрограмме.
- Нельзя использовать команды JMP и LBL для перехода в сегмент SCR или из него; но вы можете использовать эти команды для перепрыгивания через сегменты SCR или для переходов внутри сегмента SCR.
- В сегменте SCR нельзя использовать команду END.

Таблица 6–59. Допустимые операнды для команд реле управления очередностью

Входы/выходы	Типы данных	Операнды
S_bit	BOOL	S



На рис. 6–32 показаны S-стек и логический стек и воздействие команды загрузки SCR. Для команд реле управления очередностью имеет силу следующее:

- Команда загрузки реле управления очередностью (LSCR) отмечает начало сегмента SCR, а команда завершения реле управления очередностью (SCRE) отмечает конец сегмента SCR. Исполнение всех операций между командами LSCR и SCRE зависит от значения S-стека. Исполнение операций между SCRE и следующей командой LSCR не зависит от значения S-стека.

- Команда перехода к следующему реле управления очередностью (SCRT) предоставляет средство для передачи управления от активного сегмента SCR другому сегменту SCR.

Команда перехода к следующему реле управления очередностью выполняется, когда ее достигает поток сигнала. Она сбрасывает S-бит активного в данный момент сегмента и устанавливает S-бит указанного сегмента. Сброс S-бита активного сегмента не оказывает влияния на S-стек на время выполнения команды перехода к следующему реле. Поэтому сегмент SCR остается активизированным, пока он не будет покинут.

Загрузить значение Sx.y в стек SCR и логический стек.

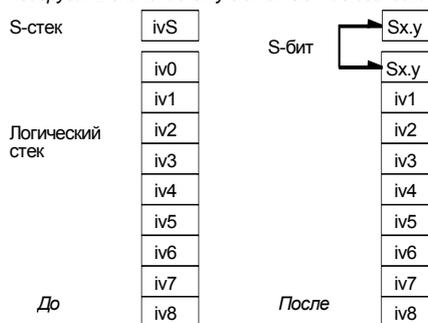
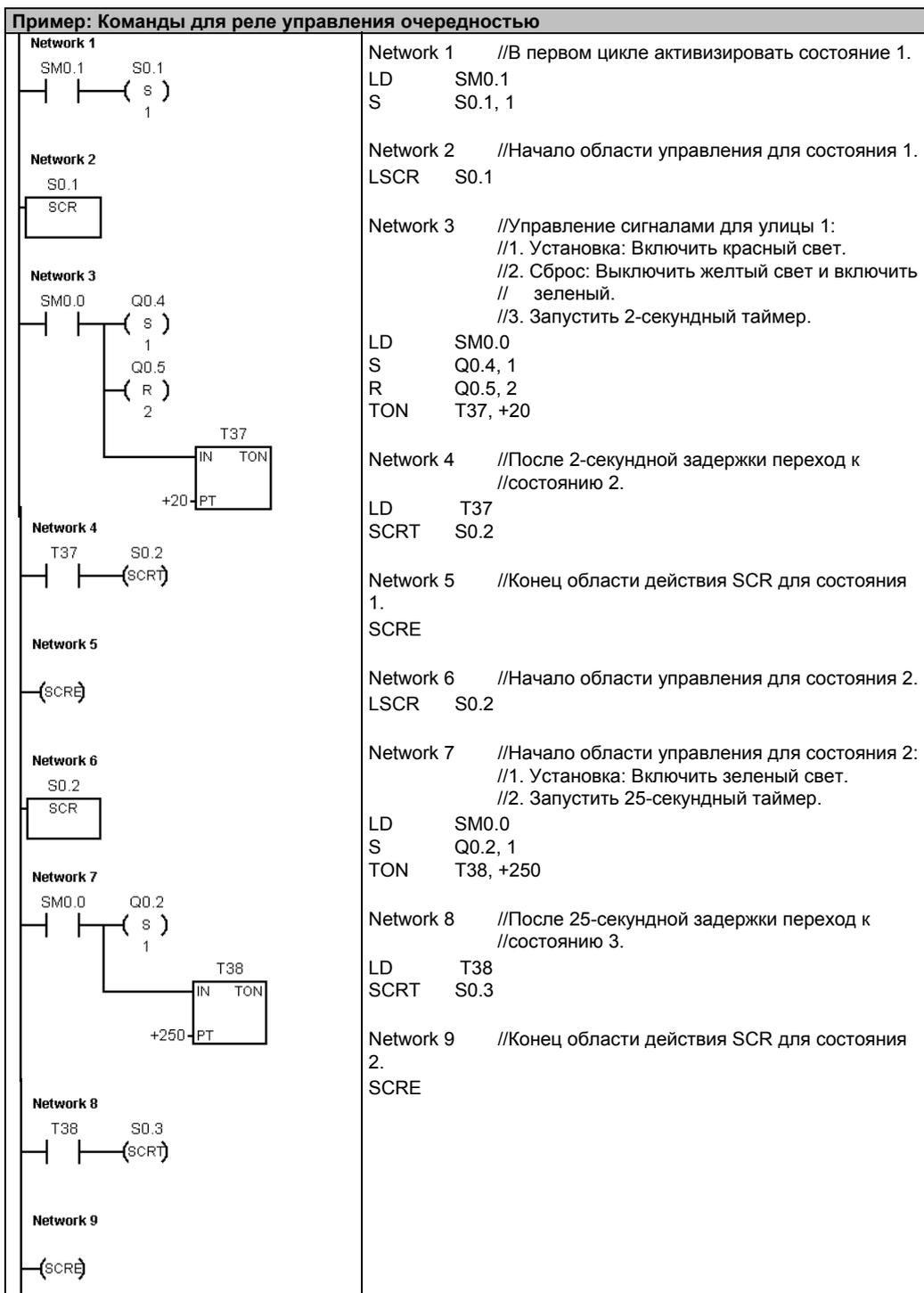


Рис. 6–32. Влияние LSCR на логический стек

- Команда условного завершения SCR (CSCRE) предоставляет средство для выхода из активного сегмента SCR без выполнения команд, расположенных между командой условного завершения SCR и командой завершения реле управления очередностью. Команда условного завершения SCR не оказывает влияния ни на S-бит, ни на S-стек.

В следующем примере бит первого цикла SM0.1 устанавливает S0.1, который в первом цикле будет в активном состоянии 1. После 2-секундной задержки T37 вызывает переход к состоянию 2. Этот переход деактивизирует сегмент SCR для состояния 1 (S0.1) и активизирует сегмент SCR для состояния 2 (S0.2).



### Разделение потоков управления

Во многих приложениях единый поток последовательных состояний должен быть разделен на два или более различных потоков. Когда поток управления разделяется на несколько потоков, все выходящие потоки должны активизироваться одновременно. Это показано на рис. 6–33.

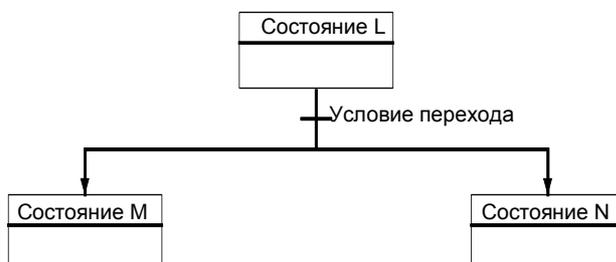


Рис. 6–33. Разделение потока управления

Разделение потоков управления может быть реализовано в программе SCR путем использования нескольких команд SCRT, активизируемых одним и тем же условием перехода, как показано в следующем примере.

**Пример: Разделение потоков управления**

<p><b>Network 1</b></p> <p>S3.4 SCR</p> <p><b>Network 2</b></p> <p>M2.3    I2.1    S3.5 (SCRT) S6.5 (SCRT)</p> <p><b>Network 3</b></p> <p>SCRE</p>	<p>Network 1    //Начало области управления для состояния L. LSCR    S3.4</p> <p>Network 2</p> <p>LD    M2.3 A    I2.1 SCRT    S3.5    //Переход к состоянию M SCRT    S6.5    //Переход к состоянию N</p> <p>Network 3    //Конец области SCR для состояния L. SCRE</p>
--	--

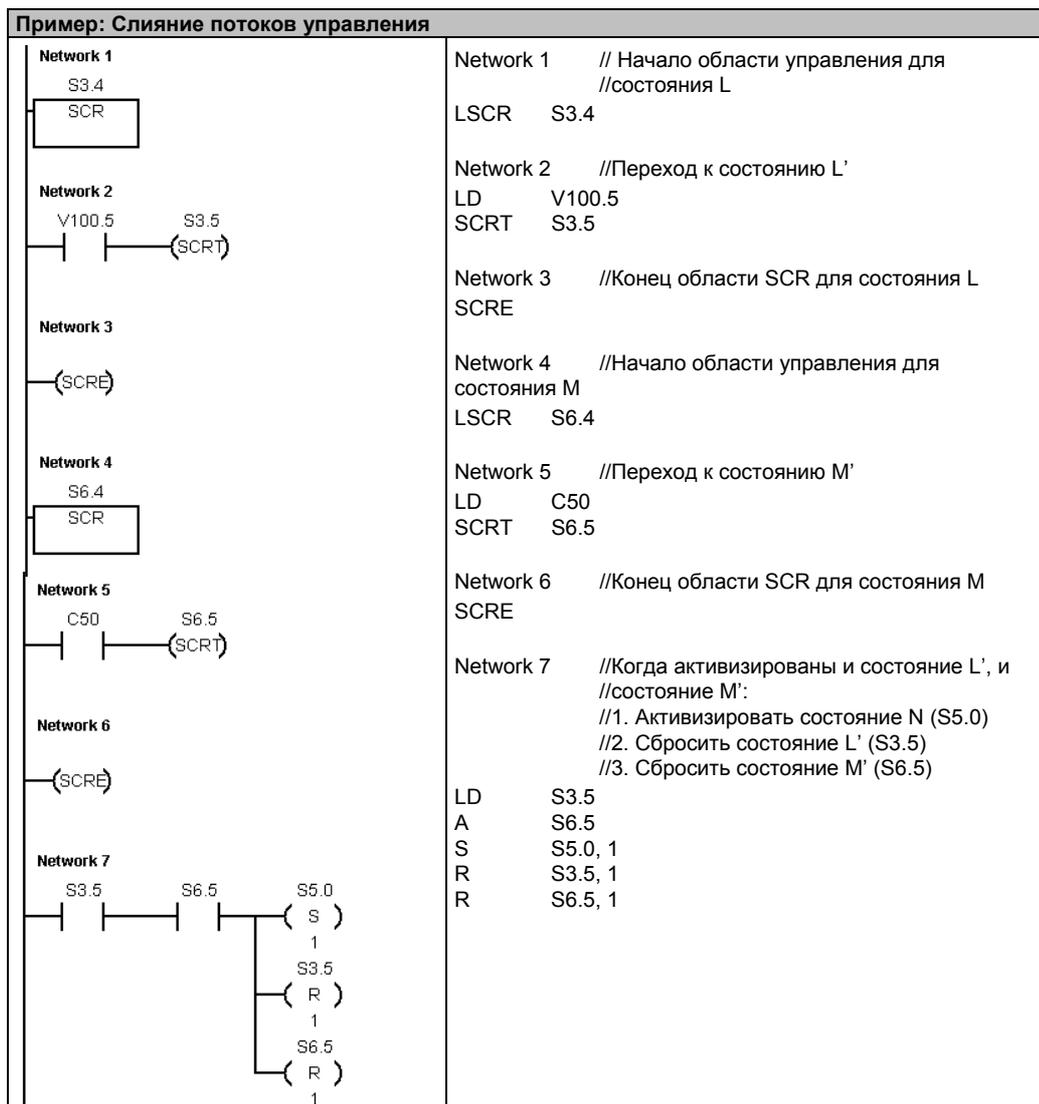
### Слияние потоков управления

Ситуация, аналогичная разделению потоков управления, возникает, когда два или более потоков последовательных состояний должны быть объединены в один поток. Когда несколько потоков объединяются в один поток, говорят, что они сливаются. Когда несколько потоков сливаются в один поток, все входящие потоки должны быть завершены до того, как будет выполняться следующее состояние. На рис. 6–34 изображено слияние двух потоков управления.

Слияние потоков управления может быть реализовано в программе SCR путем создания перехода от состояния L к состоянию L' и перехода от состояния M к состоянию M'. Когда биты SCR, представляющие L' и M', установлены, состояние N может быть активизировано, как показано в следующем примере.



Рис. 6–34. Слияние потоков управления



В других ситуациях поток управления может быть направлен в один из нескольких возможных потоков управления в зависимости от того, какое условие перехода выполнилось первым. Такая ситуация изображена на рис. 6–35, где представлена эквивалентная программа SCR.



Рис. 6–35. Разделение потока управления в зависимости от условия перехода

Пример: Условные переходы	
<b>Network 1</b> S3.4 SCR	Network 1 //Начало области управления для состояния L LSCR S3.4
<b>Network 2</b> M2.3 S3.5     (SCRT)	Network 2 //Переход к состоянию M LD M2.3 SCRT S3.5
<b>Network 3</b> I3.3 S6.5     (SCRT)	Network 3 //Переход к состоянию N LD I3.3 SCRT S6.5
<b>Network 4</b> (SCRE)	Network 4 //Конец области SCR для состояния L SCRE

## Команда управления диагностическим светодиодом

Если входной параметр IN имеет значение ноль, то диагностический светодиод выключается. Если входной параметр IN имеет значение больше нуля, то диагностический светодиод включается (желтый).

Светодиод CPU, обозначенный SF/ DIAG, может быть сконфигурирован таким образом, что он загорается желтым светом, когда выполняются условия, указанные в системном блоке данных или когда выполняется команда DIAG\_LED с параметром IN, отличным от нуля.

Триггерные кнопки для системного блока данных (конфигурирование светодиода):

- светодиод SF/ DIAG включен (желтый), если некоторый элемент в CPU управляется принудительно
- светодиод SF/ DIAG LED включен (желтый), если в модуле имеет место ошибка ввода-вывода

Деактивизируйте обе опции для такого конфигурирования светодиода, чтобы желтым светом светодиода управляла только команда DIAG\_LED. Системная ошибка CPU (SF) отображается красным светом.

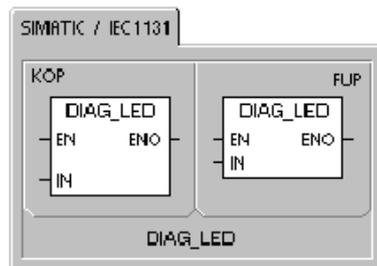
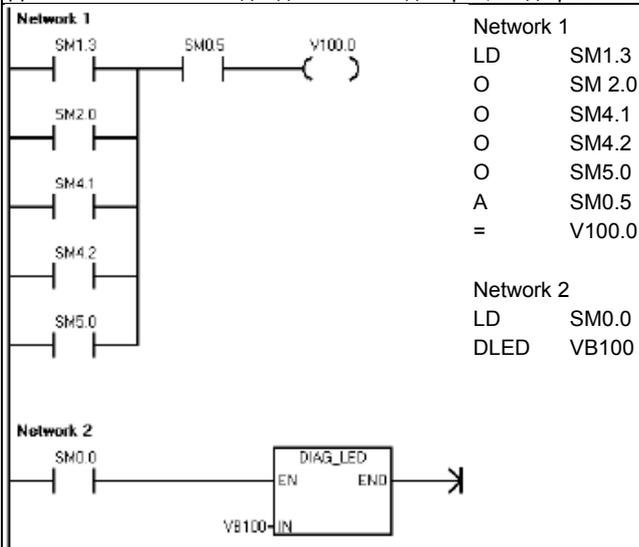


Таблица 6–60. Допустимые операнды для команды управления диагностическим светодиодом

Входы/выходы	Типы данных	Операнды
IN	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC, константа, *VD, *LD, *AC

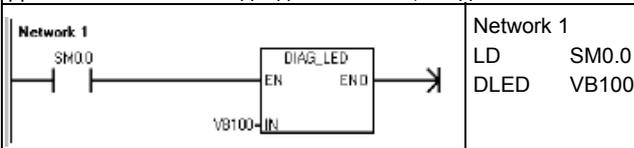
### Пример 1 для команды управления диагностическим светодиодом

Диагностический светодиод мигает при обнаружении ошибки.  
Диагностический светодиод мигает каждый раз, когда распознается одна из 5 сбойных ситуаций.



### Пример 1 для команды управления диагностическим светодиодом

Диагностический светодиод включается, когда появляется ошибка.  
Диагностический светодиод включается, когда в VB100 появляется код ошибки



## Команды сдвига и циклического сдвига Marker 35

### Команды сдвига вправо и сдвига влево

Команды сдвига сдвигают входную величину IN вправо или влево на число разрядов, указанное в N, и загружают результат в выход OUT.

Команды сдвига заполняют позиции выдвигаемых битов нулями. Если величина сдвига (N) больше или равна максимально допустимому значению (8 для операций с байтами, 16 для операций со словами и 32 для операций с двойными словами), то сдвиг производится на максимально возможную величину для данной операции. Если величина сдвига больше 0, то бит переполнения (SM1.1) принимает значение последнего выдвинутого бита. Бит нулевого значения (SM1.0) устанавливается, если результат операции сдвига равен нулю.

Байтовые операции являются беззнаковыми. Для операций со словами и двойными словами знаковый бит сдвигается, если вы используете типы данных со знаком.

**Сбойные состояния, устанавливающие ENO = 0**

- 0006 (косвенный адрес)

**Биты специальной памяти, на которые действует команда:**

- SM1.0 (ноль)
- SM1.1 (переполнение)

### Команды циклического сдвига вправо и циклического сдвига влево

Команды циклического сдвига циклически сдвигают входную величину (IN) вправо или влево на число разрядов, указанное в (N), и загружают результат по адресу (OUT). Циклический сдвиг является кольцевым.

Если величина сдвига больше или равна максимально допустимому значению (8 для операций с байтами, 16 для операций со словами и 32 для операций с двойными словами), то S7-200 выполняет операцию по модулю с величиной сдвига (получение остатка от деления заданного сдвига на максимально допустимый), чтобы получить действительную величину циклического сдвига. Результатом является величина сдвига от 0 до 7 для операций с байтами, от 0 до 15 для операций со словами и от 0 до 31 для операций с двойными словами.

Если величина сдвига равна нулю, то циклический сдвиг не производится. Если циклический сдвиг выполняется, то значение последнего циклически сдвинутого бита копируется в бит переполнения (SM1.1).

Если величина сдвига не является целым кратным 8 (для операций с байтами), 16 (для операций со словами) или 32 (для операций с двойными словами), то последний циклически выдвинутый бит копируется в бит переполнения (SM1.1). Бит нулевого значения (SM1.0) устанавливается, если подлежащая циклическому сдвигу величина равна нулю.

Операции с байтами являются беззнаковыми. Для операций со словами и двойными словами знаковый бит сдвигается, если вы используете типы данных со знаком.

**Сбойные состояния, устанавливающие ENO = 0**

- 0006 (косвенный адрес)

**Биты специальной памяти, на которые действует команда:**

- SM1.0 (ноль)
- SM1.1 (переполнение)

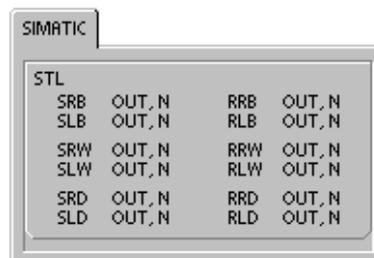
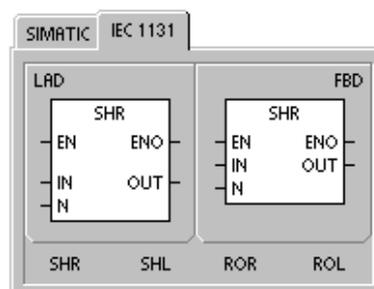
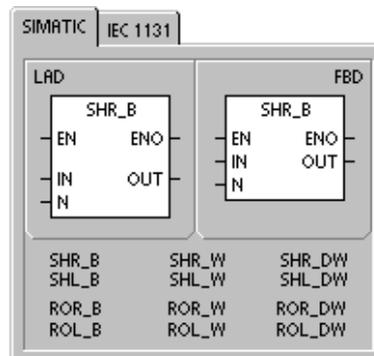


Таблица 6–61. Допустимые операнды для команд сдвига и циклического сдвига

Входы/выходы	Типы данных	Операнды
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, константа
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, константа
OUT	BYTE	OB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа

**Пример: Команды сдвига и циклического сдвига**

<p><b>Network 1</b></p>	<p><b>Network 1</b></p> <pre>LD    I4.0 RRW   AC0, 2 SLW   VW200, 3</pre>
<p><b>Циклический сдвиг</b></p> <p>Перед циклическим сдвигом: AC0 = 0100 0000 0000 0001</p> <p>После 1-го сдвига: AC0 = 1010 0000 0000 0000 (Переполнение = 1)</p> <p>После 2-го сдвига: AC0 = 0101 0000 0000 0000 (Переполнение = 0)</p> <p>Бит нулевого результата (SM1.0) = 0 Бит переполнения (SM1.1) = 0</p>	<p><b>Сдвиг</b></p> <p>Перед сдвигом: VW200 = 1110 0010 1010 1101</p> <p>После 1-го сдвига: VW200 = 1100 0101 0101 1010 (Переполнение = 1)</p> <p>После 2-го сдвига: VW200 = 1000 1010 1011 0100 (Переполнение = 1)</p> <p>После 3-го сдвига: VW200 = 0001 0101 0110 1000 (Переполнение = 1)</p> <p>Бит нулевого результата (SM1.0) = 0 Бит переполнения (SM1.1) = 1</p>

## Команда перемещения значения бита в регистр сдвига Marker 36

Команда перемещения значения бита в регистр сдвига перемещает значение в регистр сдвига. Эта команда предоставляет в распоряжение простой метод упорядочения и управления потоком изделий или данных. С помощью этой команды вы можете сдвигать весь регистр на один бит каждый цикл.

Команда перемещения значения бита в регистр сдвига вдвигает значение DATA в регистр сдвига. S\_BIT указывает на младший бит регистра сдвига. N задает длину регистра сдвига и направление сдвига (положительный сдвиг = N, отрицательный сдвиг = -N).

Каждый бит, выдвинутый из регистра командой SHRB, помещается в бит переполнения (SM1.1).

Эта команда определяется младшим битом регистра сдвига (S\_BIT) и количеством битов, задаваемых длиной (N).

### Сбойные состояния, устанавливающие ENO = 0

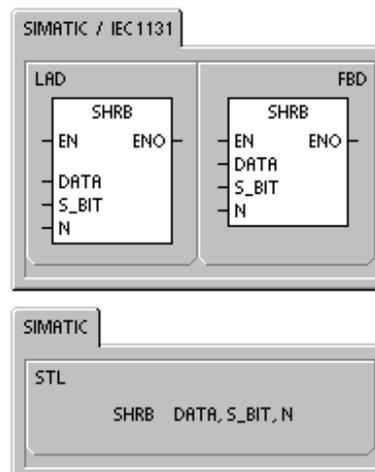
- 0006 (косвенный адрес)
- 0091 (операнд вышел за пределы допустимого диапазона)
- 0092 (ошибка в поле счета)

### Биты специальной памяти, на которые действует команда:

- SM1.1 (переполнение)

Таблица 6–62. Допустимые операнды для команды перемещения значения бита в регистр сдвига

Входы/выходы	Типы данных	Операнды
DATA, S_Bit	BOOL	I, Q, V, M, SM, S, T, C, L
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа



Адрес старшего бита регистра сдвига (MSB.b) можно рассчитать с помощью следующего уравнения:

$$MSB.b = [(\text{байт параметра } S\_BIT) + ([N] - 1 + (\text{бит параметра } S\_BIT)) / 8] \cdot [\text{остаток от деления на } 8]$$

Например: если S\_BIT – это V33.4, и N = 14, то следующий расчет показывает, что MSB.b – это V35.1.

$$\begin{aligned} MSB.b &= V33 + ([14] - 1 + 4) / 8 \\ &= V33 + 17 / 8 \\ &= V33 + 2 \text{ с остатком } 1 \\ &= V35.1 \end{aligned}$$

При отрицательном сдвиге, который задается отрицательным значением длины (N), входные данные (DATA) вдвигаются на место старшего бита регистра сдвига и выдвигают из регистра младший бит (S\_BIT). Данные, выдвинутые из регистра, помещаются в бит переполнения (SM1.1).

При положительном сдвиге, который задается положительным значением длины (N), входные данные (DATA) вдвигаются на место младшего бита регистра сдвига, указываемого параметром S\_BIT, и выдвигают из регистра сдвига старший бит. Данные, выдвинутые из регистра, помещаются в бит переполнения (SM1.1).

Максимальная длина регистра сдвига (положительная или отрицательная) равна 64 битам. На рис. 6–36 показан сдвиг битов при отрицательном и положительном значении N.

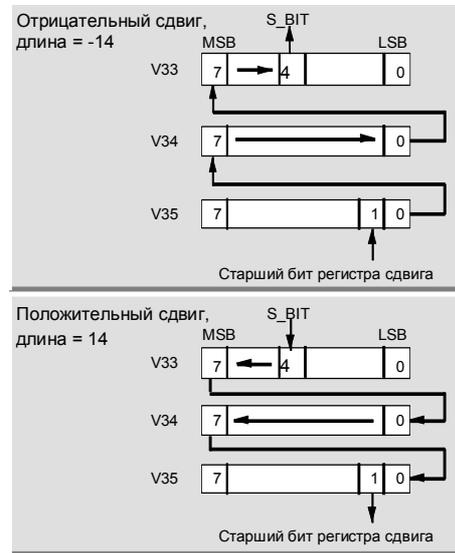


Рис. 6–36. Ввод и удаление данных из регистра сдвига

**Пример: Команда перемещения значения бита в регистр сдвига**

<p>Network 1</p>	<p>Network 1</p> <pre>LD I0.2 EU SHRB I0.3, V100.0, +4</pre>															
<p>Временная диаграмма</p>	<p>7 (MSB) 0 (LSB) S_BIT</p> <p>Перед 1-м сдвигом V100 <table border="1"><tr><td>0</td><td>1</td><td>0</td><td>1</td></tr></table> ← I0.3</p> <p>Переполнение (SM1.1) <table border="1"><tr><td>x</td></tr></table> ← S_BIT</p> <p>После 1-го сдвига V100 <table border="1"><tr><td>1</td><td>0</td><td>1</td><td>1</td></tr></table> ← I0.3</p> <p>Переполнение (SM1.1) <table border="1"><tr><td>0</td></tr></table> ← S_BIT</p> <p>После 2-го сдвига V100 <table border="1"><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr></table> ← I0.3</p> <p>Переполнение (SM1.1) <table border="1"><tr><td>1</td></tr></table> ← S_BIT</p>	0	1	0	1	x	1	0	1	1	0	0	1	1	0	1
0	1	0	1													
x																
1	0	1	1													
0																
0	1	1	0													
1																

### Команда перестановки байтов в слове

Команда перестановки байтов меняет местами старший и младший байты слова IN.

Сбросные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)

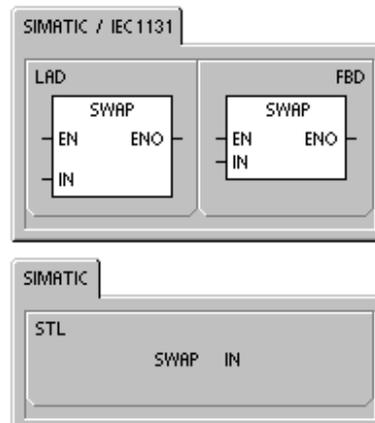
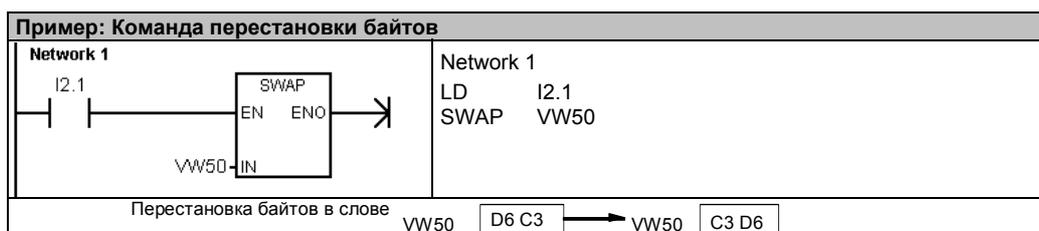


Таблица 6–63. Допустимые операнды для команды перестановки байтов в слове

Входы/выходы	Типы данных	Операнды
IN	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC



## Строковые команды Marker 38

### Длина строки

Команда определения длины строки (SLEN) возвращает длину строки, указанной в IN.

### Копирование строки

Команда копирования строки (SCPY) копирует строку, указанную в IN, в строку, указанную в OUT.

### Соединение строк

Команда соединения строк (SCAT) присоединяет строку, указанную в IN, к концу строки, указанной в OUT.

### Биты специальной памяти и ENO

Для команд определения длины строки, копирования строки и соединения строк следующие условия влияют на ENO.

**Сбойные состояния, устанавливающие ENO = 0**

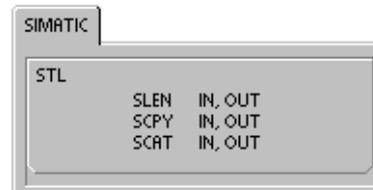
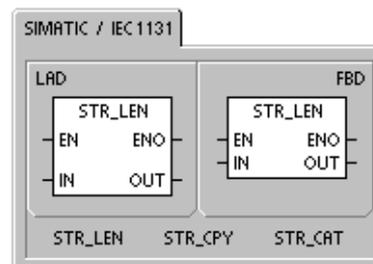
- 0006 (косвенный адрес)
- 0091 (ошибка диапазона)

Таблица 6–64. Допустимые операнды для команды определения длины строки

Входы/выходы	Типы данных	Операнды
IN	STRING	VB, LB, *VD, *LD, *AC, строковая константа
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC

Таблица 6–65. Допустимые операнды для команд копирования строки и соединения строк

Входы/выходы	Типы данных	Операнды
IN	STRING	VB, LB, *VD, *LD, *AC, строковая константа
OUT	STRING	VB, LB, *VD, *AC, *LD





## Копирование подстроки из строки

Команда копирования подстроки из строки (SSCPY) копирует указанное количество символов N из строки, указанной в IN, начиная с индекса INDX, в новую строку, указанную OUT.

### Сбойные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)
- 0091 (ошибка диапазона)
- 009B (индекс=0)

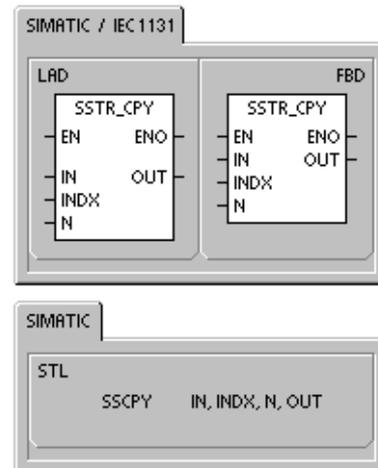


Таблица 6–66. Допустимые операнды для команды копирования подстроки из строки

Входы/выходы	Типы данных	Операнды
IN	STRING	VB, LB, *VD, *LD, *AC, строковая константа
OUT	STRING	VB, LB, *VD, *LD, *AC
INDX, N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа

### Пример: Команда копирования подстроки

<p><b>Network 1</b></p>	<p><b>Network 1</b> //Начиная с седьмого символа в строке, //находящейся по адресу VB0, скопировать //5 символов в новую строку по адресу</p> <p>VB20 LD I0.0 SSCPY VB0, 7, 5, VB20</p>												
<p>Перед выполнением программы</p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 10%;">VB0</td> <td>'H'</td><td>'E'</td><td>'L'</td><td>'L'</td><td>'O'</td><td>' '</td><td>'W'</td><td>'O'</td><td>'R'</td><td>'L'</td><td>'D'</td> </tr> </table>		VB0	'H'	'E'	'L'	'L'	'O'	' '	'W'	'O'	'R'	'L'	'D'
VB0	'H'	'E'	'L'	'L'	'O'	' '	'W'	'O'	'R'	'L'	'D'		
<p>После выполнения программы</p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 10%;">VB20</td> <td>5</td><td>'W'</td><td>'O'</td><td>'R'</td><td>'L'</td><td>'D'</td> </tr> </table>		VB20	5	'W'	'O'	'R'	'L'	'D'					
VB20	5	'W'	'O'	'R'	'L'	'D'							

## Поиск строки в строке

Команда поиска строки в строке (SFND) ищет первое появление строки IN2 в строке IN1. Поиск начинается с позиции, указанной в OUT. Если найдена последовательность символов, точно совпадающая со строкой IN2, то позиция первого символа в этой последовательности записывается в OUT. Если строка IN2 не найдена в строке IN1, то OUT устанавливается в 0.

### Сбойные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)
- 0091 (ошибка диапазона)
- 009B (индекс=0)

## Поиск первого появления символа в строке

Команда поиска первого появления символа в строке (CFND) ищет в строке IN1 первое появление любого символа из набора символов, описываемого в строке IN2. Поиск начинается с позиции, указанной в OUT. Если совпадающий символ найден, то его положение записывается в OUT. Если совпадающего символа не найдено, то OUT устанавливается в 0.

### Сбойные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)
- 0091 (ошибка диапазона)
- 009B (индекс=0)

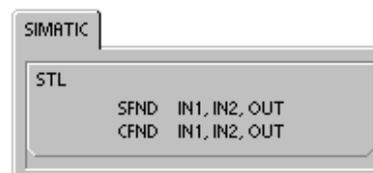
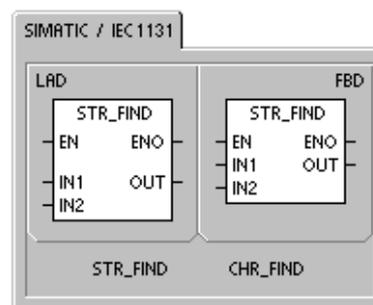


Таблица 6–67. Допустимые операнды для команд поиска строки в строке и первого символа в строке

Входы/выходы	Типы данных	Операнды
IN1, IN2	STRING	VB, LB, *VD, *LD, *AC, строковая константа
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC

**Пример: Команда поиска строки в строке**

В следующем примере строка, хранящаяся по адресу VB0, используется как команда для включения или выключения насоса. Строка 'On [Включить]' хранится по адресу VB20, а строка 'Off [Выключить]' хранится по адресу VB30. Результат команды поиска строки в строке сохраняется в AC0 (параметр OUT). Если результат не равен 0, то строка 'On' была найдена в командной строке (VB12).

Network 1 //1. Установить AC0 в 1.  
 // (AC0 используется как параметр OUT.)  
 //2. Искать в строке, находящейся по адресу VB0, строку, находящуюся по адресу VB20 ('On [Включить]'), начиная с первой позиции (AC0=1).

```
LD I0.0
MOV_B 1, AC0
STR_FIND VB0, VB20, AC0
```

VB0	12	'T'	'u'	'r'	'h'	''	'P'	'u'	'm'	'p'	''	'O'	'n'	VB12
VB20	2	'O'	'n'	VB22		VB30	3	'O'	'f'	'f'	VB33			

Если строка из VB20 найдена:

AC0

11

Если строка из VB20 не найдена:

AC0

0

**Пример: Команда поиска первого появления символа в строке**

В следующем примере строка, хранящаяся по адресу VB0, содержит значение температуры. Строка, хранящаяся по адресу VB20, хранит все цифровые символы (в том числе + и -), которые могут указывать на температуру в строке. Данная программа-пример находит начальное положение числа в этой строке, а затем преобразует цифровые символы в вещественное число. VD200 хранит значение температуры в виде вещественного числа.

Network 1 //1. Установить AC0 в 1.  
 // (AC0 используется в качестве параметра OUT и указывает на 1-ю позицию строки.)  
 //2. Найти цифровой символ в строке, находящейся по адресу VB0.  
 //3. Преобразовать строку в вещественное число.

```
LD I0.0
MOV_B 1, AC0
CFND VB0, VB20, AC0
STR VB0, AC0, VD200
```

VB0	11	'T'	'e'	'm'	'p'	''	''	'9'	'8'	'.'	'6'	'F'	VB11	
VB20	12	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'	'0'	'+'	'-'	VB32

Начальная позиция температуры, хранящейся в VB0:

AC0

7

Значение температуры в виде вещественного числа:

VD200

98.6

## Табличные команды

### Ввод значения в таблицу

Команда ввода значений в таблицу вносит в таблицу (TBL) значения, имеющие размер слова (DATA). Первым значением таблицы является ее максимальная длина (TL). Второе значение – это количество записей в таблице (EC). Новые данные добавляются к таблице после последней записи. Каждый раз, когда к таблице добавляются новые данные, количество записей увеличивается на единицу.

Таблица может иметь до 100 записей.

#### Сбойные состояния, устанавливающие ENO = 0

- SM1.4 (переполнение таблицы)
- 0006 (косвенный адрес)
- 0091 (операнд вышел за пределы допустимого диапазона)

#### Биты специальной памяти, на которые действует команда:

- SM1.4 устанавливается в 1, если вы пытаетесь переполнить таблицу

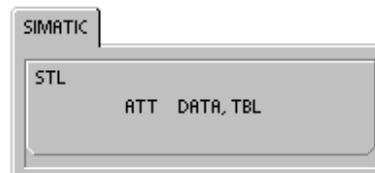
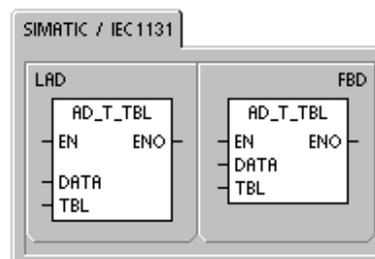


Таблица 6–68. Допустимые операнды для табличных команд

Входы/выходы	Типы данных	Операнды
DATA	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, константа
TBL	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, *VD, *LD, *AC

**Пример: Команда ввода значения в таблицу**

**Network 1**

**Network 2**

**Network 1** //Загрузить максимальную длину таблицы

```
LD SM0.1
MOVW +6, VW200
```

**Network 2**

```
LD I0.0
ATT VW100, VW200
```

Перед выполнением ATT

VW100	1234
VW200	0006
VW202	0002
VW204	5431
VW206	8942
VW208	xxxx
VW210	xxxx
VW212	xxxx
VW214	xxxx

После выполнения ATT

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

## Удаление первой записи из таблицы и удаление последней записи из таблицы

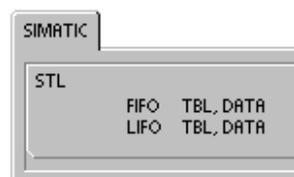
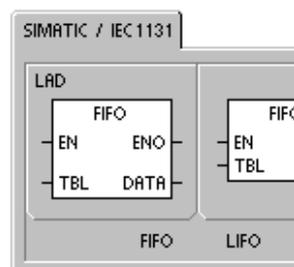
Таблица может иметь до 100 записей.

### Удаление первой записи

Команда удаления первой записи (FIFO) удаляет первую (самую старую) запись из таблицы (TBL) и пересылает значение по адресу, указанному в DATA. Все остальные записи таблицы смещаются на одну позицию вверх. Количество записей в таблице после выполнения каждой команды уменьшается на 1.

### Удаление последней записи

Команда удаления последней записи (LIFO) удаляет последнюю (самую новую) запись из таблицы (TBL) и пересылает значение по адресу, указанному в DATA. Количество записей в таблице после выполнения каждой команды уменьшается на 1.



#### Сбойные состояния, устанавливающие ENO = 0

- SM1.5 (пустая таблица)
- 0006 (косвенный адрес)
- 0091 (операнд вышел за пределы допустимого диапазона)

#### Биты специальной памяти, на которые действует команда:

- SM1.5 устанавливается в 1, если вы пытаетесь удалить запись из пустой таблицы

Таблица 6–69. Допустимые операнды для команд удаления первой и последней записи

Входы/выходы	Типы данных	Операнды
TBL	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, *VD, *LD, *AC
DATA	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AQW, *VD, *LD, *AC

**Пример: Команда удаления первой записи**

Network 1

Network 1

```
LD I4.1
FIFO VW200, VW400
```

Перед выполнением FIFO		После выполнения FIFO	
VW200	0006	VW200	0006
VW202	0003	VW202	0002
VW204	5431	VW204	8942
VW206	8942	VW206	1234
VW208	1234	VW208	xxxx
VW210	xxxx	VW210	xxxx
VW212	xxxx	VW212	xxxx
VW214	xxxx	VW214	xxxx

TL (макс. число записей)

EC (число записей)

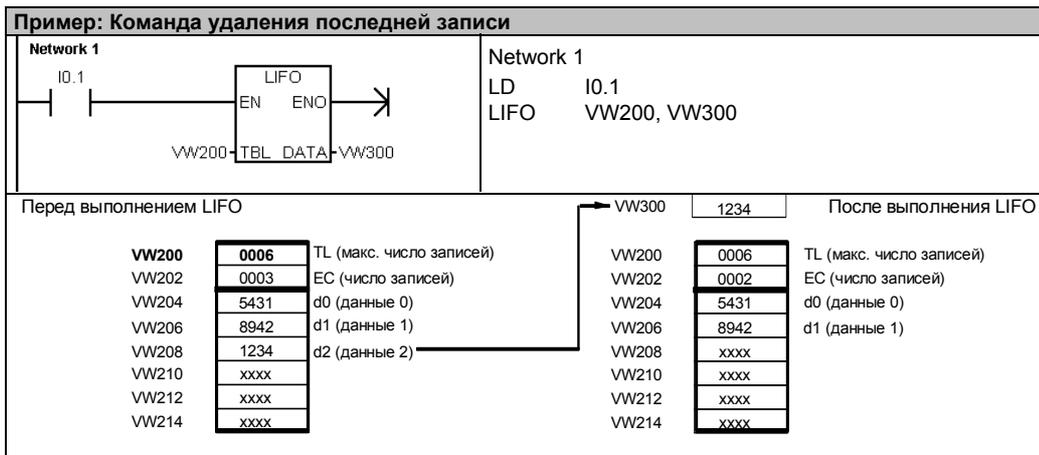
d0 (данные 0)

d1 (данные 1)

d2 (данные 2)

VW400

5431



## Заполнение памяти комбинацией двоичных разрядов

Команда заполнения памяти (FILL) записывает N последовательных слов, начиная с адреса OUT, значением слова, содержащегося по адресу IN.

N имеет диапазон от 1 до 255.

**Сбойные состояния, устанавливающие ENO = 0**

- 0006 (косвенный адрес)
- 0091 (операнд вышел за пределы допустимого диапазона)

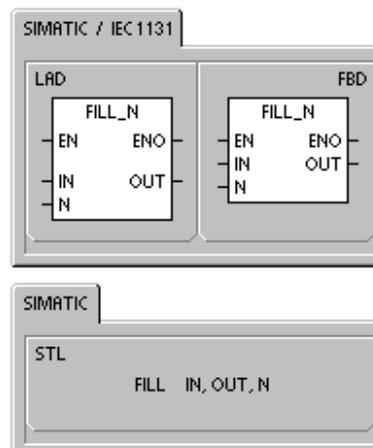


Таблица 6–70. Допустимые операнды для команды заполнения памяти

Входы/выходы	Типы данных	Операнды
IN	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, константа
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, константа
OUT	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AQW, *VD, *LD, *AC



## Поиск значения в таблице

Команда поиска значения в таблице (FND) ищет в таблице данные, удовлетворяющие определенному критерию. Команда поиска значения в таблице ищет в таблице TBL, начиная с записи INDX, значение или образец данных PTN, удовлетворяющее критерию поиска, определенному CMD. Командному параметру CMD дается числовое значение от 1 до 4, что соответственно означает =, <>, < и >.

Если найдена запись, удовлетворяющая критерию поиска, то INDX указывает на эту запись в таблице. Для нахождения следующей записи, удовлетворяющей критерию поиска, то INDX должен быть увеличен, прежде чем команда Поиск в таблице будет вызвана снова. Если запись, удовлетворяющая критерию поиска, не найдена, то INDX имеет значение, равное количеству записей.

Таблица может иметь до 100 записей. Записи данных (область поиска) нумеруются от 0 до максимального значения, равного 99.

### Сбойные состояния, устанавливающие ENO = 0

- 0006 (косвенный адрес)
- 0091 (операнд вышел за пределы допустимого диапазона)

Таблица 6–71. Допустимые операнды для команды поиска значения в таблице

Входы/выходы	Типы данных	Операнды
TBL	WORD	IW, QW, VW, MW, SMW, T, C, LW, *VD, *LD, *AC
PTN	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, константа
INDX	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
CMD	BYTE	(константа) 1: равно (=), 2: не равно (<>), 3: меньше (<), 4: больше (>)



### Совет

Когда вы используете команду поиска в таблицах, созданных с помощью команд ввода значения в таблицу (ATT), удаления первой записи (LIFO) и удаления последней записи (FIFO), то количество записей и сами записи данных согласуются непосредственно. Слово, определяющее максимальное количество записей, требуемое для команд ATT, LIFO и FIFO, не требуется для команды поиска. См. рис. 6–37.

Поэтому вы должны установить операнд TBL команды поиска в таблице на один адрес слова (два байта) выше, чем операнд TBL соответствующей команды ATT, LIFO или FIFO.

#### Формат таблицы для ATT, LIFO и FIFO

VW200	0006	TL (макс. кол-во записей)
VW202	0006	ЕС (число записей)
VW204	xxxx	d0 (данные 0)
VW206	xxxx	d1 (данные 1)
VW208	xxxx	d2 (данные 2)
VW210	xxxx	d3 (данные 3)
VW212	xxxx	d4 (данные 4)
VW214	xxxx	d5 (данные 5)

#### Формат таблицы для TBL\_FIND

VW202	0006	ЕС (число записей)
VW204	xxxx	d0 (данные 0)
VW206	xxxx	d1 (данные 1)
VW208	xxxx	d2 (данные 2)
VW210	xxxx	d3 (данные 3)
VW212	xxxx	d4 (данные 4)
VW214	xxxx	d5 (данные 5)

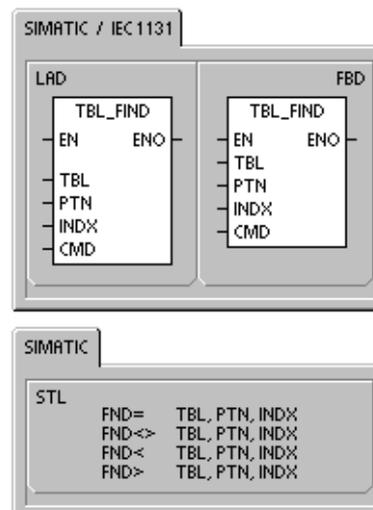


Рис. 6–37. Различия в форматах таблиц у команд "Поиск в таблице" и команд ATT, LIFO и FIFO

**Пример: Команда поиска значения в таблице**

**Network 1**

**Network 1**

LD I2.1  
FND= VW202, 16#3130, AC1

Если I2.1 включен, то в таблице ищется число, равное 3130 HEX.

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

Если таблица была создана командой ATT, LIFO или FIFO, то VW200 содержит максимально разрешенное количество записей и не требуется для команд поиска.

AC1  Для поиска с самой верхней записи в таблице AC1 должен быть установлен в 0

Поиск в таблице

AC1  AC1 содержит номер записи данных, которая соответствует критерию поиска.

AC1  Перед поиском остальных записей в таблице увеличить INDX на 1

Поиск в таблице

AC1  AC1 содержит номер записи данных, которая соответствует второму совпадению, найденному в таблице (d4)

AC1  Перед поиском остальных записей в таблице увеличить INDX на 1.

Поиск в таблице

AC1  AC1 содержит значение, равное количеству записей. Вся таблица была просмотрена в безуспешных поисках еще одного совпадения.

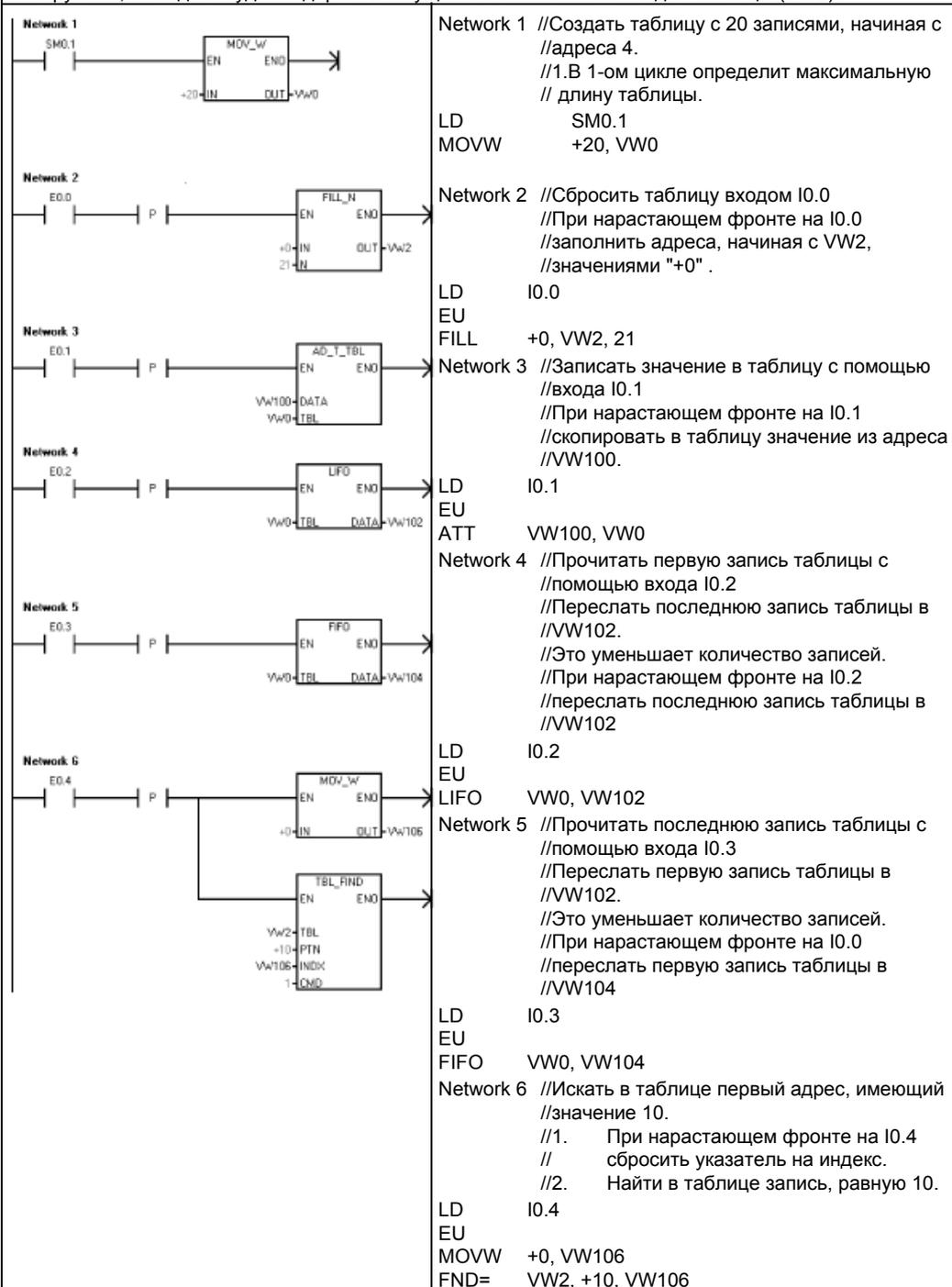
AC1  Перед новым поиском в таблице INDX должен быть установлен в 0.

**Пример: Создание таблицы**

Следующая программа создает таблицу с 20 записями. Первый адрес таблицы содержит длину таблицы (в данном случае 20 записей). Второй адрес показывает текущее количество записей в таблице. Остальные адреса содержат записи. Таблица может иметь до 100 записей. Сюда не входят параметры, определяющие максимальную длину таблицы и текущее количество записей (здесь VW0 и VW2). Текущее количество записей в таблице (здесь VW2) CPU автоматически увеличивает или уменьшает на 1 при выполнении каждой команды.

Перед началом работы с таблицей назначьте максимальное количество записей таблицы. В противном случае вы не сможете вводить записи в таблицу. Кроме того, обеспечьте, чтобы все команды чтения и записи активизировались с помощью фронтов.

Для поиска в таблице индекс (VW106) перед выполнением поиска должен быть установлен в 0. Если найдено совпадение, индекс будет указывать номер соответствующей записи, а если совпадение не обнаружено, то индекс будет содержать текущее количество записей для таблицы (VW2).



## Таймерные команды

### Таймерные команды SIMATIC

#### Таймер с задержкой включения Таймер с задержкой включения с запоминанием

Команды "Таймер с задержкой включения" (TON) и "Таймер с задержкой включения с запоминанием" (TONR) отсчитывают время, когда включен разрешающий вход. Номер таймера (Тхх) определяет его разрешающую способность, и эта разрешающая способность теперь отображается в блоке команды.

#### Таймер с задержкой выключения

Таймер с задержкой выключения (TOF) используется для задержки выключения выхода на фиксированный интервал времени после выключения входа. Номер таймера (Тхх) определяет его разрешающую способность, и эта разрешающая способность теперь отображается в блоке команды.

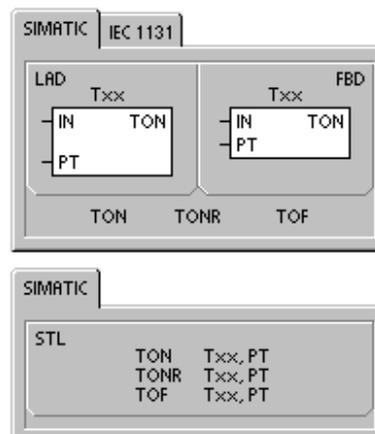


Таблица 6–72. Допустимые операнды для таймерных команд SIMATIC

Входы/выходы	Типы данных	Операнды
Txx	WORD	константа (от T0 до T255)
IN	BOOL	I, Q, V, M, SM, S, T, C, L, поток сигнала
PT	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, константа



#### Совет

Таймерам с задержкой выключения (TOF) и с задержкой включения (TON) нельзя задавать один и тот же номер (Тхх). Например, у вас не могут одновременно использоваться TON T32 и TOF T32.

Как показано в таблице 6–73, эти три вида таймеров выполняют различные задачи измерения времени:

- Таймер с задержкой включения TON может использоваться для отсчета отдельного интервала.
- Таймер с задержкой включения с запоминанием TONR может использоваться для накопления нескольких отсчитанных интервалов времени.
- Таймер с задержкой выключения TOF может использоваться для увеличения интервала времени после выключения (или сбоя), например, для охлаждения двигателя после его отключения.

Таблица 6–73. Действия таймеров

Тип	Текущее время >= предустановленному	Состояние разрешающего входа (IN)	Выключение-включение питания / первый цикл
TON	Бит таймера установлен Отсчет текущего значения продолжается до 32 767	ON: Текущее значение отсчитывает время OFF: Бит таймера сброшен, текущее значение = 0	Бит таймера сброшен Текущее значение = 0
TONR	Бит таймера установлен Отсчет текущего значения продолжается до 32 767	ON: Текущее значение отсчитывает время OFF: Бит таймера и текущее значение сохраняют последнее состояние	Бит таймера сброшен Текущее значение может быть сохранено <sup>1</sup>
TOF	Бит таймера сброшен Текущее время = предустановленному, отсчет времени прекращен	ON: Бит таймера установлен, текущее значение = 0 OFF: Таймер выполняет отсчет времени после перехода из включенного состояния в выключенное	Бит таймера сброшен Текущее значение = 0

<sup>1</sup> Текущее значение таймера с запоминанием может быть выбрано в качестве сохраняемого значения на время выключения питания. Информацию о сохранении памяти для CPU S7–200 вы найдете в главе 4.



Советы по  
программированию

Советы по программированию на компакт-диске с документацией содержат пример программы, использующей таймер с задержкой включения (TON). См. Совет 31

Команды TON и TONR отсчитывают время, когда включен разрешающий вход. Когда текущее значение становится больше или равно предустановленному времени, бит таймера устанавливается.

- Текущее значение таймера TON сбрасывается, когда выключается разрешающий вход, тогда как текущее значение таймера TONR сохраняется, когда этот вход выключается.
- Вы можете использовать таймер TONR для накопления времени, когда этот вход включается и выключается. Для стирания текущего значения TONR используется команда Сброс (R).
- Таймеры TON и TONR продолжают счет после достижения предустановленного значения, они останавливают счет при достижении максимального значения, равного 32767.

Команда TOF используется для задержки выключения выхода на фиксированный интервал времени после выключения входа. Когда включается разрешающий вход, немедленно включается бит таймера, а текущее значение устанавливается в 0. Когда вход выключается, таймер ведет отсчет времени, пока истекшее время не достигнет предустановленного значения.

- Когда предустановленное время достигнуто, бит таймера сбрасывается, а отсчет текущего значения прекращается; однако, если вход включается снова, прежде чем TOF достигнет предустановленного значения, то бит таймера остается установленным.
- Чтобы таймер TOF начал отсчет времени, к его разрешающему входу должен быть приложен падающий фронт.
- Если таймер TOF находится внутри области SCR, и область SCR не активна, то текущее значение устанавливается в 0, бит таймера выключается, и отсчет текущего значения не ведется.



#### Совет

Таймер TONR может быть сброшен только командой сброса (R). Команду сброса можно использовать также для сброса любого таймера TON или TOF. Команда сброса выполняет следующие операции:

- Бит таймера сбрасывается
- Текущее значение таймера = 0

После сброса таймеры TOF требуют для своего перезапуска появления отрицательного фронта на разрешающем входе.

## Определение разрешающей способности таймера

Таймеры отсчитывают интервалы времени. Разрешающая способность (или база времени) таймера определяет промежуток времени на один интервал. Например, TON с разрешающей способностью 10 мс отсчитывает количество 10-миллисекундных интервалов, прошедших после активизации TON: отсчет 50 на 10-миллисекундном таймере представляет 500 мс. Имеются таймеры SIMATIC с тремя разрешающими способностями: 1 мс, 10 мс и 100 мс. Как показано в таблице 6-74, номер таймера определяет его разрешающую способность.

	<p><b>Совет</b></p> <p>Чтобы гарантировать минимальный интервал времени, увеличьте предустановленное значение (PV) на 1. Например: чтобы гарантировать минимальный интервал времени не менее 2100 мс для 100–миллисекундного таймера, установите PV на 22.</p>
---	--

Таблица 6–74. Номера таймеров и разрешения

Тип таймера	Разрешение	Максимальное значение	Номер таймера
TONR (с запоминанием)	1 мс	32,767 с (0,546 мин.)	T0, T64
	10 мс	327,67 с (5,46 мин.)	T1 – T4, T65 – T68
	100 мс	3276,7 с (54,6 мин.)	T5 – T31, T69 – T95
TON, TOF (без запоминания)	1 мс	32,767 с (0,546 мин.)	T32, T96
	10 мс	327,67 с (5,46 мин.)	T33 – T36, T97 – T100
	100 мс	3276,7 с (54,6 мин.)	T37 – T63, T101 – T255

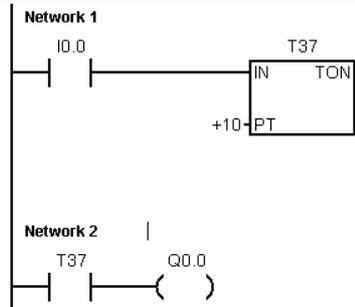
### Влияние разрешающей способности на работу таймера

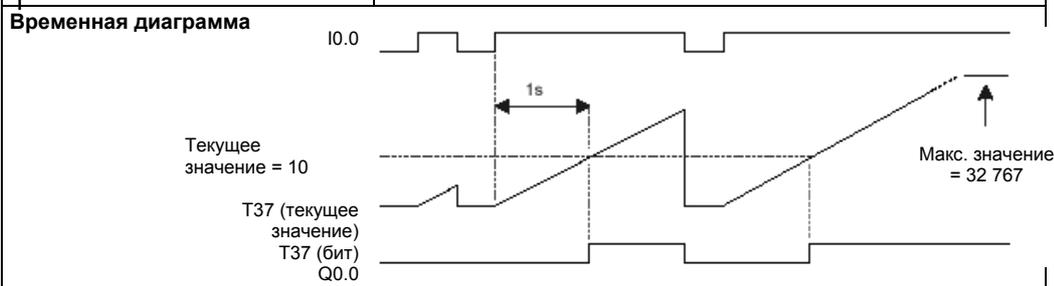
У таймера с разрешающей способностью 1 мс бит таймера и текущее значение обновляются асинхронно с циклом обработки программы. Для циклов, превышающих 1 мс, бит таймера и текущее значение обновляются несколько раз в течение цикла.

У таймера с разрешающей способностью 10 мс бит таймера и текущее значение обновляются в начале каждого цикла обработки программы. Текущее значение и бит таймера остаются неизменными на протяжении цикла, и интервалы времени, накопленные в течение цикла, добавляются к текущему значению в начале каждого цикла обработки программы.

У таймера с разрешающей способностью 100 мс бит таймера и текущее значение обновляются, когда выполняется команда; поэтому, чтобы таймер сохранял правильное значение времени, обратите внимание на то, чтобы ваша программа выполняла команду для 100–миллисекундного таймера только один раз за цикл.

#### Пример: Таймер SIMATIC с задержкой включения

	<p>Network 1 //100-миллисекундный таймер T37 заканчивает //отсчет времени по истечении (10 x 100 мс = 1 с) //I0.0 установлен = активизировать T37, //I0.0 сброшен = деактивизировать и сбросить T37</p> <p>LD I0.0 TON T37, +10</p> <p>Network 2 //Бит T37 управляется таймером T37</p> <p>LD T37 = Q0.0</p>
---	--

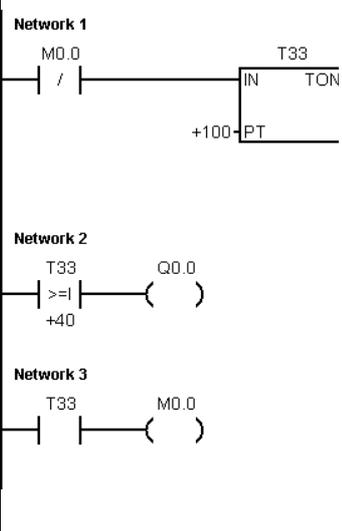




**Совет**

Чтобы гарантировать, что выход самосбрасывающегося таймера включался на время одного цикла каждый раз, когда таймер достигает предустановленного значения, вы должны использовать на разрешающем входе размыкающий контакт вместо бита таймера.

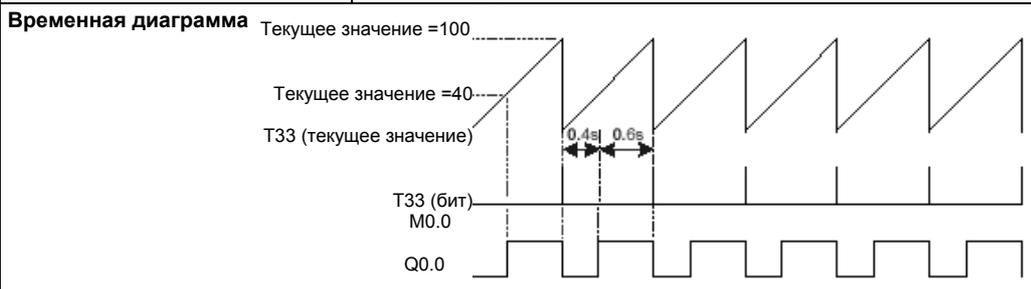
**Пример: Самосбрасывающийся таймер SIMATIC с задержкой включения**



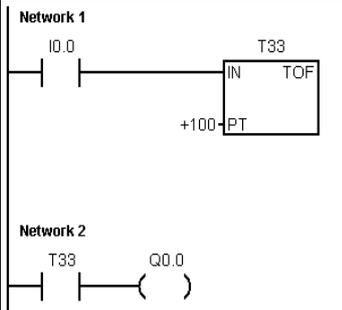
Network 1 //10-миллисекундный таймер T33 заканчивает отсчет //времени по истечении (100 x 10 мс = 1 с) //Импульс M0.0 слишком быстр для наблюдения в //представлении статуса программы  
LDN M0.0  
TON T33, +100

Network 2 //Сравнение осуществляется с темпом, позволяющим //наблюдать его результат в представлении статуса //программы. Включать Q0.0 через (40 x 10 мс) для //получения сигнала 40% выкл./60% вкл.  
LDW>= T33, +40  
= Q0.0

Network 3 //Импульс T33 (бит) слишком быстр для наблюдения в //представлении статуса программы. //Сбросить таймер с помощью M0.0 по истечении //периода (100 x 10 мс)  
LD T33  
= M0.0

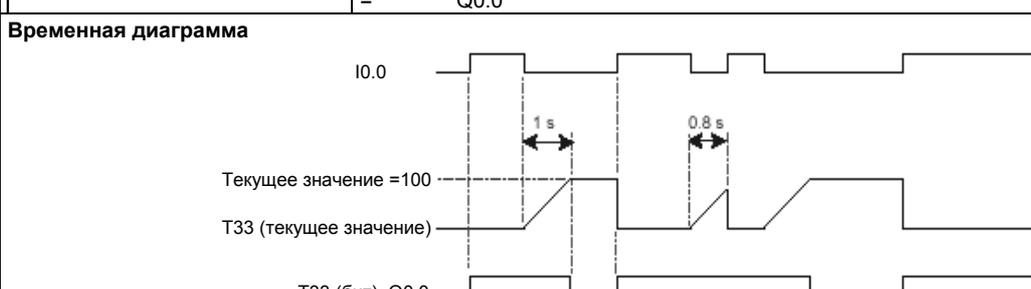


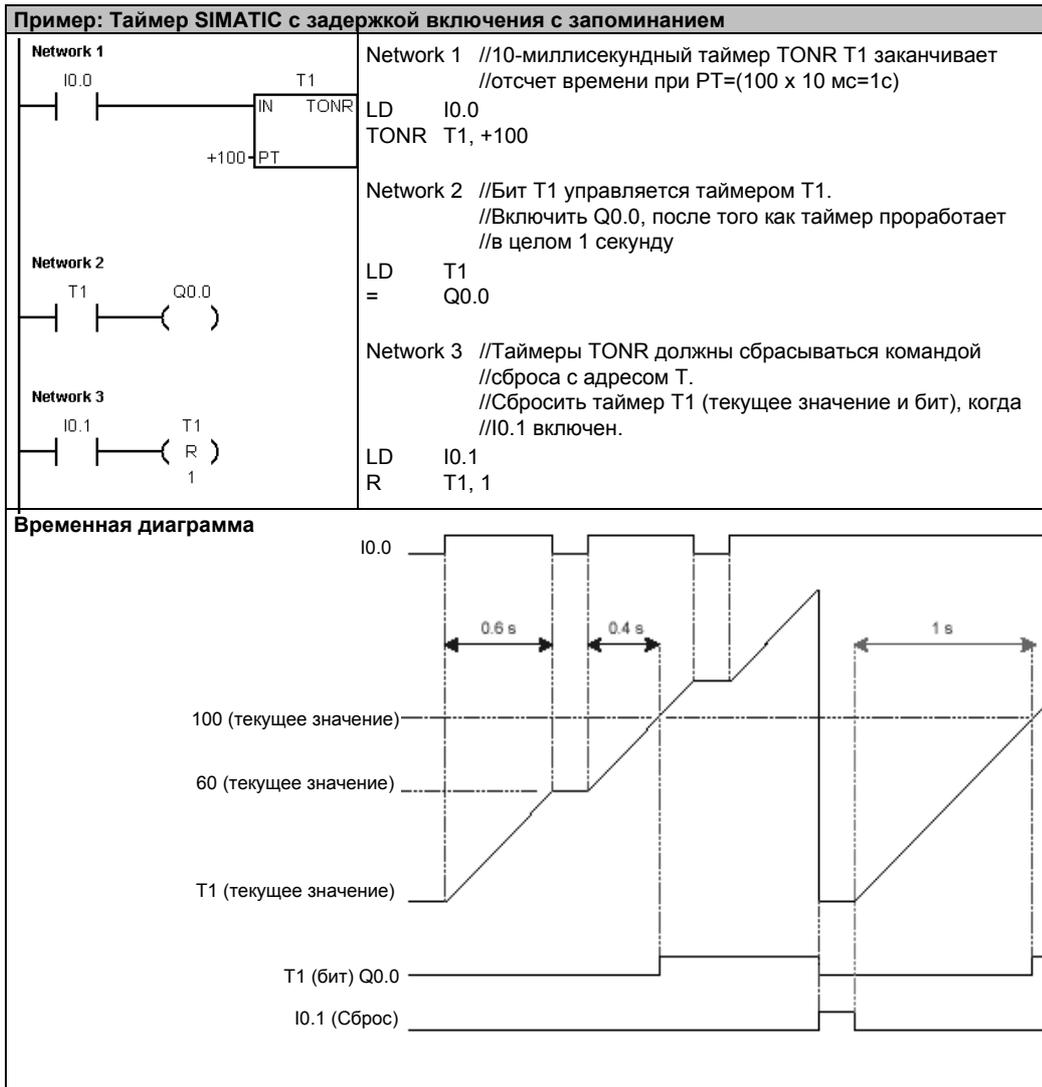
**Пример: Таймер SIMATIC с задержкой выключения**



Network 1 //10-миллисекундный таймер T33 заканчивает отсчет //времени по истечении (100 x 10 мс = 1 с) //При выключении I0.0 T33 активизируется // При включении I0.0 T33 деактивируется и //сбрасывается  
LD I0.0  
TOF T33, +100

Network 2 //Таймер T33 управляет Q0.0 через контакт таймера //T33  
LD T33  
= Q0.0





## Таймерные команды IEC

### Таймер с задержкой включения

Команда "Таймер с задержкой включения" (TON) ведет отсчет времени, когда включен разрешающий вход.

### Таймер с задержкой выключения

Команда "Таймер с задержкой выключения" (TOF) задерживает выключение выхода на фиксированный интервал времени после выключения входа.

### Импульсный таймер

Импульсный таймер (TP) генерирует импульсы заданной длительности.

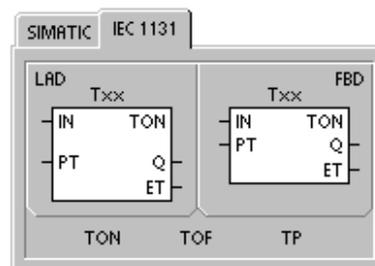


Таблица 6–75. Допустимые операнды для таймерных команд IEC

Входы/выходы	Типы данных	Операнды
Txx	TON, TOF, TP	константа (T32 – T63, T96 – T255)
IN	BOOL	I, Q, V, M, SM, S, T, C, L, поток сигнала
PT	INT	IW, QW, VW, MW, SMW, SW, LW, AC, AIW, *VD, *LD, *AC, константа
Q	BOOL	I, Q, V, M, SM, S, L
ET	INT	IW, QW, VW, MW, SMW, SW, LW, AC, AQW, *VD, *LD, *AC



#### Совет

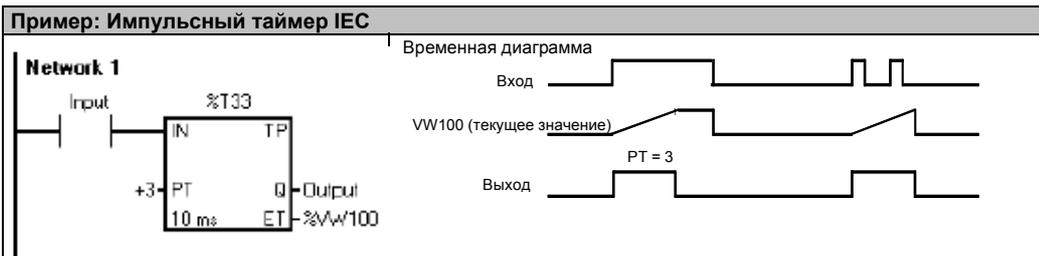
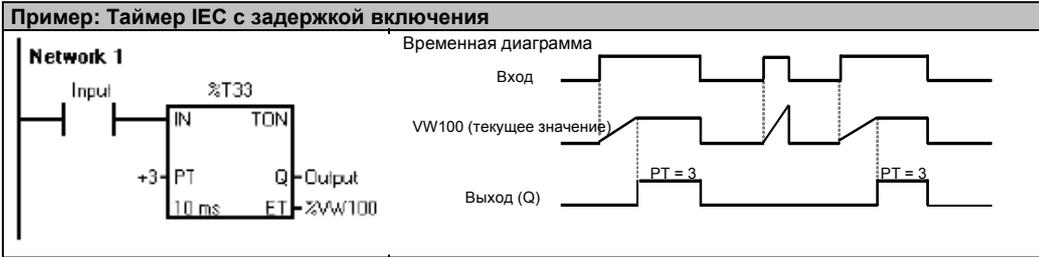
Нельзя совместно использовать одинаковые номера таймеров для TOF, TP и TON. Например, вы не можете иметь TON T32 и TOF T32.

- Команда TON ведет отсчет времени до предварительно заданного значения с момента, когда разрешающий вход (IN) становится истинным. Когда истекшее время (ET) равно заданному времени (PT), выходной бит таймера (Q) включается. Выходной бит сбрасывается, когда разрешающий вход выключается. Когда заданное время достигнуто, отсчет времени прекращается и таймер блокируется.
- Команда TOF задерживает выключение выхода на фиксированный интервал времени после выключения входа. Она ведет отсчет времени до предварительно заданного значения с момента, когда разрешающий вход (IN) выключается. Когда истекшее время (ET) равно предварительно установленному времени (PT), выходной бит таймера (Q) выключается. Как только предварительная установка достигнута, выходной бит таймера выключается, а истекшее время сохраняется до тех пор, пока разрешающий вход (IN) не включится снова. Если разрешающий вход переходит в выключенное состояние на интервал времени, более короткий, чем предустановленное время, то выходной бит остается включенным.
- Команда TP генерирует импульсы заданной длительности. Когда разрешающий вход (IN) включается, то включается и выходной бит (Q). Выходной бит остается включенным для указанного импульса в течение заранее заданного времени (PT). Как только истекшее время (ET) достигнет предварительной установки (PT), выходной бит выключается. Истекшее время сохраняется, пока не выключится разрешающий вход. Когда выходной бит включается, он остается включенным до истечения времени импульса.

Каждый отсчет текущего значения кратен базе времени. Например, значение 50 в 10–миллисекундном таймере представляет величину 500 мс. Таймеры IEC (TON, TOF и TP) имеют с тремя разрешающими способностями. Разрешающая способность определяется номером таймера, как показано в таблице 6–76.

Таблица 6–76. Разрешающая способность таймеров IEC

Разрешающая способность	Максимальное значение	Номер таймера
1 мс	32,767 с (0,546 минуты)	T32, T96
10 мс	327,67 с (5,46 минуты)	T33 – T36, T97 – T100
100 мс	3276,7 с (54,6 минуты)	T37 – T63, T101 – T255



## Интервальные таймеры

### Начало отсчета времени интервала

Команда начала отсчета времени интервала (BITIM) считывает текущее значение встроенного 1-миллисекундного счетчика и сохраняет это значение в OUT. Максимальный интервал времени для миллисекундного значения типа DWORD составляет 2 в 32-ой степени или 49,7 дня.

### Расчет времени интервала

Команда расчета времени интервала (CITIM) рассчитывает разность между текущим временем и временем, поступающие в IN. Эта разность сохраняется в OUT. Максимальный интервал времени для миллисекундного значения типа DWORD составляет 2 в 32-ой степени или 49,7 дня. CITIM автоматически обрабатывает переполнение 1-миллисекундного таймера, которое возникает внутри максимального интервала, в зависимости от того, когда была исполнена команда BITIM.

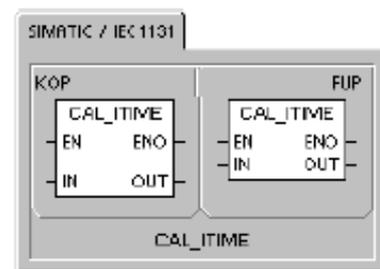
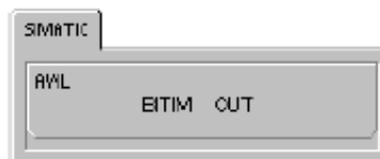
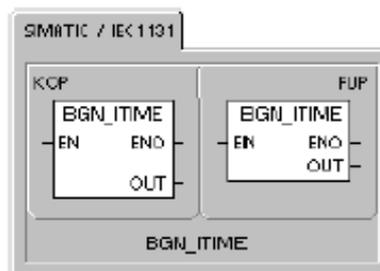
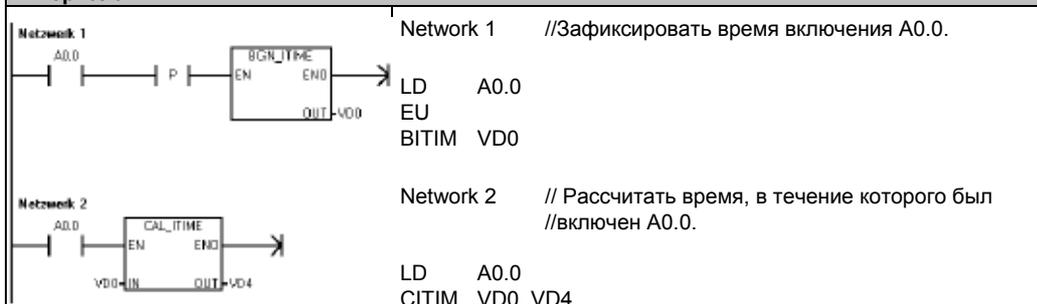


Таблица 6–77. Допустимые операнды для интервальных таймеров

Входы/выходы	Типы данных	Операнды
IN	DWORD	VD, ID, QD, MD, SMD, SD, LD, HC, AC, *VD, *LD, *AC
OUT	DWORD	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

### Пример: Команды SIMATIC для начала отсчета времени интервала и расчета времени интервала



## Операции с подпрограммами

Команда вызова подпрограммы (CALL) передает управление подпрограмме SBR\_N. Команду вызова подпрограммы можно использовать с параметрами или без них. Как только исполнение подпрограммы завершается, управление возвращается команде, следующей за вызовом подпрограммы.

Команда условного возврата из подпрограммы (CRET) завершает подпрограмму в зависимости от результата предшествующей логической операции.

Для добавления подпрограммы выберите команду меню **Edit → Insert → Subroutine [Редактировать → Вставить → Подпрограмма]**.

**Сбойные состояния, устанавливающие ENO = 0**

- 0008 (превышена максимальная вложенность для подпрограмм)
- 0006 (косвенный адрес)

В главной программе вы можете вкладывать подпрограммы друг в друга (помещать вызов подпрограммы внутри другой подпрограммы) на глубину до восьми уровней. В программе обработки прерывания вложение подпрограмм друг в друга невозможно.

Вызов подпрограммы не может быть помещен ни в какую другую подпрограмму, вызываемую из программы обработки прерывания. Рекурсия (вызов подпрограммы, вызывающей саму себя) не запрещена, но с подпрограммами ее следует использовать с осторожностью.

Таблица 6–78. Допустимые операнды для команды вызова подпрограммы

Входы/выходы	Типы данных	Операнды
SBR_N	WORD	константа для CPU 221, CPU 222, CPU 224: от 0 до 63 для CPU 224XP и CPU 226 от 0 до 127
IN	BOOL BYTE WORD, INT  DWORD, DINT  STRING	V, I, Q, M, SM, S, T, C, L, поток сигнала VB, IB, QB, MB, SMB, SB, LB, AC, *VD, *LD, *AC <sup>1</sup> , константа VW, T, C, IW, QW, MW, SMW, SW, LW, AC, AIW, *VD, *LD, *AC <sup>1</sup> , константа VD, ID, QD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC <sup>1</sup> , &VB, &IB, &QB, &MB, &T, &C, &SB, &AI, &AQ, &SMB, константа *VD, *LD, *AC, константа
IN/OUT	BOOL BYTE WORD, INT DWORD, DINT	V, I, Q, M, SM <sup>2</sup> , S, T, C, L VB, IB, QB, MB, SMB <sup>2</sup> , SB, LB, AC, *VD, *LD, *AC <sup>1</sup> VW, T, C, IW, QW, MW, SMW <sup>2</sup> , SW, LW, AC, *VD, *LD, *AC <sup>1</sup> VD, ID, QD, MD, SMD <sup>2</sup> , SD, LD, AC, *VD, *LD, *AC <sup>1</sup>
OUT	BOOL BYTE WORD, INT DWORD, DINT	V, I, Q, M, SM <sup>2</sup> , S, T, C, L VB, IB, QB, MB, SMB <sup>2</sup> , SB, LB, AC, *VD, *LD, *AC <sup>1</sup> VW, T, C, IW, QW, MW, SMW <sup>2</sup> , SW, LW, AC, AQW, *VD, *LD, *AC <sup>1</sup> VD, ID, QD, MD, SMD <sup>2</sup> , SD, LD, AC, *VD, *LD, *AC <sup>1</sup>

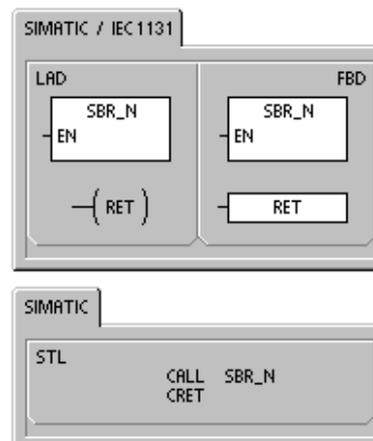
<sup>1</sup> Смещение должно быть не меньше 1  
<sup>2</sup> Смещение должно быть не меньше 30



### Совет

STEP 7-Micro/WIN автоматически добавляет команду безусловного возврата из каждой подпрограммы.

Когда вызывается подпрограмма, весь логический стек сохраняется, вершина стека устанавливается в единицу, все остальные ячейки стека устанавливаются в ноль и управление передается вызываемой подпрограмме. Когда эта подпрограмма завершается, стек восстанавливается со значениями, сохраненными в точке вызова, а управление возвращается в вызывающую программу.



Аккумуляторы являются общими для подпрограмм и вызывающей программы. При использовании подпрограммы операции сохранения и восстановления к аккумуляторам не применяются.

Если подпрограмма вызывается в одном и том же цикле несколько раз, то нельзя применять команды «Нарастающий фронт», « Падающий фронт», а также таймеры и счетчики.

### Вызов подпрограммы с параметрами

Подпрограмма может содержать передаваемые параметры. Параметры определяются в таблице локальных переменных подпрограммы. Параметру должно быть назначено символическое имя (не более 23 символов), тип переменной и тип данных. В подпрограмму и из нее может быть передано шестнадцать параметров.

Поле типа переменной в таблице локальных переменных определяет, передается ли переменная в подпрограмму (IN), в подпрограмму и из нее (IN\_OUT), или она передается из подпрограммы (OUT). Типы параметров для подпрограммы описаны в таблице 6–79. Для добавления параметра поместите курсор на поле того типа параметров (IN, IN\_OUT или OUT), который вы хотите добавить. Щелкните правой кнопкой мыши, чтобы вызвать меню для выбора. Выберите пункт Insert [Вставить], а затем пункт Row Below [Строка снизу]. Под текущей записью появится место для записи еще одного параметра выбранного типа.

Таблица 6–79. Типы параметров для подпрограммы

Параметр	Описание
IN	Параметры передаются в подпрограмму. Если параметр является прямым адресом (например, VB10), то в подпрограмму передается значение, находящееся по указанному адресу. Если параметр является косвенным адресом (например, *AC1), то в подпрограмму передается значение, находящееся по адресу, на который сделана ссылка. Если параметр является константой (16#1234) или адресом (&VB100), то в подпрограмму передается значение константы или адреса.
IN_OUT	Значение, находящееся по указанному адресу параметра, передается в подпрограмму, а результирующее значение возвращается по тому же самому адресу. Константы (например, 16#1234) и адреса (например, &VB100) не могут быть параметрами типа IN_OUT.
OUT	Результирующее значение из подпрограммы возвращается по указанному адресу параметра. Константы (например, 16#1234) и адреса (например, &VB100) не могут быть параметрами типа OUT. Так как выходные параметры не сохраняют значения, присвоенного последним исполнением подпрограммы, то вы должны присваивать значения выходам при каждом вызове подпрограммы. Обратите внимание, что команды SET и RESET влияют на значения булевых операндов только в том случае, если поток сигнала включен ON.
TEMP	Локальная память, не используемая для передаваемых параметров, может использоваться для временного хранения данных внутри подпрограммы.

Как показано на рис. 6–38, поле типа данных в таблице локальных переменных определяет размер и формат параметра. Ниже перечислены типы параметров:

- BOOL: Этот тип данных используется для отдельных битовых входов и выходов. IN3 в следующем примере является булевым входом.
- BYTE, WORD, DWORD: Эти типы данных определяют входной или выходной параметр без знака размером 1, 2 или 4 байта соответственно.

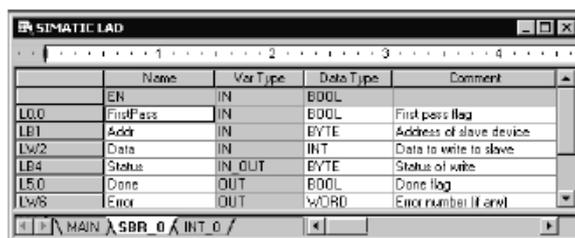


Рис. 6–38. Таблица локальных переменных

- INT, DINT: Эти типы данных определяют входной или выходной параметр без знака размером 1, 2 или 4 байта соответственно.
- REAL: Этот тип данных определяет число с плавающей точкой IEEE однократной точности (4 байта).
- STRING: Этот тип данных используется в качестве четырехбайтового указателя на строку.
- Power Flow [Поток сигнала]: Булев поток сигнала разрешен только для битовых (булевых) входов. Это описание сообщает STEP 7–Micro/WIN 32, что этот входной параметр является результатом достижения подпрограммы потоком сигнала, основанным на комбинации битовых логических операций. Входы с булевым потоком сигнала должны находиться в таблице локальных переменных перед любым другим типом входов. Таким способом можно использовать только входные параметры. Разрешающий вход (EN) и вход IN1 в следующем примере используют булеву логику.

<b>Пример: Вызов подпрограммы</b>	
<p>Ниже приведены два примера на STL. Первый набор команд STL может быть отображен только в редакторе STL, так как булевы параметры, используемые в качестве входов типа "Поток сигнала", не сохраняются в локальной памяти.</p> <p>Второй набор команд STL может быть отображен также и в редакторах LAD и FBD, так как для сохранения состояний булевых входных параметров, которые в LAD и FBD показаны как входы, принимающие поток сигнала, используется локальная память.</p>	
	<p><b>Только STL:</b></p> <pre>Network 1 LD    IO.0 CALL  SBR_0, IO.1, VB10, I1.0, &amp;VB100, *AC1, VD200</pre> <p><b>Для правильного отображения в LAD и FBD:</b></p> <pre>Network 1 LD    IO.0 =     L60.0 LD    IO.1 =     L63.7 LD    L60.0 CALL  SBR_0, L63.7, VB10, I1.0, &amp;VB100, *AC1, VD200</pre>

Адресные параметры, например, IN4 (&VB100) передаются в подпрограмму как DWORD (двойное слово без знака). Тип постоянного параметра должен быть указан для параметра в вызывающей программе с помощью описателя константы перед значением константы. Например, чтобы передать в качестве параметра константу, имеющую размер двойного слова без знака, со значением 12 345, постоянный параметр должен быть задан как DW#12345. Если описатель константы для параметра опущен, то константа может быть воспринята как имеющая другой тип.

Автоматическое преобразование типов для входных и выходных параметров не производится. Например, если таблица локальных переменных указывает, что параметр имеет тип данных REAL, а вызывающая программа задает для этого параметра двойное слово (DWORD), то это значение в подпрограмме будет рассматриваться как двойное слово.

Когда значения передаются в подпрограмму, они помещаются в локальную память подпрограммы. Самый левый столбец таблицы локальных переменных показывает адрес в локальной памяти для каждого передаваемого параметра. Значения входных параметров копируются в локальную память подпрограммы, когда подпрограмма вызывается. Значения выходных параметров копируются из локальной памяти подпрограммы в указанные адреса выходных параметров, когда исполнение подпрограммы завершается.

Размер и тип элемента данных представляются в коде параметра. Значения параметров ставятся в соответствие локальной памяти в подпрограмме следующим образом:

- Значения параметров ставятся в соответствие локальной памяти в порядке, задаваемом командой вызова подпрограммы с параметрами, начиная с L.0.
- От одного до восьми последовательных битовых значений параметров ставятся в соответствие отдельному байту, начиная с Lx.0 и вплоть до Lx.7.
- Значения, имеющие тип байт, слово или двойное слово ставятся в соответствие локальной памяти на границах байтов (LBx, LWx или LDx).

В команде вызова подпрограммы с параметрами параметры должны быть расположены в следующем порядке: сначала входные параметры, за ними параметры типа IN\_OUT, а затем выходные параметры.

выходные параметры.

Если вы программируете на STL, то формат команды CALL имеет вид:

CALL номер подпрограммы, параметр 1, параметр 2, ... , параметр n

Пример: Команды вызова подпрограммы и возврата из подпрограммы		
M A I N	<p><b>Network 1</b></p>	<p>Network 1 //В первом цикле вызвать подпрограмму 0 для //инициализации.</p> <pre>LD SM0.1 CALL SBR_0</pre>
S B R 0	<p><b>Network 1</b></p> <p><b>Network 2</b></p>	<p>Network 1 //Вы можете использовать условный возврат // для выхода из подпрограммы перед //последним сегментом.</p> <pre>LD M14.3 CRET</pre> <p>Network 2 //Этот сегмент будет пропущен, если M14.3 //установлен.</p> <pre>LD SM0.0 MOVB 10, VB0</pre>

Пример: Вызов подпрограммы со строками		
<p>В этом примере различный строковый литерал копируется по определенному адресу в зависимости от входа. Уникальный адрес этой строки сохраняется. Затем адрес строки передается в подпрограмму с помощью косвенного адреса. Типом данных входного параметра подпрограммы является строка. Затем подпрограмма смещает строку на другой адрес.</p> <p>Строковый литерал также может быть передан в подпрограмму. Ссылка на строку в подпрограмме всегда одна и та же.</p>		
M A I N	<p><b>Network 1</b></p> <p><b>Network 2</b></p> <p><b>Network 3</b></p> <p><b>Network 4</b></p>	<p>Network 1 //</p> <pre>LD I0.0 SSCPY «string1», VB100 AENO MOVD &amp;VB100, VD0</pre> <p>Network2 //</p> <pre>LD I0.1 SSCPY «string2», VB200 AENO MOVD &amp;VB200, VD0</pre> <p>Network3 //</p> <pre>LD I0.2 CALL SBR_0, *VD0</pre>
S B R 0	<p><b>Network 1</b></p>	<p>Network 1 //</p> <pre>LD SM0.0 SSCPY *LD0, VB300</pre>

