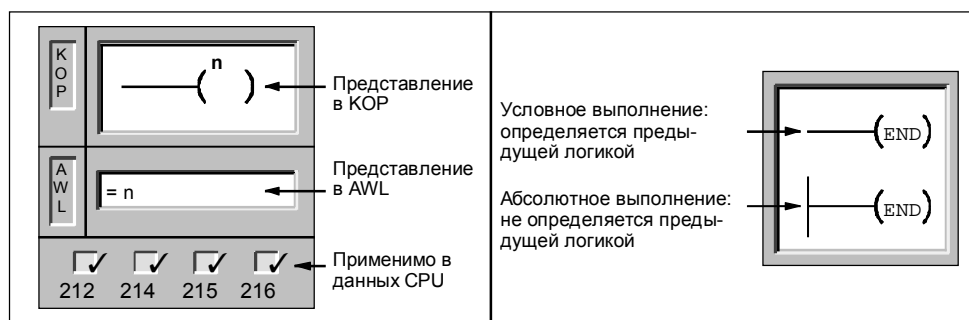


Набор операций

9

В этой главе используются следующие соглашения для иллюстрации эквивалентных операций контактного плана (KOP) и списка команд (AWL) и указания CPU, в которых применима соответствующая операция.



Обзор главы

Раздел	Описание	Страница
9.1	Допустимые диапазоны CPU S7-200	9-2
9.2	Операции над контактами	9-4
9.3	Операции над выходами	9-9
9.4	Таймерные операции и операции со счетчиками	9-11
9.5	Арифметические операции, инкрементирование и декрементирование	9-18
9.6	PID-операция	9-25
9.7	Операции передачи, сдвига и циклического сдвига	9-37
9.8	Операции управления программой	9-48
9.9	Операции со стеком	9-62
9.10	Логические операции	9-65
9.11	Табличные операции и операции поиска	9-70
9.12	Операции преобразования	9-75
9.13	Операции с быстрыми счетчиками	9-80
9.14	Операции с быстрыми выходами	9-96
9.15	Прерывания	9-107
9.16	Коммуникационные операции	9-116
9.17	Операции с часами реального времени	9-125

9.1 Допустимые диапазоны CPU S7–200

Таблица 9–1. Свод областей памяти и функциональных возможностей CPU S7–200

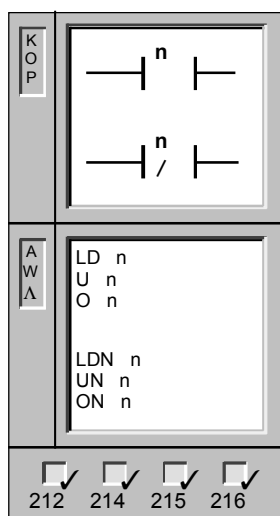
Описание	CPU 212	CPU 214	CPU 215	CPU 216
Размер программы пользователя	512 слов	2 К слов	4 К слов	4 К слов
Размер данных пользователя	512 слов	2 К слов	2,5 К слов	2,5 К слов
Отображение процесса на входах	E0.0 - E7.7	E0.0 - E7.7	E0.0 - E7.7	E0.0 - E7.7
Отображение процесса на выходах	A0.0 - A7.7	A0.0 - A7.7	A0.0 - A7.7	A0.0 - A7.7
Аналоговые входы (защищенные от записи)	AEW0 - AEW30	AEW0 - AEW30	AEW0 - AEW30	AEW0 - AEW30
Аналоговые выходы (защищенные от записи)	AAW0 - AAW30	AAW0 - AAW30	AAW0 - AAW30	AAW0 - AAW30
Память переменных (V) Область, устойчивая к нулевому напряжению (макс.)	V0.0 - V1023.7 V0.0 - V199.7	V0.0 - V4095.7 V0.0 - V1023.7	V0.0 - V5119.7 V0.0 - V5119.7	V0.0 - V5119.7 V0.0 - V5119.7
Меркеры (M) Область, устойчивая к нулевому напряжению (макс.)	M0.0 - M15.7 MB0 - MB13	M0.0 - M31.7 MB0 - MB13	M0.0 - M31.7 MB0 - MB13	M0.0 - M31.7 MB0 - MB13
Специальные меркеры (SM) Защищенные от записи	SM0.0 - SM45.7 SM0.0 - SM29.7	SM0.0 - SM85.7 SM0.0 - SM29.7	SM0.0 - SM194.7 SM0.0 - SM29.7	SM0.0 - SM194.7 SM0.0 - SM29.7
Таймеры Формирование задержки включения с запоминанием 1 мс	64 (T0 - T63) T0	128 (T0 - T127) T0, T64	256 (T0 - T255) T0, T64	256 (T0 - T255) T0, T64
Формирование задержки включения с запоминанием 10 мс	T1 - T4	T1 - T4, T65 - T68	T1 - T4, T65 - T68	T1 - T4, T65 - T68
Формирование задержки включения с запоминанием 100 мс	T5 - T31	T5 - T31, T69 - T95	T5 - T31, T69 - T95	T5 - T31, T69 - T95
Формирование задержки включения 1 мс	T32	T32, T96	T32, T96	T32, T96
Формирование задержки включения 10 мс	T33 - T36	T33–T36, T97–T100	T33–T36, T97–T100	T33–T36, T97–T100
Формирование задержки включения 100 мс	T37 - T63	T37–T63, T101–T127	T37–T63, T101–T255	T37–T63, T101–T255
Счетчики	Z0 - Z63	Z0 - C127	Z0 - C255	Z0 - C255
Быстрые счетчики	HC0	HC0 - HC2	HC0 - HC2	HC0 - HC2
Реле шагового управления	S0.0 - S7.7	S0.0 - S15.7	S0.0 - S31.7	S0.0 - S31.7
Аккумуляторы	AC0 - AC3	AC0 - AC3	AC0 - AC3	AC0 - AC3
Переходы/Метки перехода	0 - 63	0 - 255	0 - 255	0 - 255
Вызовы/Подпрограммы	0 - 15	0 - 63	0 - 63	0 - 63
Программы обработки прерываний	0 - 31	0 - 127	0 - 127	0 - 127
События прерываний	0, 1, 8 - 10, 12	0 - 20	0 - 23	0 - 26
PID–регулятор	не поддержив.	не поддерживается	0 - 7	0 - 7
Порты	0	0	0	0 и 1

Таблица 9–2. Области операндов CPU S7–200

Формат доступа	CPU 212	CPU 214	CPU 215	CPU 216
Бит (Байт.Бит)	V 0.0 - 1023.7 E 0.0 - 7.7 A 0.0 - 7.7 M 0.0 - 15.7 SM 0.0 - 45.7 T 0 - 63 Z 0 - 63 S 0.0 - 7.7	V 0.0 - 4095.7 E 0.0 - 7.7 A 0.0 - 7.7 M 0.0 - 31.7 SM 0.0 - 85.7 T 0 - 127 Z 0 - 127 S 0.0 - 15.7	V 0.0 - 5119.7 E 0.0 - 7.7 A 0.0 - 7.7 M 0.0 - 31.7 SM 0.0 - 194.7 T 0 - 255 Z 0 - 255 S 0.0 - 31.7	V 0.0 - 5119.7 E 0.0 - 7.7 A 0.0 - 7.7 M 0.0 - 31.7 SM 0.0 - 194.7 T 0 - 255 Z 0 - 255 S 0.0 - 31.7
Байт	VB 0 - 1023 EB 0 - 7 AB 0 - 7 MB 0 - 15 SMB 0 - 45 AC 0 - 3 SB 0 - 7 Константа	VB 0 - 4095 EB 0 - 7 AB 0 - 7 MB 0 - 31 SMB 0 - 85 AC 0 - 3 SB 0 - 15 Константа	VB 0 - 5119 EB 0 - 7 AB 0 - 7 MB 0 - 31 SMB 0 - 194 AC 0 - 3 SB 0 - 31 Константа	VB 0 - 5119 EB 0 - 7 AB 0 - 7 MB 0 - 31 SMB 0 - 194 AC 0 - 3 SB 0 - 31 Константа
Слово	VW 0 - 1022 T 0 - 63 Z 0 - 63 EW 0 - 6 AW 0 - 6 MW 0 - 14 SMW 0 - 44 AC 0 - 3 AEW 0 - 30 AAW 0 - 30 SW 0 - 6 Константа	VW 0 - 4094 T 0 - 127 Z 0 - 127 EW 0 - 6 AW 0 - 6 MW 0 - 30 SMW 0 - 84 AC 0 - 3 AEW 0 - 30 AAW 0 - 30 SW 0 - 14 Константа	VW 0 - 5118 T 0 - 255 Z 0 - 255 EW 0 - 6 AW 0 - 6 MW 0 - 30 SMW 0 - 193 AC 0 - 3 AEW 0 - 30 AAW 0 - 30 SW 0 - 30 Константа	VW 0 - 5118 T 0 - 255 Z 0 - 255 EW 0 - 6 AW 0 - 6 MW 0 - 30 SMW 0 - 193 AC 0 - 3 AEW 0 - 30 AAW 0 - 30 SW 0 - 30 Константа
Двойное слово	VD 0 - 1020 ED 0 - 4 AD 0 - 4 MD 0 - 12 SMD 0 - 42 AC 0 - 3 HC 0 SD 0 - 4 Константа	VD 0 - 4092 ED 0 - 4 AD 0 - 4 MD 0 - 28 SMD 0 - 82 AC 0 - 3 HC 0 - 2 SD 0 - 12 Константа	VD 0 - 5116 ED 0 - 4 AD 0 - 4 MD 0 - 28 SMD 0 - 191 AC 0 - 3 HC 0 - 2 SD 0 - 28 Константа	VD 0 - 5116 ED 0 - 4 AD 0 - 4 MD 0 - 28 SMD 0 - 191 AC 0 - 3 HC 0 - 2 SD 0 - 28 Константа

9.2 Операции над контактами

Стандартные контакты



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

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

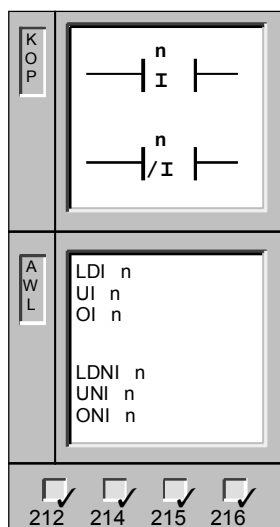
Размыкающий контакт замкнут (включен), если значение бита с адресом n равно 0.

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

Операнды: n : E, A, M, SM, T, Z, V

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

Контакты с непосредственным доступом



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

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

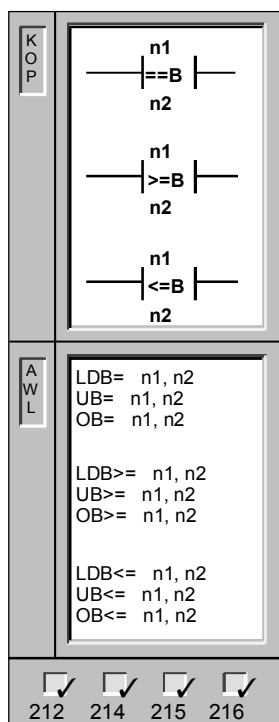
Размыкающий контакт с непосредственным доступом замкнут (включен), если значение бита заданного входа n равно 0.

В AWL размыкающий контакт с непосредственным доступом представляется операциями **Непосредственная загрузка логического отрицания бита**, **Непосредственное логическое И логического отрицания бита** и **Непосредственное логическое ИЛИ логического отрицания бита**. Эти операции загружают логическое отрицание непосредственного значения бита с адресом n в вершину стека или логически связывают логическое отрицание этого бита с значением в вершине стека через “И” или “ИЛИ”.

Операнды: n : E

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

Сравнение байтов



Операция **Сравнение байтов** сравнивает два значения n1 и n2 друг с другом. Вы можете производить следующие сравнения: n1 = n2, n1 >= n2 и n1 <= n2.

Операнды: n1, n2: VB, EB, AB, MB, SMB, AC, константа, *VD, *AC, SB

В KOP контакт замкнут, если результатом сравнения является истина.

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

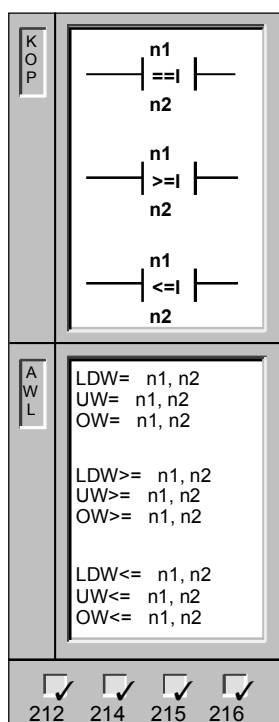
Сравнения байтов не учитывают знак.

Указание: Вы можете выполнять сравнения <>, < и >, используя операцию NOT совместно с операциями >=, = или <=. Две следующие операции соответствуют сравнению <> (не равно) между VB100 и значением 50:

```

LDB=    VB100, 50
NOT
    
```

Сравнение слов



Операция **Сравнение слов** сравнивает два значения n1 и n2 друг с другом. Вы можете производить следующие сравнения: n1 = n2, n1 >= n2 и n1 <= n2.

Операнды: n1, n2: VW, T, Z, EW, AW, MW, SMW, AC, AEW, константа, *VD, *AC, SW

В KOP контакт замкнут, если результатом сравнения является истина.

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

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

Указание: Вы можете выполнять сравнения <>, < и >, используя операцию NOT совместно с операциями >=, = или <=. Две следующие операции соответствуют сравнению <> (не равно) между VW100 и значением 50:

```

LDW=    VW100, 50
NOT
    
```

Сравнение двойных слов

K O P	
A W L	LDD= n1, n2 UD= n1, n2 OD= n1, n2
	LDD>= n1, n2 UD>= n1, n2 OD>= n1, n2
	LDD<= n1, n2 UD<= n1, n2 OD<= n1, n2
	<input checked="" type="checkbox"/> 212
	<input checked="" type="checkbox"/> 214
	<input checked="" type="checkbox"/> 215
	<input checked="" type="checkbox"/> 216

Операция **Сравнение двойных слов** сравнивает два значения n1 и n2 друг с другом. Вы можете производить следующие сравнения: n1 = n2, n1 >= n2 и n1 <= n2.

Операнды: n1, n2: VD, ED, AD, MD, SMD, AC, константа, *VD, *AC, SD, HC,

В KOP контакт замкнут, если результатом сравнения является истина.

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

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

Указание: Вы можете выполнять сравнения <>, < и >, используя операцию NOT совместно с операциями >=, = или <=. Две следующие операции соответствуют сравнению <> (не равно) между VD100 и значением 50:

```
LDD=    VD100, 50
NOT
```

Сравнение действительных чисел

K O P	
A W L	LDR= n1, n2 UR= n1, n2 OR= n1, n2
	LDR>= n1, n2 UR>= n1, n2 OR>= n1, n2
	LDR<= n1, n2 UR<= n1, n2 OR<= n1, n2
	<input checked="" type="checkbox"/> 212
	<input checked="" type="checkbox"/> 214
	<input checked="" type="checkbox"/> 215
	<input checked="" type="checkbox"/> 216

Операция **Сравнение действительных чисел** сравнивает два значения n1 и n2 друг с другом. Вы можете производить следующие сравнения: n1 = n2, n1 >= n2 и n1 <= n2.

Операнды: n1, n2: VD, ED, AD, MD, SMD, AC, константа, *VD, *AC, SD

В KOP контакт замкнут, если результатом сравнения является истина.

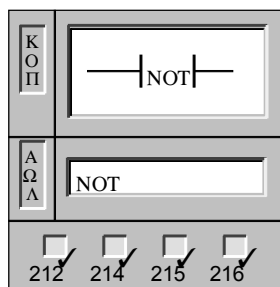
В AWL эти операции загружают значение "1" в вершину стека или логически связывают значение "1" с значением в вершине стека через "И" или "ИЛИ", если результатом сравнения является истина.

Сравнения действительных чисел учитывают знак.

Указание: Вы можете выполнять сравнения <>, < и >, используя операцию NOT совместно с операциями >=, = или <=. Две следующие операции соответствуют сравнению <> (не равно) между VD100 и значением 50:

```
LDR=    VD100, 50
NOT
```

NOT

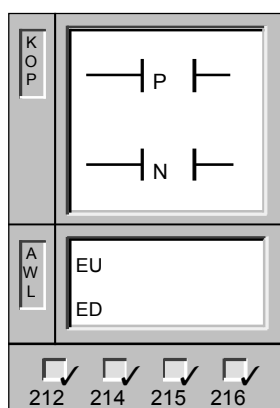


Контакт **NOT** изменяет состояние потока сигнала. Если поток сигнала достигает контакта NOT, то он у контакта останавливается. Если поток сигнала не достигает контакта NOT, то у контакта создается поток сигнала.

В AWL операция **NOT** изменяет вершину стека с "0" на "1" или с "1" на "0".

Операнды: нет

Нарастающий фронт и спадающий фронт



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

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

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

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

Операнды: нет

Примеры контактов

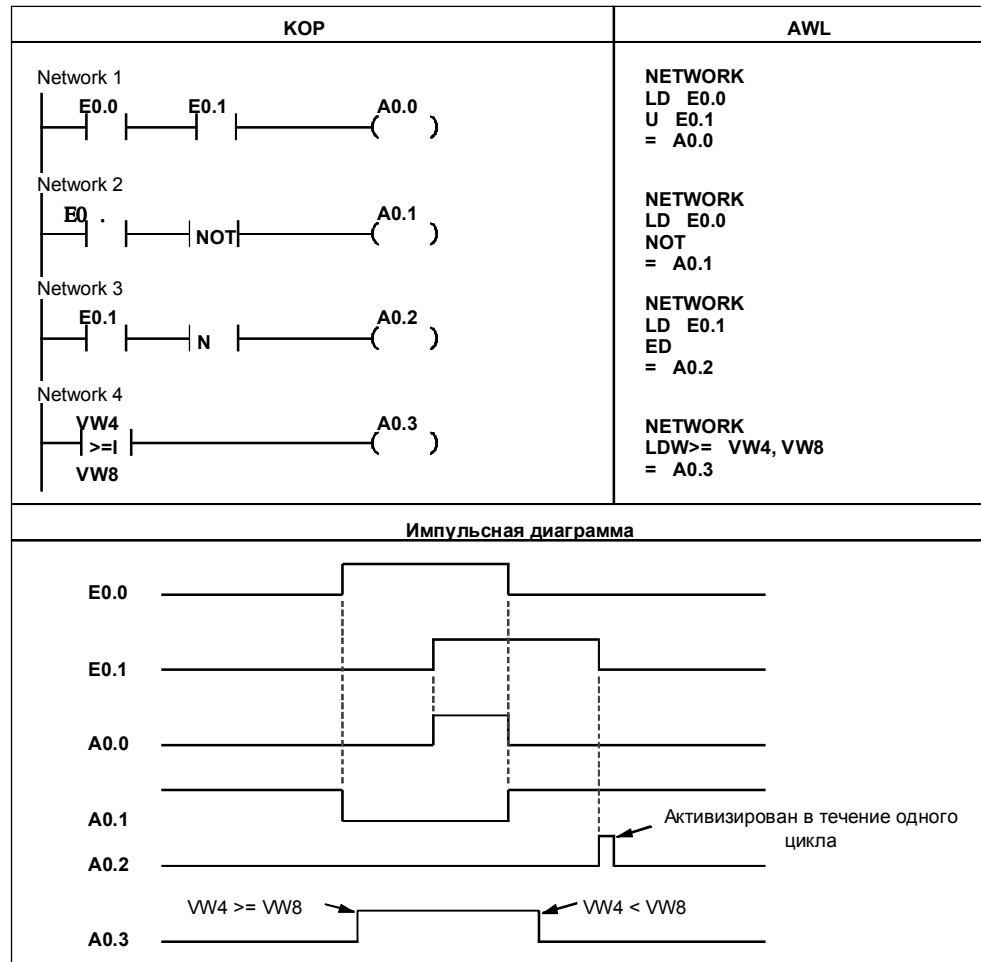
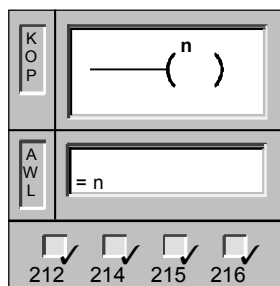


Рис. 9-1. Пример логических операций над контактами в KOP и AWL

9.3 Операции над выходами

Присваивание

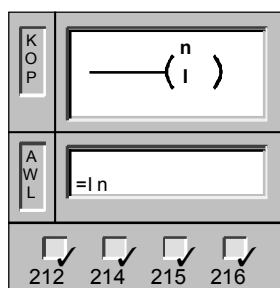


Если выполняется операция **Присваивание**, то заданный параметр (n) включается.

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

Операнды: n: E, A, M, SM, T, Z, V, S

Прямое присваивание значения биту



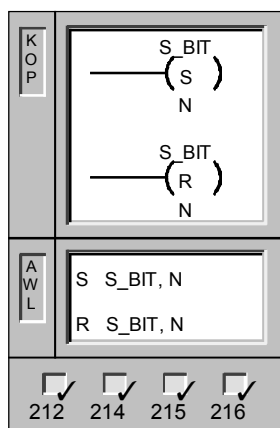
Если выполняется операция **Прямое присваивание значения биту**, то заданный физический выход (n) непосредственно включается.

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

Операнды: n: A

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

Установка и сброс



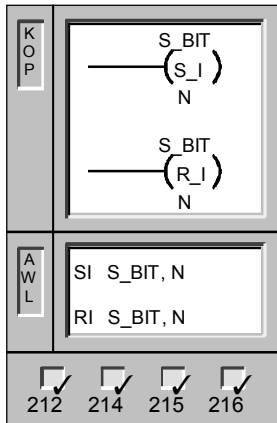
Если выполняются операции **Установка** и **Сброс**, то заданное количество (N) входов или выходов, начиная с S_BIT, устанавливается (включается) или сбрасывается (выключается).

Операнды: S_BIT: E, A, M, SM, T, Z, V, S

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

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

Прямая установка и сброс



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

Операнды: S_BIT: A
 N: EB, AB, MB, SMB, VB, AC, константа, *VD, *AC, SB

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

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

Примеры операций над выходами

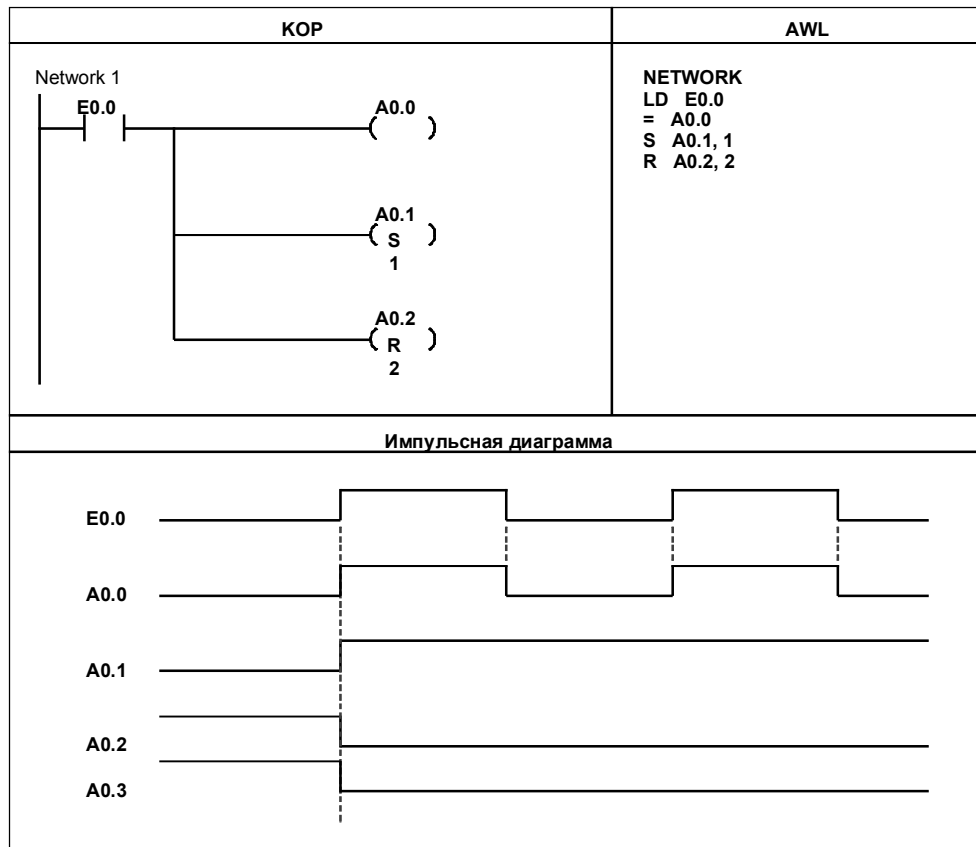
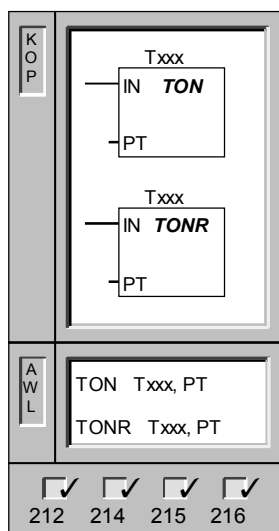


Рис. 9-2. Примеры операций над выходами в KOP и AWL

9.4 Таймерные операции и операции со счетчиками

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

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



Если операция “Запуск таймера как формирователя задержки включения” деактивируется, то таймер сбрасывается. Если операция “Запуск таймера как формирователя задержки включения с запоминанием” деактивируется, то таймер останавливается. Оба таймера останавливаются, когда они достигают максимального значения.

Операнды:	Txxx:	TON	TONR
	1 мс	T32, T96	T0, T64
	10 мс	T33 - T36 T97 - T100	T1 - T4 T65 - T68
	100 мс	T37 - T63 T101 - T255	T5 - T31 T69 - T95

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

Таймеры TON и TONR доступны с тремя разными разрешающими способностями. Разрешающая способность определяется номером таймера (см. табл. 9–3). Каждый отсчет в текущем значении представляет величину, кратную базе времени. Например, для таймера с разрешающей способностью 10 мс отсчет 50 соответствует текущему значению 500 мс.

Таблица 9–3. Номера таймеров и разрешающие способности

Тай-мер	Разрешающая способность	Максимальное значение	CPU 212	CPU 214	CPU 215/216
TON	1 мс	32,767 секунд (с)	T32	T32, T96	T32, T96
	10 мс	327,67 с	T33 - T36	T33 - T36, T97 - T100	T33 - T36, T97 - T100
	100 мс	3276,7 с	T37 - T63	T37 - T63, T101 - T127	T37 - T63, T101 - T255
TONR	1 мс	32,767 с	T0	T0, T64	T0, T64
	10 мс	327,67 с	T1 - T4	T1 - T4, T65 - T68	T1 - T4, T65 - T68
	100 мс	3276,7 с	T5 - T31	T5 - T31, T69 - T95	T5 - T31, T69 - T95

Описание таймерных операций для S7–200

С помощью таймеров Вы можете выполнять функции, управляемые временем. S7–200 предоставляет две разные таймерные операции: запуск таймера как формирователя задержки включения (TON) и запуск таймера как формирователя задержки включения с запоминанием (TONR). Эти два типа таймеров (TON и TONR) различаются по своей реакции на состояние входа разблокировки. Оба таймера TON и TONR отсчитывают значение времени, когда активизирован вход разблокировки. При выключенном входе разблокировки оба таймера не работают, однако в то время как таймер TON автоматически сбрасывается, таймер TONR сохраняет свое последнее значение времени, а не сбрасывается. Поэтому лучше всего использовать таймер TON, когда нужен отдельный интервал времени, и использовать таймер TONR, когда Вы хотите накапливать несколько интервалов времени.

Таймеры S7–200 имеют следующие свойства:

- Таймеры управляются через один единственный вход разблокировки и обладают текущим значением, указывающим значение времени, истекшего с момента разблокировки. Кроме того, таймеры имеют в своем распоряжении предварительно устанавливаемое значение (PT), которое сравнивается с текущим значением каждый раз, когда актуализируется текущее значение или выполняется таймерная операция.
- Соответственно результату сравнения предварительно установленного значения с текущим значением устанавливается или сбрасывается бит таймера.
- Если текущее значение больше, чем предварительно установленное значение, или равно ему, то бит таймера (T–бит) включается.

Указание

Некоторые из текущих значений времени могут определяться как ретранзитные. Биты таймеров не являются ретранзитными и устанавливаются только в результате сравнения текущего значения с предварительно установленным значением.

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

С помощью нескольких таймеров с разрешающей способностью 1 мс Вы можете создать событие прерывания. Подробную информацию о прерываниях, управляемых временем, Вы найдете в разделе 9.15.

Актуализация таймеров с разрешающей способностью 1 мс

CPU S7–200 имеет в своем распоряжении таймеры, которые актуализируются один раз в миллисекунду (таймеры с разрешающей способностью 1 мс) системной программой, ответственной за актуализацию базы времени. Эти таймеры служат для точного управления операциями.

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

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

Сброс разблокированного таймера с разрешающей способностью 1 мс выключает этот таймер, его текущее значение устанавливается в нуль, а бит таймера стирается.

Указание

Системная программа, ответственная за системную базу времени величиной 1 мс, не зависит от разблокировки и блокировки таймеров. Таймер с разрешающей способностью 1 мс разблокируется в произвольный момент времени внутри текущего интервала 1 мс. Поэтому этот интервал для таймера с разрешающей способностью 1 мс может иметь максимальную длительность 1 мс. Вам следует задавать в предварительной установке значение, которое на 1 больше, чем самый короткий желаемый интервал. Например, чтобы иметь в распоряжении интервал величиной не менее 56 мс при использовании таймера с разрешающей способностью 1 мс, Вам нужно задать предварительно устанавливаемое значение времени равным 57.

Актуализация таймеров с разрешающей способностью 10 мс

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

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

Сброс разблокированного таймера с разрешающей способностью 10 мс выключает этот таймер, его текущее значение устанавливается в нуль, а бит таймера стирается.

Указание

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

Актуализация таймеров с разрешающей способностью 100 мс

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

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

Указание

Так как накопление интервалов величиной 100 мс происходит независимо от разблокировки и блокировки таймеров, то таймеры с разрешающей способностью 100 мс разблокируются в произвольный момент времени внутри определенного интервала величиной 100 мс. Поэтому этот интервал для таймера с разрешающей способностью 100 мс может иметь максимальную длительность 100 мс. Вам следует задавать в предварительной установке значение, которое на 1 больше, чем самый короткий желаемый интервал. Например, чтобы иметь в распоряжении интервал величиной не менее 2100 мс при использовании таймера с разрешающей способностью 100 мс, Вам нужно задать предварительно устанавливаемое значение времени равным 22.

Актуализация текущего значения таймера

Воздействие, оказываемое различной актуализацией текущих значений таймеров, определяется тем, как Вы используете таймеры. Рассмотрите, например, таймерную операцию на рисунке 9–3.

- Если используется таймер с разрешающей способностью 1 мс, то А0.0 включается на время цикла всякий раз, когда текущее значение таймера актуализируется после исполнения команды “Размыкающий контакт Т32” и перед исполнением команды “Замыкающий контакт Т32”.
- Если используется таймер с разрешающей способностью 10 мс, то А0.0 не включается ни разу, потому что бит таймера Т33 включается с начала цикла до момента времени, когда выполняется таймерный блок. После выполнения таймерного блока текущее значение таймера и бит таймера устанавливаются в нуль. Если выполняется команда “Замыкающий контакт Т33”, то Т33 не активизируется и А0.0 выключается.
- Если используется таймер с разрешающей способностью 100 мс, то А0.0 включается на время цикла всегда, когда текущее значение таймера достигает предварительно установленного значения.

При использовании размыкающего контакта А0.0 вместо бита таймера в качестве входа разблокировки таймерного блока вместо бита таймера, то гарантируется, что выход А0.0 включается на время цикла всякий раз, когда таймер достигает предварительно установленного значения (см. рис. 9–3). На рис. 9–4 и 9–5 показаны примеры таймерных операций для контактного плана и списка команд.

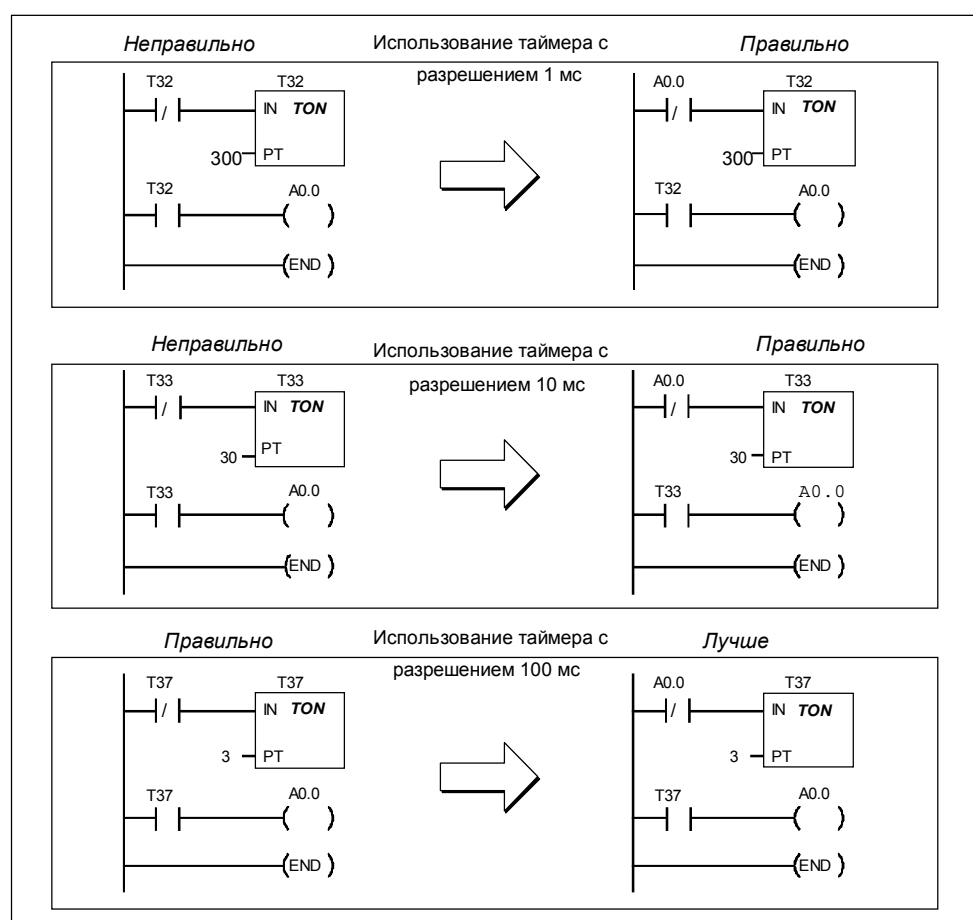


Рис. 9-3. Пример автоматического перезапуска таймера

Пример операции “Запуск таймера как формирователя задержки включения”

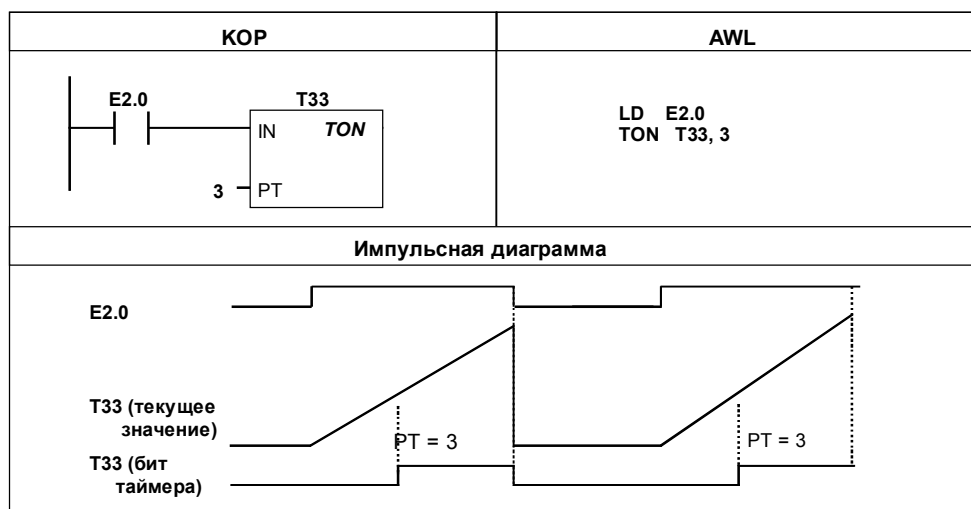


Рис. 9-4. Пример запуска таймера как формирователя задержки включения для KOP и AWL

Пример операции “Запуск таймера как формирователя задержки включения с запоминанием”

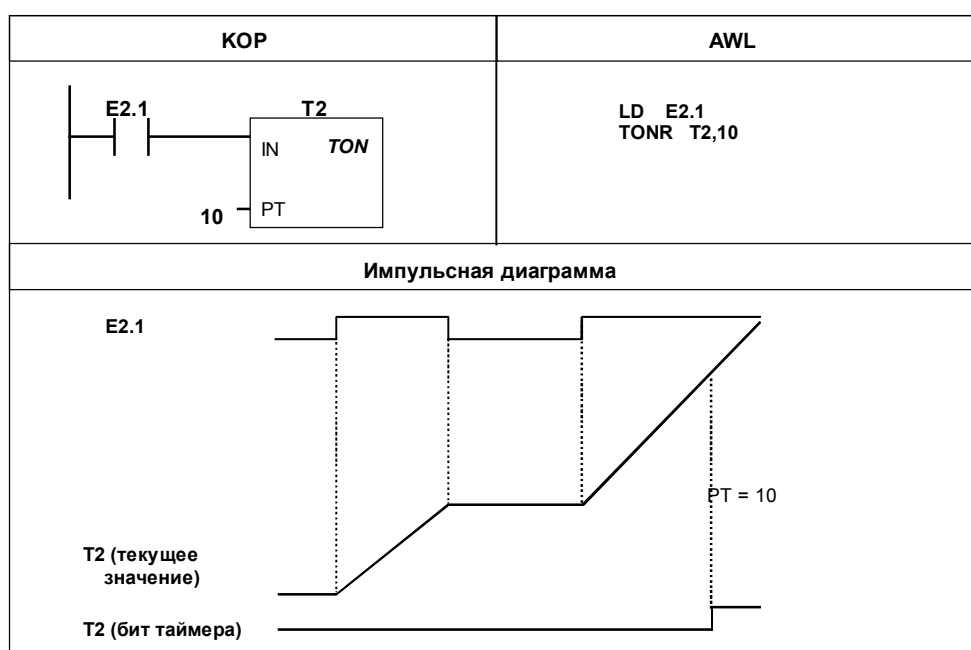
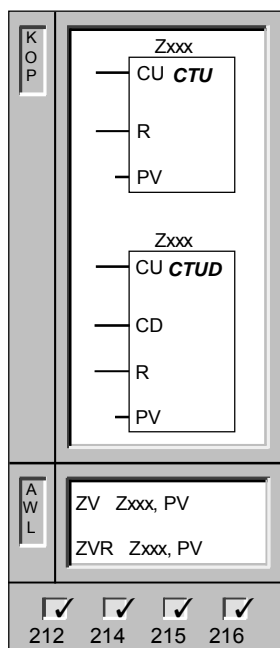


Рис. 9-5. Пример операции запуска таймера как формирователя задержки включения с запоминанием для KOP и AWL

Прямой счет и реверсивный счет



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

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

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

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

Операнды: Z_{xxx} : от 0 до 255
 PV: VW, T, Z, EW, AW, MW, SMW, AC, AEW, константа, *VD, *AC, SW

Описание операций со счетчиками S7-200

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

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

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

Прямой и реверсивный счетчики имеют текущее значение, в котором хранится текущее значение счетчика. Эти счетчики имеют в своем распоряжении также предварительно устанавливаемое значение (PV), которое при выполнении операции сравнивается с текущим значением. Если текущее значение больше, чем предварительно установленное значение, или равно ему, то активизируется бит счетчика (Z-бит). Во всех других случаях бит счетчика выключается.

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

Указание

Так как каждый счетчик обладает собственным текущим значением, то не назначайте одинаковый номер нескольким счетчикам (прямой и реверсивный счетчики имеют доступ к одному и тому же текущему значению).

Примеры операций со счетчиками

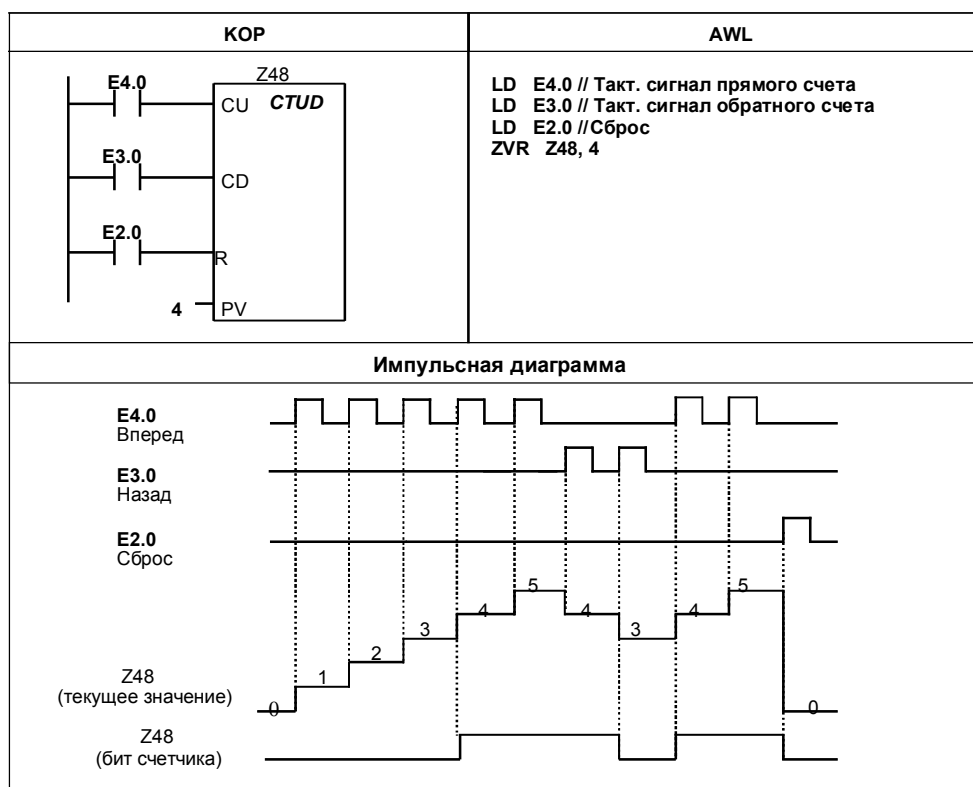
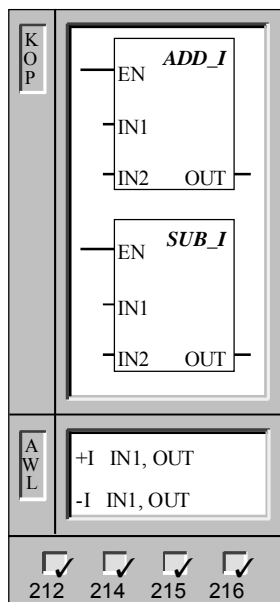


Рис. 9-6. Пример операций счетчика в KOP и AWL

9.5 Арифметические операции, инкрементирование и декрементирование

Сложение и вычитание целых чисел (16 бит)



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

Операнды: IN1, IN2: VW, T, Z, EW, AW, MW, SMW, AC, AEW, константа, *VD, *AC, SW

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

KOP: $IN1 + IN2 = OUT$
 $IN1 - IN2 = OUT$

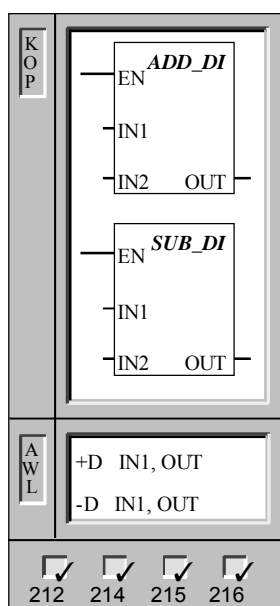
AWL: $IN1 + OUT = OUT$
 $OUT - IN1 = OUT$

Указание: При программировании в KOP Вы можете указать, что IN1 совпадает с OUT. Таким образом, Вы экономите место в памяти.

Эти операции влияют на следующие специальные меркеры:

SM1.0 (нуль); SM1.1 (переполнение); SM1.2 (отрицательный)

Сложение и вычитание целых чисел (32 бита)



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

Операнды: IN1, IN2: VD, ED, AD, MD, SMD, AC, HC, константа, *VD, *AC, SD

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

KOP: $IN1 + IN2 = OUT$
 $IN1 - IN2 = OUT$

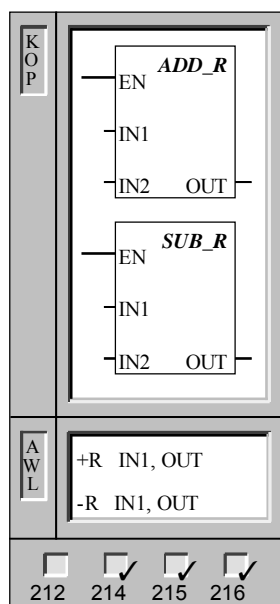
AWL: $IN1 + OUT = OUT$
 $OUT - IN1 = OUT$

Указание: При программировании в KOP Вы можете указать, что IN1 совпадает с OUT. Таким образом, Вы экономите место в памяти.

Эти операции влияют на следующие специальные меркеры:

SM1.0 (нуль); SM1.1 (переполнение); SM1.2 (отрицательный)

Сложение и вычитание действительных чисел



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

Операнды: IN1, IN2: VD, ED, AD, MD, SMD, AC,
константа, *VD, *AC, SD

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

KOP: IN1 + IN2 = OUT
IN1 - IN2 = OUT

AWL: IN1 + OUT = OUT
OUT - IN1 = OUT

Указание: При программировании в KOP Вы можете указать, что IN1 совпадает с OUT. Таким образом, Вы экономите место в памяти.

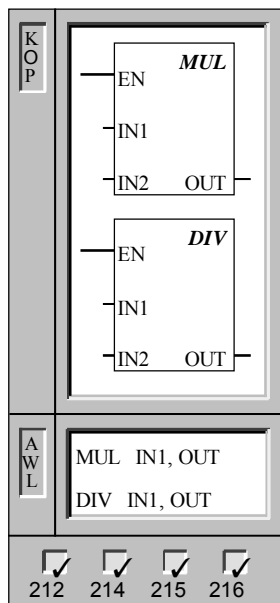
Эти операции влияют на следующие специальные меркеры:

SM1.0 (нуль); SM1.1 (переполнение); SM1.2 (отрицательный)

Указание

Действительные числа или числа с плавающей точкой представляются в формате, описанном в стандарте ANSI/IEEE 754–1985 (одинарная точность). Подробную информацию об этих числах Вы найдете в этом стандарте.

Умножение и деление целых чисел (16 бит)



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

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

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

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

Операнды: IN1, IN2: VW, T, Z, EW, AW, MW, SMW, AC, AEW, константа, *VD, *AC, SW

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

KOP: IN1 * IN2 = OUT
IN1 / IN2 = OUT

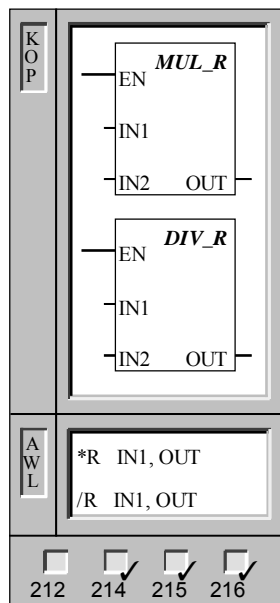
AWL: IN1 * OUT = OUT
OUT / IN1 = OUT

Указание: При программировании в KOP Вы можете указать, что IN1 совпадает с OUT. Таким образом, Вы сэкономите место в памяти.

Эти операции влияют на следующие специальные меркеры:

SM1.0 (нуль); SM1.1 (переполнение); SM1.2 (отрицательный); SM1.3 (деление на нуль)

Умножение и деление действительных чисел



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

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

Операнды: IN1, IN2: VD, ED, AD, MD, SMD, AC, константа, *VD, *AC, SD

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

KOP: $IN1 * IN2 = OUT$
 $IN1 / IN2 = OUT$

AWL: $IN1 * OUT = OUT$
 $OUT / IN1 = OUT$

Указание: При программировании в KOP Вы можете указать, что IN1 совпадает с OUT. Таким образом, Вы экономите место в памяти.

Эти операции влияют на следующие специальные меркеры:

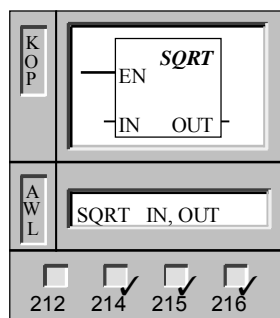
SM1.0 (нуль); SM1.1 (переполнение); SM1.2 (отрицательный); SM1.3 (деление на нуль)

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

Указание

Действительные числа или числа с плавающей точкой представляются в формате, описанном в стандарте ANSI/IEEE 754–1985 (одинарная точность). Подробную информацию об этих числах Вы найдете в этом стандарте.

Извлечение квадратного корня из действительного числа



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

$\sqrt{IN} = OUT$

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

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

Эти операции влияют на следующие специальные меркеры:

SM1.0 (нуль); SM1.1 (переполнение); SM1.2 (отрицательный)

Примеры арифметических операций

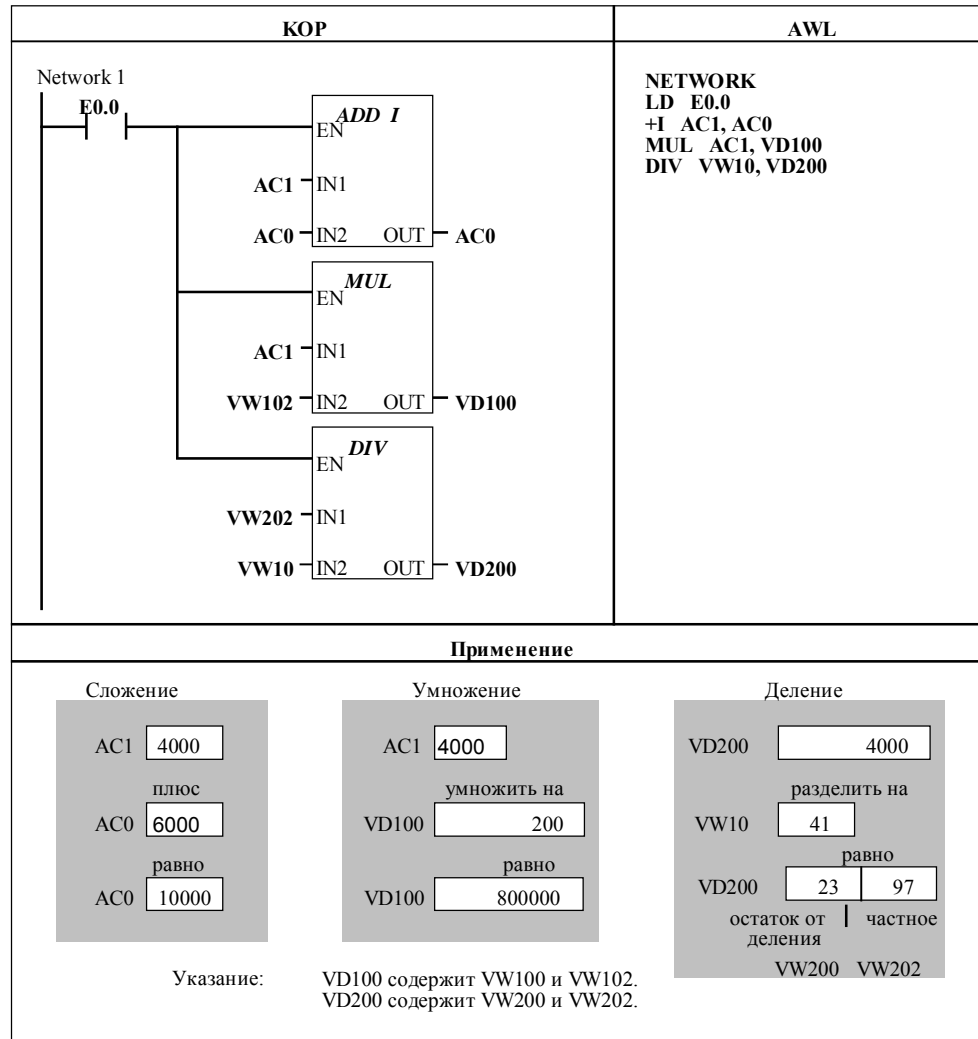
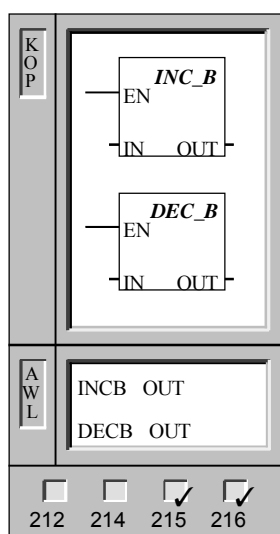


Рис. 9-7. Пример арифметических операций в KOP и AWL

Инкрементирование и декрементирование байта



Операции **Увеличить байт на 1** и **Уменьшить байт на 1** прибавляет или вычитает “1” из значения входного байта.

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

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

KOP: $IN + 1 = OUT$
 $IN - 1 = OUT$

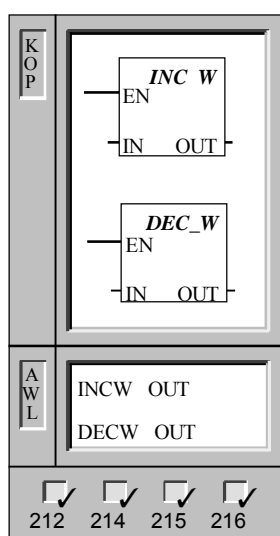
AWL: $OUT + 1 = OUT$
 $OUT - 1 = OUT$

Операции “Увеличить байт на 1” и “Уменьшить байт на 1” не учитывают знака.

Указание: При программировании в KOP Вы можете указать, что IN1 совпадает с OUT. Таким образом, Вы экономите место в памяти.

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

Инкрементирование и декрементирование слова



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

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

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

KOP: $IN + 1 = OUT$
 $IN - 1 = OUT$

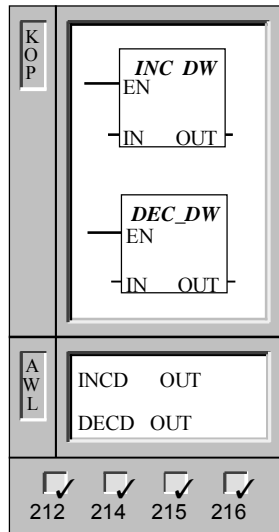
AWL: $OUT + 1 = OUT$
 $OUT - 1 = OUT$

Операции “Увеличить слово на 1” и “Уменьшить слово на 1” учитывают знак ($16\#7FFF > 16\#8000$).

Указание: При программировании в KOP Вы можете указать, что IN1 совпадает с OUT. Таким образом, Вы экономите место в памяти.

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

Инкрементирование и декрементирование двойного слова



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

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

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

KOP: $IN + 1 = OUT$
 $IN - 1 = OUT$

AWL: $OUT + 1 = OUT$
 $OUT - 1 = OUT$

Операции “Увеличить двойное слово на 1” и “Уменьшить двойное слово на 1” учитывают знак ($16\#7FFFFFFF > 16\#80000000$).

Указание: При программировании в KOP Вы можете указать, что IN1 совпадает с OUT. Таким образом, Вы экономите место в памяти.

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

Примеры инкрементирования и декрементирования

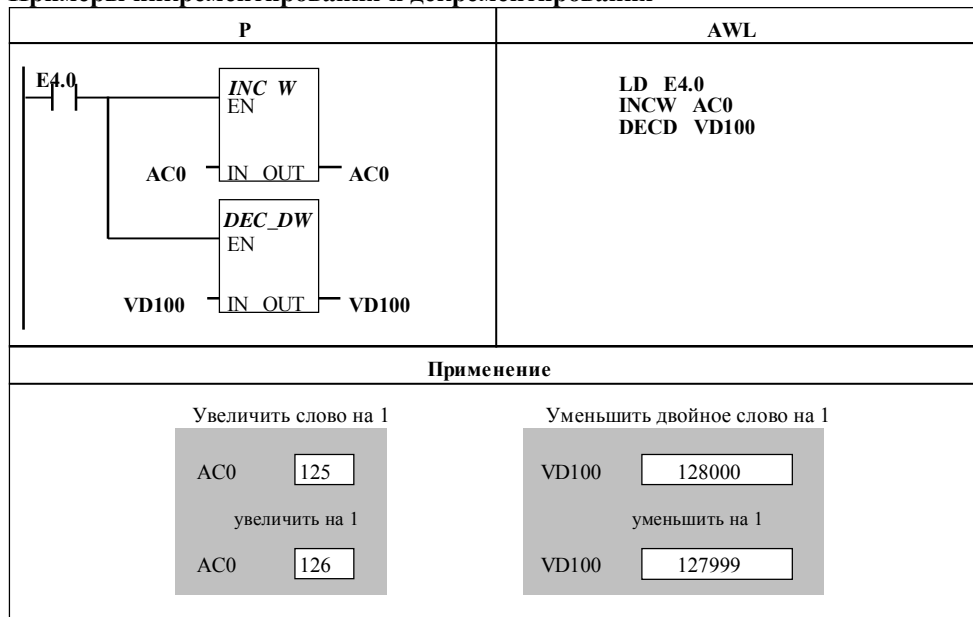
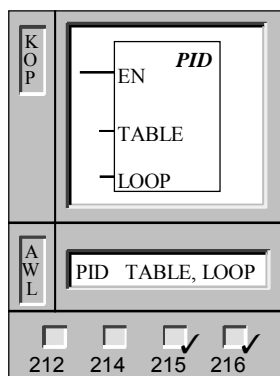


Рис. 9-8. Примеры инкрементирования и декрементирования в KOP и AWL

9.6 PID– операция

PID–регулятор



Операция **PID–регулятор** рассчитывает контур PID–регулирования для заданного контура регулирования LOOP с помощью информации о входных величинах и конфигурации в параметре TABLE.

Операнды: TABLE: VB
 LOOP: от 0 до 7

Эти операции влияют на следующие специальные меркеры:

SM1.1 (переполнение)

Операция “PID–регулятор” (пропорционально–интегрально–дифференциальный регулятор) служит для расчета PID–регулятора. Для того, чтобы могли выполняться расчеты PID–регулятора, должна быть активизирована вершина стека (существует поток сигнала).

Операция имеет два операнда: TABLE содержит начальный адрес таблицы контура регулирования и LOOP содержит номер контура регулирования, который является константой в диапазоне от 0 до 7. В программе разрешается использовать максимум восемь операций PID. Если две или более PID–операций используют одинаковый номер контура регулирования (даже если адреса таблиц у них различны), то вычисления для PID–регуляторов влияют друг на друга и результат будет непредсказуемым.

Таблица контура регулирования хранит девять параметров для управления и контроля контура регулирования. Эти параметры включают в себя текущее и предыдущее значение переменной процесса (фактическое значение), заданное значение, регулирующее воздействие, коэффициент усиления, интервал съема данных, время воздействия по интегралу (сброс), время воздействия по производной и интегральную сумму (смещение).

Чтобы PID–вычисления могли производиться с желаемым интервалом съема данных, операция PID должна выполняться либо в программе обработки прерываний, управляемых временем, либо в главной программе со скоростью, управляемой таймером. Частота съема данных должна предоставляться в качестве входа операции PID через таблицу контура регулирования.

PID-алгоритм

В установившемся режиме PID-регулятор воздействует на выходную величину так, чтобы свести ошибку регулирования (e) к нулю. Ошибка регулирования представляет собой разность между заданным значением и значением переменной процесса (фактическим значением). Принцип PID-регулятора базируется на следующем уравнении, представляющем регулирующее воздействие $M(t)$ как сумму пропорциональной, интегральной и дифференциальной составляющей:

$$M(t) = K_C * e + K_C \int_0^t e dt + M_{initial} + K_C * de/dt$$

Регулир. воздействие	=	Пропорциональная составляющая	+	Интегральная составляющая	+	Дифференциальная составляющая
----------------------	---	-------------------------------	---	---------------------------	---	-------------------------------

- $M(t)$ регулирующее воздействие в зависимости от времени
- K_C коэффициент усиления
- e ошибка регулирования (разность между заданным значением и фактическим значением)
- $M_{initial}$ начальное значение регулирующего воздействия

Чтобы использовать эту функцию управления в цифровом компьютере, непрерывная функция должна быть преобразована в дискретную форму путем периодического съема значения ошибки регулирования с последующим расчетом регулирующего воздействия. Соответствующее уравнение, представляющее основу для компьютерных вычислений, имеет вид:

$$M_n = K_C * e_n + K_I * \sum_1^n e + M_{initial} + K_D * (e_n - e_{n-1})$$

Регулир. воздейств.	=	Пропорциональная составляющая	+	Интегральная составляющая	+	Дифференциальная составляющая
---------------------	---	-------------------------------	---	---------------------------	---	-------------------------------

- M_n вычисляемое регулирующее воздействие в дискретный момент времени n
- K_C коэффициент усиления
- e_n значение ошибки регулирования в дискретный момент времени n
- e_{n-1} предыдущее значение ошибки регулирования (в дискретный момент времени $n - 1$)
- K_I коэффициент пропорциональности интегральной составляющей
- $M_{initial}$ начальное значение регулирующего воздействия
- K_D коэффициент пропорциональности дифференциальной составляющей

Из этого уравнения ясно, что интегральная составляющая представляет собой функцию всех ошибок регулирования от первого отсчета до текущего отсчета. Дифференциальная составляющая является функцией текущего и предыдущего отсчета, в то время как пропорциональная составляющая является функцией только текущего отсчета. В компьютере не является ни целесообразным, ни необходимым хранение всех ошибок регулирования.

Компьютер вычисляет регулирующее воздействие каждый раз, когда производится замер значения ошибки регулирования. Эти вычисления начинаются при первом замере. Поэтому должны храниться только предыдущее значение ошибки регулирования и предыдущее значение интегральной составляющей. Так как функции регулятора в цифровом компьютере постоянно повторяются, то можно упростить уравнение, решаемое при каждом замере ошибки. Ниже приводится упрощенное уравнение:

$$M_n = K_C * e_n + K_I * e_n + M_n + K_D * (e_n - e_{n-1})$$

Регулир. воздействие	=	Пропорциональная составляющая	+	Интегральная составляющая	+	Дифференциальная составляющая
----------------------	---	-------------------------------	---	---------------------------	---	-------------------------------

M_n	вычисляемое регулирующее воздействие в дискретный момент времени n
K_C	коэффициент усиления
e_n	значение ошибки регулирования в дискретный момент времени n
e_{n-1}	предыдущее значение ошибки регулирования (в дискретный момент времени $n-1$)
K_I	коэффициент пропорциональности интегральной составляющей
M_X	предыдущее значение интегральной составляющей (в дискретный момент времени $n-1$)
K_D	коэффициент пропорциональности дифференциальной составляющей

CPU использует видоизмененную форму представленного выше упрощенного уравнения для вычисления регулирующего воздействия в контуре регулирования. Ниже приводится видоизмененное уравнение:

M_n	=	$M_P n$	+	$M_I n$	+	$M_D n$
Регулир. воздействие.	=	Пропорциональная составляющая	+	Интегральная составляющая	+	Дифференциальная составляющая

M_n	вычисляемое регулирующее воздействие в дискретный момент времени n
$M_P n$	значение пропорциональной составляющей регулирующего воздействия в дискретный момент времени n
$M_I n$	значение интегральной составляющей регулирующего воздействия в дискретный момент времени n
$M_D n$	значение дифференциальной составляющей регулирующего воздействия в дискретный момент времени n

Пропорциональная составляющая

Пропорциональная составляющая M_P является произведением коэффициента усиления (K_C), задающего точность вычисления регулирующего воздействия, и ошибки (e), представляющей собой разность между заданным значением (SW) и фактическим значением (IW), т.е. значением переменной процесса в заданный дискретный момент времени. Используемое CPU уравнение для пропорциональной компоненты имеет следующий вид:

$$M_P n = K_C * (SW_n - IW_n)$$

$M_P n$	значение пропорциональной составляющей регулирующего воздействия в дискретный момент времени n
K_C	коэффициент усиления
SW_n	заданное значение в дискретный момент времени n
IW_n	фактическое значение (значение переменной процесса) в дискретный момент времени n

Интегральная составляющая

Интегральная составляющая M_I пропорциональна сумме ошибок регулирования по времени. Используемое CPU уравнение для интегральной составляющей имеет следующий вид:

$$M_I n = K_C * T_S / T_I * (SW_n - IW_n) + M_X$$

$M_I n$	значение интегральной составляющей регулирующего воздействия в дискретный момент времени n
K_C	коэффициент усиления
T_S	интервал съема данных в контуре регулирования
T_I	время изотропа контура регулирования
SW_n	заданное значение в дискретный момент времени n
IW_n	фактическое значение (значение переменной процесса) в дискретный момент времени n
M_X	значение интегральной составляющей в дискретный момент времени $n-1$ (называемое также интегральной суммой или смещением)

Интегральная сумма или смещение (MX) представляет собой текущую сумму всех предыдущих значений интегральной составляющей. После каждого вычисления M_{I_n} интегральная сумма актуализируется значением M_{I_n} . При этом речь может идти о согласовании или ограничении (подробную информацию об этом Вы найдете в разделе "Переменные и диапазоны"). Начальное значение интегральной суммы обычно устанавливается равным значению регулирующего воздействия ($M_{initial}$) непосредственно перед первым вычислением регулирующего воздействия для контура регулирования. Интегральная составляющая содержит различные константы: коэффициент усиления (K_C), интервал съема данных (T_S) и время изодрома (T_I). Интервал съема данных представляет собой период времени, в течение которого PID-регулятор вычисляет новое значение регулирующего воздействия. Время изодрома представляет собой время, посредством которого регулируется влияние интегральной составляющей при вычислении регулирующего воздействия.

Дифференциальная составляющая

Дифференциальная составляющая MD пропорциональна изменению ошибки регулирования. Уравнение для дифференциальной составляющей имеет следующий вид:

$$MD_n = K_C * T_D / T_S * ((SW_n - IW_n) - (SW_{n-1} - IW_{n-1}))$$

Чтобы при изменениях заданного значения избежать скачков в регулирующем воздействии из-за влияния производной, в данном уравнении принимается, что заданное значение является константой ($SW_n = SW_{n-1}$). Поэтому вычисляется изменение фактического значения (переменной процесса), а не ошибки регулирования. Это показывает следующее уравнение:

$$MD_n = K_C * T_D / T_S * (SW_n - IW_n - SW_{n-1} + IW_{n-1})$$

или также:

$$MD_n = K_C * T_D / T_S * (IW_{n-1} - IW_n)$$

MD_n	значение составляющей регулирующего воздействия в дискретный момент времени n
K_C	коэффициент усиления
T_S	интервал съема данных в контуре регулирования
T_D	время упреждения контура регулирования (называется также постоянной времени дифференцирующего звена)
SW_n	заданное значение в дискретный момент времени n
SW_{n-1}	заданное значение в дискретный момент времени n - 1
IW_n	фактическое значение (значение переменной процесса) в дискретный момент времени n
IW_{n-1}	фактическое значение (значение переменной процесса) в дискретный момент времени n - 1

Для вычисления следующего значения дифференциальной составляющей должно запоминаться фактическое значение, а не ошибка регулирования. В момент первого съема данных значение IW_{n-1} инициализируется значением IW_n .

Выбор регулятора

Во многих устройствах регулирования часто нужны только один или два разных регулятора. Например, может использоваться только пропорциональный регулятор или могут использоваться один пропорциональный и один интегральный регулятор. Вы можете выбирать необходимые регуляторы, устанавливая постоянные параметры на определенное значение.

Если Вам не нужно воздействие по интегралу (нет I-составляющей в PID-вычислениях), то Вы должны задать бесконечную величину для времени воздействия по интегралу.

Если Вам не нужно воздействие по производной (нет D-составляющей в PID-вычислениях), то Вы должны задать значение 0,0 для времени воздействия по производной.

И при отсутствии воздействия по интегралу интегральная составляющая не будет равна нулю из-за начального значения интегральной суммы MX.

Если Вам не нужно пропорциональное воздействие (нет P-составляющей в PID-вычислениях), а нужен только I- или ID-регулятор, то Вы должны задать значение 0,0 для коэффициента усиления. Коэффициент усиления в контуре регулирования является множителем в уравнениях для вычисления интегральной и дифференциальной составляющей. Следовательно, если Вы задаете значение 0,0 для коэффициента усиления, то при вычислении интегральной и дифференциальной составляющей в качестве коэффициента усиления используется значение 1,0.

Преобразование и нормализация входных значений

Контур регулирования имеет две входные переменные, заданное значение и фактическое значение (переменная процесса). Заданное значение обычно представляет собой фиксированное значение, например, уставку скорости для регулятора скорости движения в легковом автомобиле. Переменная процесса представляет собой значение, которое связано с регулирующим воздействием контура регулирования и поэтому является мерой влияния, оказываемого регулирующим воздействием на регулируемую систему. В примере с регулятором скорости движения автомобиля переменная процесса является входной величиной тахометра, измеряющего скорость вращения колес.

Оба значения - заданное и фактическое - являются аналоговыми значениями, величина, диапазон и единица измерения которых может быть разной. Прежде чем эти значения могут быть использованы операцией PID, они должны быть преобразованы в нормализованное представление с плавающей точкой.

Для этого аналоговое значение, представленное как целое число (16 бит), нужно преобразовать сначала в число с плавающей точкой, или в действительное число. Следующие команды показывают, как можно преобразовать целое число в действительное число:

```
XORD  AC0, AC0           // Очистка аккумулятора.
MOVW  AEW0, AC0         // Запись аналогового значения в аккумулятор LDW>=
      AC0, 0            // Если аналоговое значение положительно, то
JMP    0                // преобразовать в действительное число.
NOT    0                // В противном случае,
ORD    16#FFFF0000, AC0 // снабдить значение в AC0 знаком.
LBL    0
DTR    AC0, AC0         // Преобразовать целое число (32 бита) в
                        // действительное число.
```

На следующем шаге нужно действительное число, представляющее аналоговое значение, преобразовать в нормализованное значение в диапазоне от 0,0 до 1,0. Нормализация заданного значения или значения переменной процесса производится с помощью следующего уравнения:

$$R_{Norm} = (R_{ur} / Spanne) + Versatz$$

R_{Norm} нормализованное действительное число, представляющее аналоговое значение

R_{ur} ненормализованное действительное число, представляющее аналоговое значение

$Versatz$ 0,0 для униполярных значений
(смещение) 0,5 для биполярных значений

$Spanne$ разность между максимально возможным значением и минимально возможным значением

32000 для униполярных значений (типовой случай)

64000 для биполярных значений (типовой случай)

Следующие команды показывают, как можно нормализовать биполярное значение в AC0 (для которого разность $Spanne$ равна 64000) в дополнение к представленным выше командам:

```
/R    64000,0, AC0      // Нормализация значения в аккумуляторе
+R    0,5, AC0         // Смещение значения в диапазон 0,0 - 1,0
MOVW  AC0, VD100      // Запись нормализованного значения в TABLE
```

Преобразование регулирующего воздействия контура регулирования в масштабированное целочисленное значение

Регулирующее воздействие - это управляемая переменная, как, например, дроссельный клапан в регуляторе скорости движения легкового автомобиля. Регулирующее воздействие представляет собой нормализованное действительное число в диапазоне от 0,0 до 1,0. Прежде чем станет возможным регулировать аналоговый выход с помощью регулирующего воздействия, нужно преобразовать регулирующее воздействие в масштабированное целочисленное значение (16 бит). Этот процесс противоположен преобразованию заданного или фактического значения в нормализованное. Сначала нужно преобразовать регулирующее воздействие в масштабированное действительное число. Для этого используется следующее уравнение:

$$R_{Skal} = (M_n - \text{Versatz}) * \text{Spanne}$$

R_{Skal} масштабированное действительное значение регулирующего воздействия

M_n нормализованное действительное значение регулирующего воздействия

Versatz 0,0 для униполярных значений
(смещение) 0,5 для биполярных значений

Spanne разность между максимально возможным и минимально возможным значением

32000 для униполярных значений (типовой случай)

64000 для биполярных значений (типовой случай)

Следующие команды показывают, как нужно масштабировать регулирующее воздействие:

```
MOVVR VD108, AC0 // Передача регулирующего воздействия в
// аккумулятор
-R 0,5, AC0 // Команда, задаваемая только для
// биполярных значений
*R 64000,0, AC0 // Масштабирование значения в аккумуляторе
```

После этого нужно масштабированное действительное число, представляющее регулирующее воздействие, преобразовать в целое число (16 бит). Следующие команды показывают, как выполнить такое преобразование:

```
TRUNC AC0, AC0 // Преобразование действительного числа в
// целое число (32 бита)
MOVW AC0, AAW0 // Запись целого числа (32 бита) на аналоговый
// выход
```

Контур регулирования прямого и обратного действия

Контур регулирования имеет прямое действие, если его коэффициент усиления положителен, и обратное действие, если коэффициент усиления отрицателен. (Для I- или ID-регулятора с коэффициентом усиления 0,0 прямое действие имеет место при положительном времени воздействия по интегралу и по производной. При задании отрицательных значений для этих времен получается контур регулирования обратного действия).

Переменные и диапазоны

Переменная процесса (фактическое значение) и заданное значение являются входными величинами в PID-вычислениях. Поэтому эти поля таблицы контура регулирования считываются, но не изменяются операцией PID.

Регулирующее воздействие вычисляется PID-регулятором, так что поле регулирующего воздействия в таблице контура регулирования актуализируется после каждого PID-вычисления. Регулирующее воздействие фиксируется в диапазоне от 0,0 до 1,0. Поле регулирующего воздействия может использоваться в качестве входного значения для начального регулирующего воздействия, когда нужно перейти от ручного регулирования к автоматическому регулированию с помощью PID-регулятора (подробную информацию по этому вопросу Вы получите в следующем разделе по режимам работы).

Если используется интегральный регулятор, то значение интегральной суммы актуализируется в PID-вычислении, и актуализированная интегральная сумма используется в качестве входной величины в следующем PID-вычислении. Если рассчитанное регулирующее воздействие выходит за пределы диапазона \square то есть регулирующее воздействие меньше, чем 0,0, или больше, чем 1,0), то интегральная сумма адаптируется по следующему уравнению:

$$MX = 1,0 - (MP_n + MD_n) \quad \text{если рассчитанное регулирующее воздействие } M_n > 1.0$$

или

$$MX = - (MP_n + MD_n) \quad \text{если рассчитанное регулирующее воздействие } M_n < 0.0$$

MX значение адаптированной интегральной суммы
 MP_n значение пропорциональной составляющей регулирующего воздействия в дискретный момент времени n
 MD_n значение дифференциальной составляющей регулирующего воздействия в дискретный момент времени n
 M_n значение регулирующего воздействия в дискретный момент времени n

Если интегральная сумма адаптируется, как описано, то чувствительность системы улучшается, когда вычисляемое регулирующее воздействие возвращается в допустимый диапазон. Вычисляемая интегральная сумма также фиксируется в диапазоне от 0,0 до 1,0 и записывается в поле интегральной суммы таблицы контура регулирования после выполнения PID-вычисления. Значение, записанное в таблицу контура регулирования, используется для следующего PID-вычисления. Вы можете изменить значение интегральной суммы в таблице контура регулирования перед выполнением операции PID, чтобы таким образом через интегральную сумму воздействовать на определенные ситуации в различных приложениях. Однако действуйте осторожно, когда Вы адаптируете интегральную сумму вручную. Для каждого значения, записываемого в качестве интегральной суммы в таблицу контура регулирования, речь должна идти о действительном числе в диапазоне от 0,0 до 1,0. Для переменных процесса в таблице хранится опорное значение, которое может использоваться для дифференциальной составляющей PID-регулятора. Вам нельзя изменять это значение.

Режимы работы

Для контуров PID-регулирования S7-200 не существует встроенного управления режимами работы. PID-вычисление активизируется потоком сигнала у блока PID. Поэтому PID-вычисления выполняются в автоматическом режиме циклически. В ручном режиме PID-вычисления не выполняются.

Операция PID имеет бит предыстории для запоминания предыдущего состояния сигнала, подобный тому, какой имеется в счетчиках. С помощью этого бита операция распознает нарастающий фронт. При нарастающем фронте операция выполняет ряд шагов для плавного перехода от ручного режима к автоматическому. Для плавности перехода к автоматическому режиму операции PID в качестве входной величины должно предоставляться значение регулирующего воздействия ручного режима (внесенное в таблицу контура регулирования как значение для M_n) до того, как происходит переход на автоматический режим. Для того, чтобы обеспечить плавный переход с ручного режима на автоматический при нарастающем фронте операция PID обрабатывает значения в таблице контура регулирования следующим образом:

- Заданное значение (SW_n) устанавливается равным фактическому значению (IW_n).
- Старое фактическое значение (IW_{n-1}) устанавливается равным фактическому значению (IW_n).
- Интегральная сумма (MX) устанавливается равной регулирующему воздействию (M_n).

По умолчанию бит предыстории PID "установлен". Такое состояние создается при запуске CPU и при каждой смене режима работы CPU со STOP на RUN. Если после перехода в режим работы RUN поток сигнала у блока PID появляется в первый раз, то фронт в сигнальном токе не распознается, а также не выполняются никакие шаги для плавного перехода к автоматическому режиму.

Сигналы прерываний, специальные операции и т.д.

Операция PID представляет собой простую и тем не менее очень мощную операцию для PID-вычисления. Если потребуются другие виды обработки, например, обработка прерываний или специальные вычисления переменных в контуре регулирования, то Вы должны реализовать эти функции посредством операций, поддерживаемых Вашим CPU.

Сбойные ситуации

При компиляции CPU сообщает о возникновении ошибки компиляции, если параметр операции, начальный адрес таблицы контура регулирования или номер контура PID-регулирования лежит вне допустимого диапазона. Тогда компиляция не является успешной.

Некоторые из входных величин в таблице контура регулирования не проверяются операцией PID по их диапазону. Поэтому Вы должны обращать внимание на то, чтобы переменные процесса/фактические значения и заданные значения (а также интегральная сумма и предыдущие значения переменных процесса, если они используются в качестве входных величин) были действительными числами в диапазоне от 0,0 до 1,0.

Если при выполнении арифметических операций в PID-вычислениях обнаруживается ошибка, то устанавливается специальный меркер SM1.1 (переполнение или недопустимое значение) и выполнение операции PID заканчивается.

Таблица контура регулирования

Таблица контура регулирования имеет длину 36 байтов и имеет следующий формат:

Таблица 9-4. Формат таблицы контура регулирования

Смещение	Поле	Формат	Тип данных	Описание
0	Переменная процесса/ фактическое значение (IW_n)	Двойное слово - Действительное число	In	Содержит фактическое значение или переменную процесса, которая должна быть масштабирована в диапазоне от 0,0 до 1,0.
4	Заданное значение (SW_n)	Двойное слово - Действительное число	In	Содержит заданное значение, которое должно быть масштабировано в диапазоне от 0,0 до 1,0.
8	Регулирующее воздействие (M_n)	Двойное слово - Действительное число	In/Out	Содержит вычисляемое регулирующее воздействие, которое должно быть масштабировано в диапазоне от 0,0 до 1,0.
12	Коэффициент усиления (K_C)	Двойное слово - Действительное число	In	Содержит коэффициент усиления <input type="checkbox"/> коэффициент пропорциональности). Он может быть положительным или отрицательным.
16	Период дискретизации (T_S)	Двойное слово - Действительное число	In	Содержит интервал съема информации в секундах. Это значение должно быть положительным.
20	Время изодрома (T_I)	Двойное слово - Действительное число	In	Содержит время изодрома в минутах. Это значение должно быть положительным.
24	Время упреждения (T_D)	Двойное слово - Действительное число	In	Содержит время упреждения в минутах. Это значение должно быть положительным.
28	Интегральная сумма/Смещение (MX)	Двойное слово - Действительное число	In/Out	Содержит интегральную сумму или смещение в диапазоне от 0,0 до 1,0.
32	Предыдущее фактическое значение /Переменная процесса (IW_{n-1})	Двойное слово - Действительное число	In/Out	Содержит предыдущее значение переменной процесса или предыдущее фактическое значение с момента последнего выполнения операции PID.

Пример программирования операции PID

В этом примере нужно в резервуаре поддерживать постоянное давление воды. Из резервуара производится забор воды постоянно, однако с переменной скоростью. Насос с переменным числом оборотов привода закачивает воду в резервуар так, чтобы обеспечить требуемое давление воды в резервуаре и не допустить опустошения резервуара.

Заданным значением для данной системы является установка уровня воды, эквивалентного заполнению резервуара на 75%. Переменная процесса (фактическое значение) поставляется поплавком, показывающим количество воды в резервуаре. Данное фактическое значение может лежать в диапазоне от 0% до 100%. Регулирующее воздействие представляет собой значение в диапазоне от 0% до 100% для числа оборотов привода насоса.

Заданное значение заранее фиксируется и записывается непосредственно в таблицу контура регулирования. Переменная процесса (фактическое значение) поставляется поплавком в виде униполярного аналогового значения. Регулирующее воздействие записывается в виде униполярного аналогового значения и служит для регулирования числа оборотов насоса. Величина разности Spanne для аналогового входа и для аналогового выхода равна 32000.

В данном примере используются только пропорциональный и интегральный регуляторы. Коэффициент усиления в контуре регулирования и константы времени были определены расчетным путем и могут адаптироваться, чтобы добиться оптимального регулирования.

Расчетными значениями коэффициента усиления и постоянных времени являются следующие:

K_C - 0,25

T_S - 0,1 секунды

T_I - 30 минут

Скорость насоса регулируется вручную до тех пор, пока уровень воды в резервуаре не составит 75%. Потом открывается вентиль, чтобы вода вытекала из резервуара. Одновременно насос переключается с ручного режима на автоматический. Для перехода с ручного режима на автоматический используется цифровой вход. Этот вход определен следующим образом:

E0.0 - ручной/автоматический режим; 0 - ручной, 1 - автоматический

В ручном режиме оператор записывает число оборотов насоса в виде действительного числа в диапазоне от 0,0 до 1,0 в VD108 .

Ниже приводится программа для данного приложения:

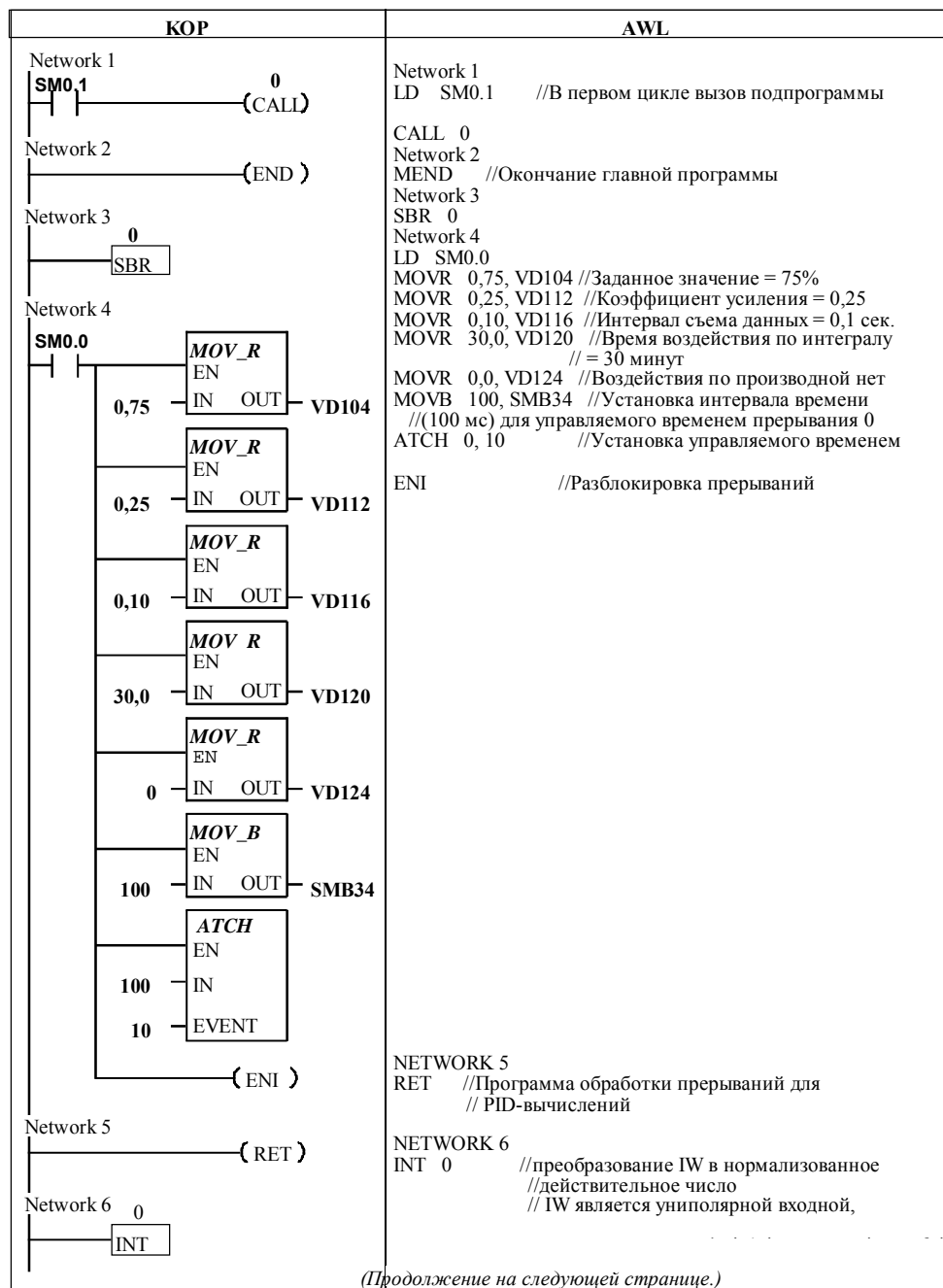


Рис. 9-9. Пример PID-регулятора

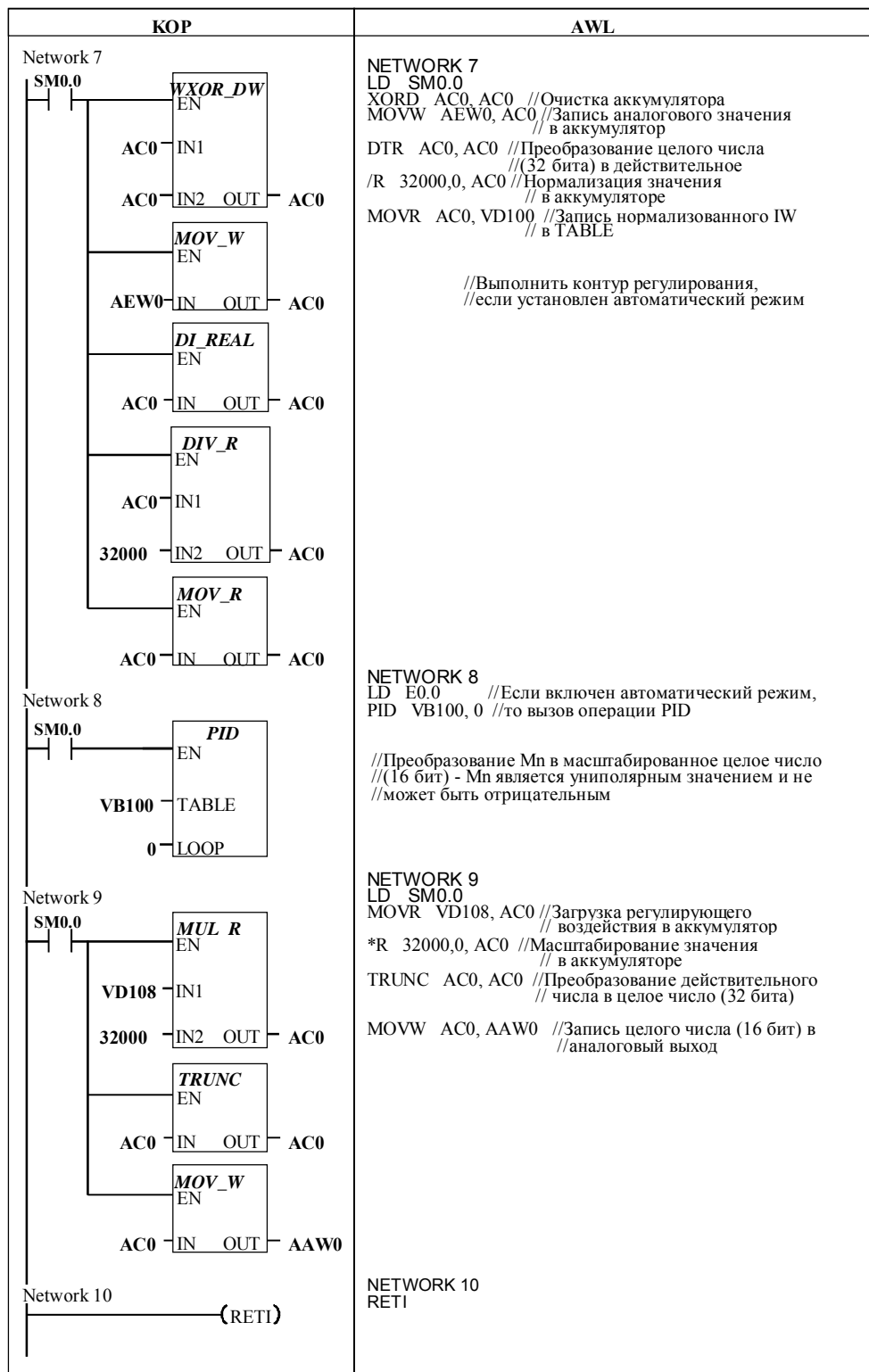
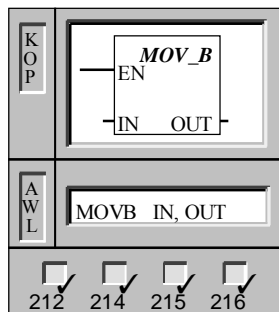


Рис. 9-10. Пример PID-регулятора, продолжение

9.7 Операции передачи, сдвига и циклического сдвига

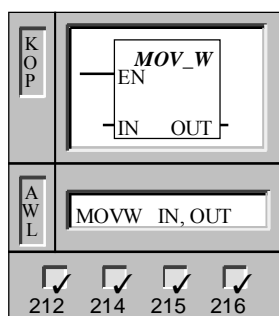
Передача байта



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

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

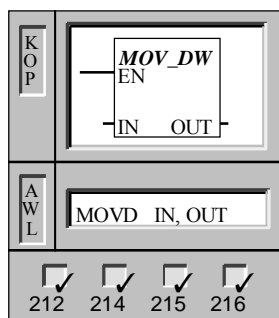
Передача слова



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

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

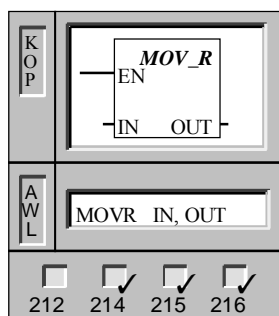
Передача двойного слова



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

Операнды: IN: VD, ED, AD, MD, SMD, AC,
HC, константа, *VD, *AC,
&VB, &EB, &AB, &MB, &T, &Z,
&SB, SD
OUT: VD, ED, AD, MD, SMD, AC, *VD,
*AC, SD

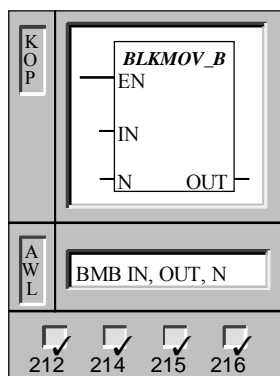
Передача действительного числа



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

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

Блочная передача байтов

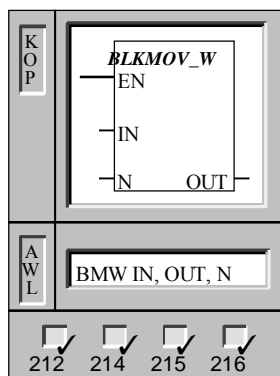


Операция **Блочная передача байтов** передает заданное количество байтов (N) из входного массива, начинающегося с IN, в выходной массив, начинающийся с OUT. N может лежать в диапазоне от 1 до 255.

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

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

Блочная передача слов

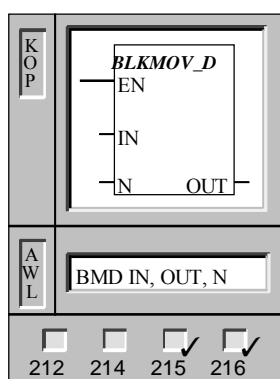


Операция **Блочная передача слов** передает заданное количество слов (N) из входного массива, начинающегося с IN, в выходной массив, начинающийся с OUT. N может лежать в диапазоне от 1 до 255.

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

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

Блочная передача двойных слов

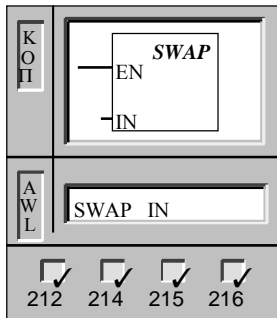


Операция **Блочная передача двойных слов** передает заданное количество двойных слов (N) из входного массива, начинающегося с IN, в выходной массив, начинающийся с OUT. N может лежать в диапазоне от 1 до 255.

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

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

Обмен байтов в слове



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

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

Примеры операций передачи и обмена

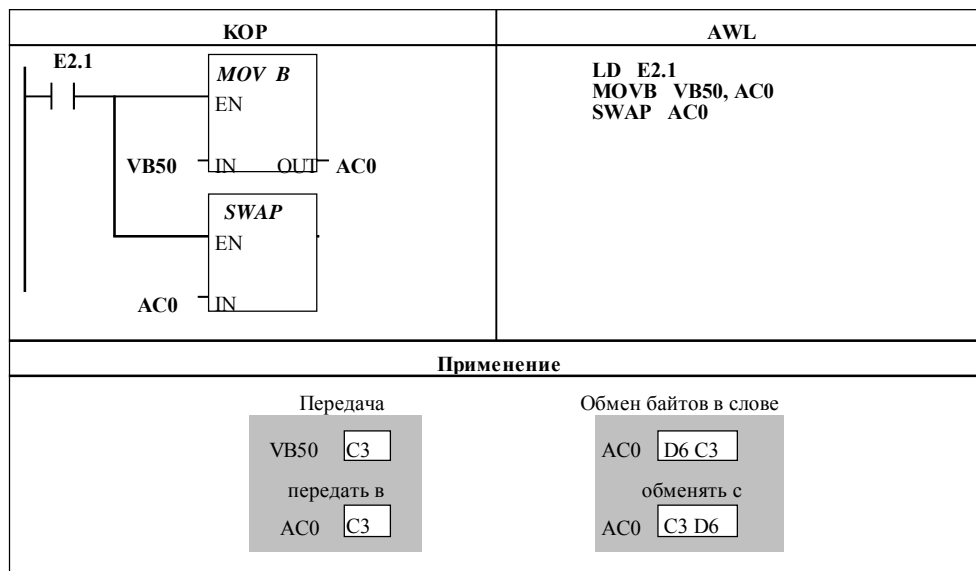


Рис. 9-11. Примеры операций передачи и обмена в форме KOP и AWL

Пример операции блочной передачи

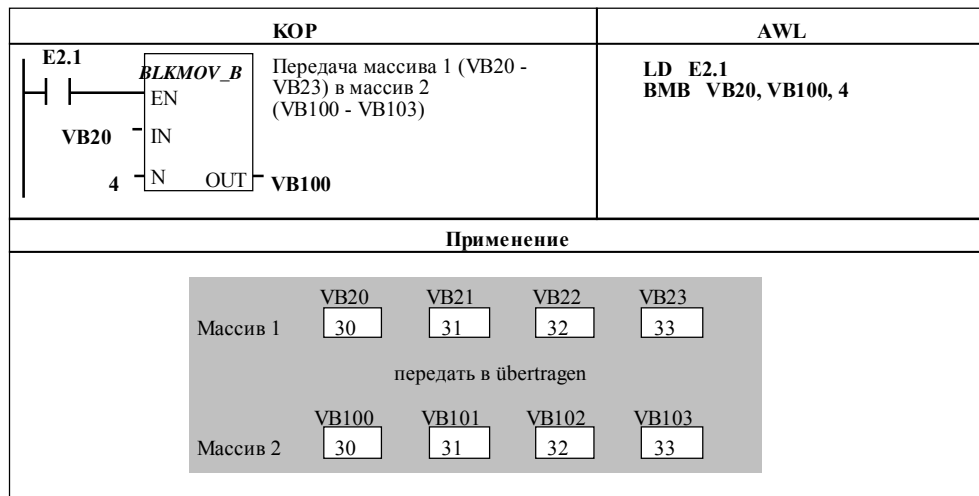
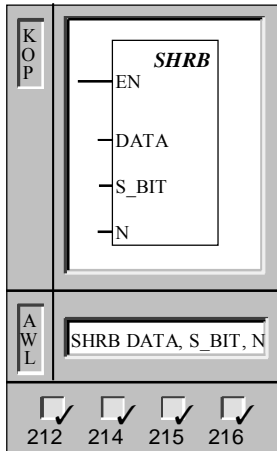


Рис. 9-12. Пример операции блочной передачи в форме KOP и AWL

Ввод значения в регистр сдвига

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

Операнды: DATA, S_BIT: E, A, M, SM, T, Z, V, S
 N: VB, EB, AB, MB, SMB, AC, константа, *VD, *AC, SB

Описание операции “Ввод значения в регистр сдвига”

Операция “Ввод значения в регистр сдвига” предоставляет простой способ упорядочивания потока изделий или данных и управления ими. С помощью операции “Ввод значения в регистр сдвига” Вы можете один раз за цикл сдвигать на один бит весь регистр сдвига. Регистр сдвига определяется младшим битом (S_BIT) и количеством битов, задаваемых длиной (N). На рис. 9–14 показан пример операции “Ввод значения в регистр сдвига”.

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

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

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

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

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

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

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

“Выдвигаемые” данные записываются в меркер переполнения (SM1.1). Регистр сдвига имеет максимальную длину 64 бита (положительную или отрицательную). На рис. 9–13 показаны функции сдвига с положительным и отрицательным значениями длины N.

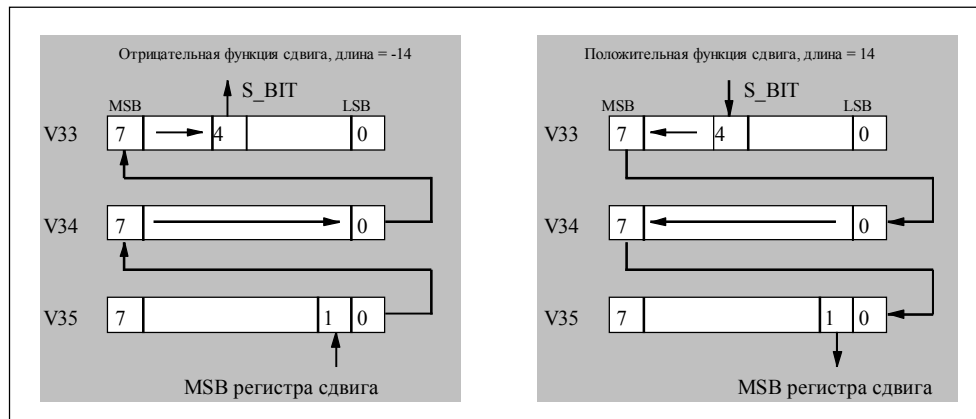


Рис. 9-13. Положительная и отрицательная функции сдвига в регистре сдвига

Пример операции “Ввод значения в регистр сдвига”

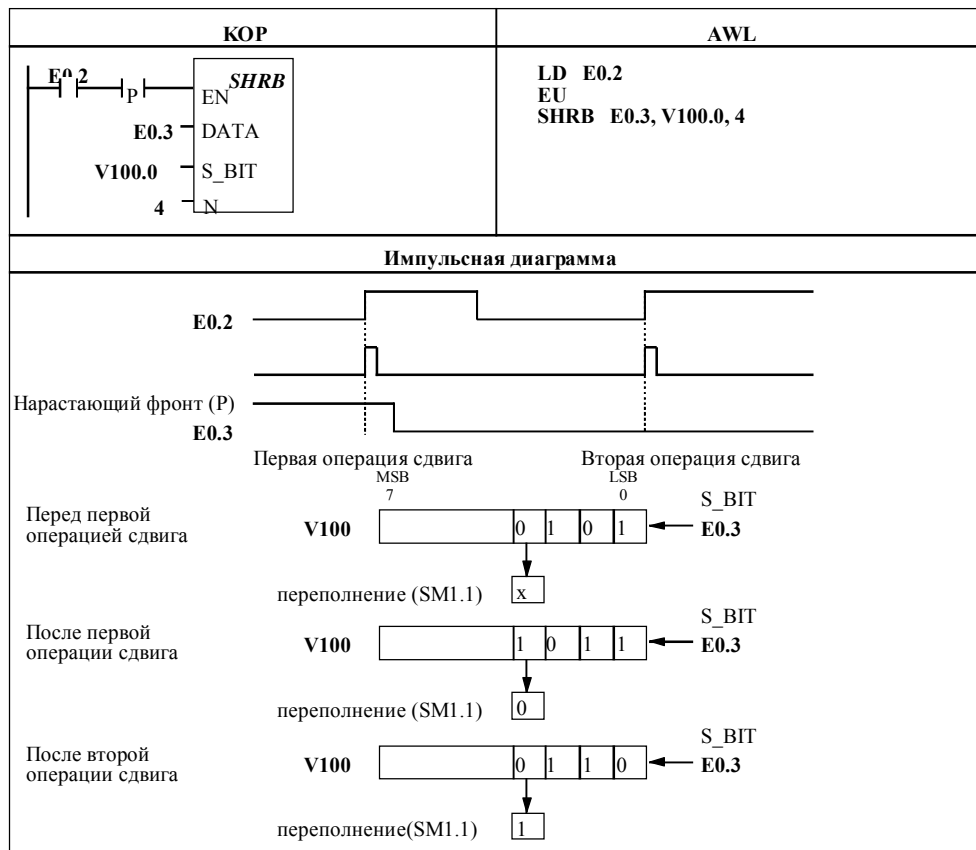
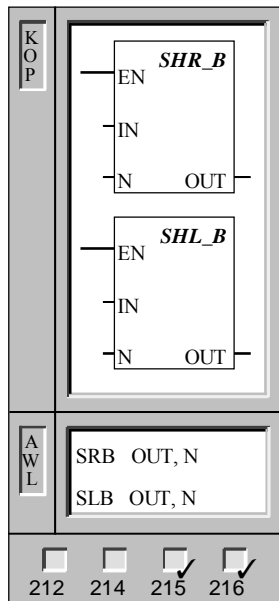


Рис. 9-14. Пример операции “Ввод значения в регистр сдвига” в KOP и AWL

Сдвиг байта вправо и сдвиг байта влево



Операции **Сдвиг байта вправо** и **Сдвиг байта влево** сдвигают значение байта (IN) на величину сдвига (N) вправо или влево и загружают результат в выходной байт (OUT).

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

Эти операции сдвига заполняют места “выдвигаемых” битов нулями.

Если величина сдвига (N) больше или равна 8 (байт), то значение сдвигается максимум 8 раз. Если величина сдвига больше нуля, то меркер переполнения принимает значение “выдвигаемого” бита.

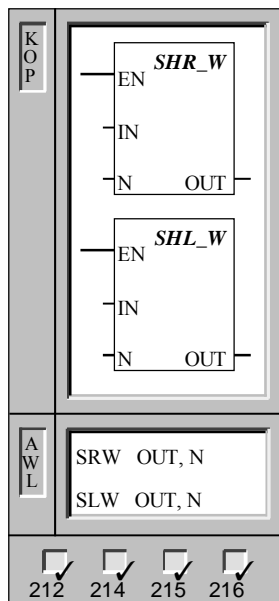
Операции “Сдвиг байта вправо” и “Сдвиг байта влево” не учитывают знака.

Указание: При программировании в KOP Вы можете указать, что IN совпадает с OUT. Таким образом, Вы экономите место в памяти.

Эти операции влияют на следующие специальные меркеры:

SM1.0 (нуль); SM1.1 (переполнение)

Сдвиг слова вправо и сдвиг слова влево



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

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

Эти операции сдвига заполняют места “выдвигаемых” битов нулями.

Если величина сдвига (N) больше или равна 16 (слово), то значение сдвигается максимум 16 раз. Если величина сдвига больше нуля, то меркер переполнения принимает значение “выдвигаемого” бита.

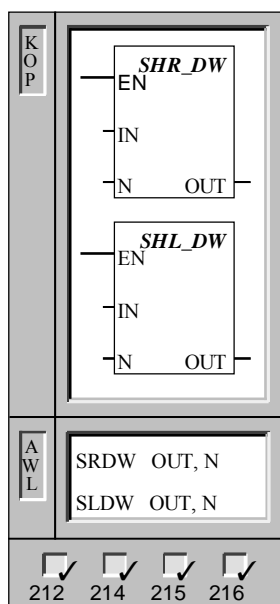
Операции “Сдвиг слова вправо” и “Сдвиг слова влево” не учитывают знака.

Указание: При программировании в KOP Вы можете указать, что IN совпадает с OUT. Таким образом, Вы экономите место в памяти.

Эти операции влияют на следующие специальные меркеры:

SM1.0 (нуль); SM1.1 (переполнение)

Сдвиг двойного слова вправо и сдвиг двойного слова влево



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

Операнды: IN: VD, ED, AD, MD, SMD, AC, HC,
 константа, *VD, *AC, SD
 N: VB, EB, AB, MB, SMB, AC,
 константа, *VD, *AC, SB
 OUT: VD, ED, AD, MD, SMD, AC, *VD,
 *AC, SD

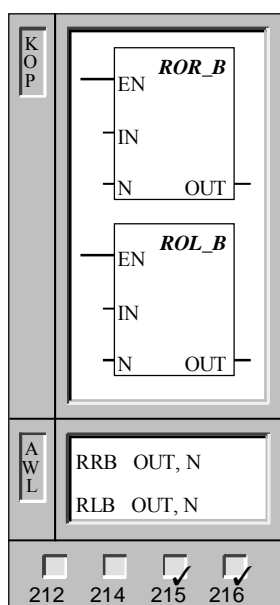
Эти операции сдвига заполняют места “выдвигаемых” битов нулями. Если величина сдвига (N) больше или равна 32 (двойное слово), то значение сдвигается максимум 32 раза. Если величина сдвига больше нуля, то меркер переполнения принимает значение “выдвигаемого” бита.

Операции “Сдвиг двойного слова вправо” и “Сдвиг двойного слова влево” не учитывают знака.

Указание: При программировании в KOP Вы можете указать, что IN совпадает с OUT. Таким образом, Вы экономите место в памяти.

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

Циклический сдвиг байта вправо и циклический сдвиг байта влево



Операции **Циклический сдвиг байта вправо** и **Циклический сдвиг байта влево** циклически сдвигают значение байта (IN) на величину сдвига (N) вправо или влево и загружают результат в выходной байт (OUT).

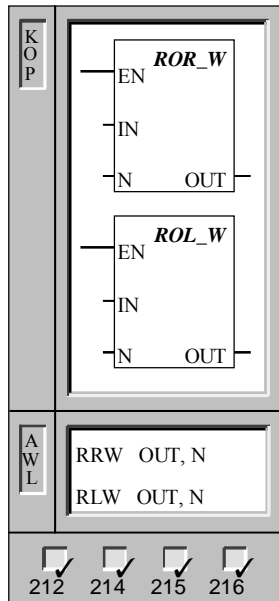
Операнды: IN: VB, EB, AB, MB, SMB, SB, AC,
 *VD, *AC, SB
 N: VB, EB, AB, MB, SMB, SB, AC,
 константа, *VD, *AC, SB
 OUT: VB, EB, AB, MB, SMB, SB, AC,
 *VD, *AC, SB

Если величина сдвига (N) больше или равна 8 (байт), то перед циклическим сдвигом выполняется операция по модулю 8. В результате получается величина сдвига в диапазоне от 0 до 7. Если величина сдвига равна нулю, то циклический сдвиг не происходит. Если циклический сдвиг происходит, то значение циклически “выдвигаемого” бита копируется в меркер переполнения.

Операции “Циклический сдвиг байта вправо” и “Циклический сдвиг байта влево” не учитывают знака.

Указание: При программировании в KOP Вы можете указать, что IN совпадает с OUT. Таким образом, Вы экономите место в памяти.

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

Циклический сдвиг слова вправо и циклический сдвиг слова влево

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

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

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

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

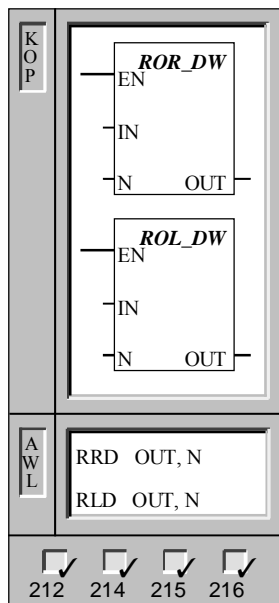
Если величина сдвига (N) больше или равна 16 (слово), то перед циклическим сдвигом выполняется операция по модулю 16. Отсюда получается величина сдвига в диапазоне от 0 до 15. Если величина сдвига равна нулю, то циклический сдвиг не происходит. Если циклический сдвиг происходит, то значение циклически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

Операции “Циклический сдвиг слова вправо” и “Циклический сдвиг слова влево” не учитывают знака.

Указание: При программировании в KOP Вы можете указать, что IN совпадает с OUT. Таким образом, Вы экономите место в памяти.

Эти операции влияют на следующие специальные меркеры:

SM1.0 (нуль); SM1.1 (переполнение)

Циклический сдвиг двойного слова вправо и циклический сдвиг двойного слова влево

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

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

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

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

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

Операции “Циклический сдвиг двойного слова вправо” и “Циклический сдвиг двойного слова влево” не учитывают знака.

Указание: При программировании в KOP Вы можете указать, что IN совпадает с OUT. Таким образом, Вы экономите место в памяти.

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

Примеры операций сдвига и циклического сдвига

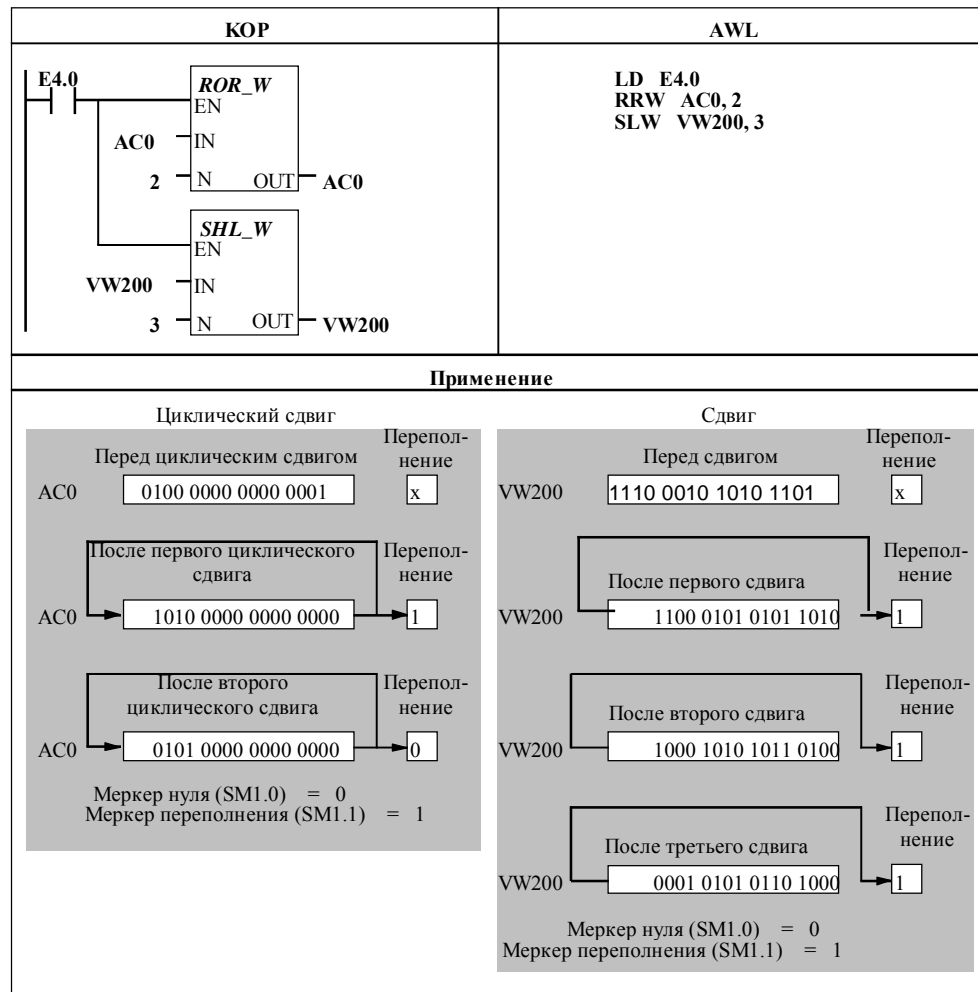
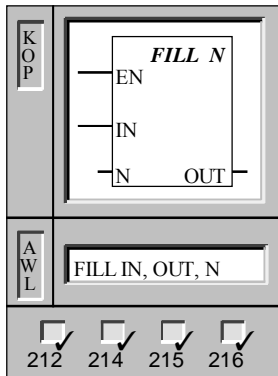


Рис. 9-15. Примеры операций сдвига и циклического сдвига в форме KOP и AWL

Заполнение памяти битовой комбинацией



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

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

Пример заполнения памяти битовой комбинацией

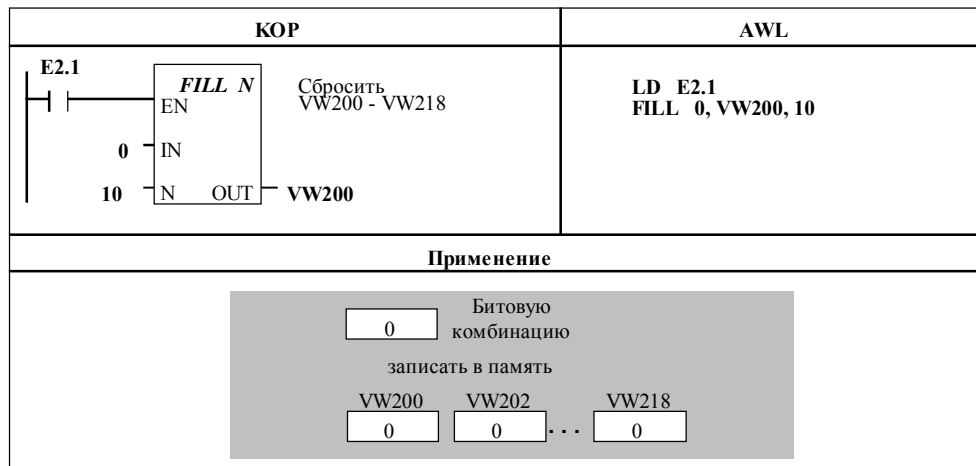
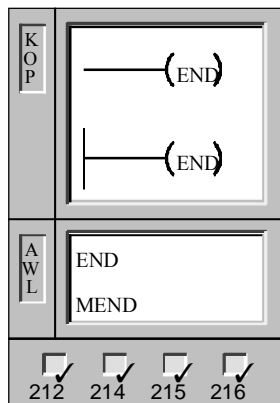


Рис. 9-16. Пример заполнения памяти битовой комбинацией в форме KOP и AWL

9.8 Операции управления программой

END



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

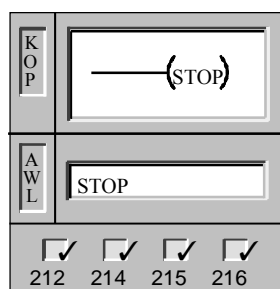
Главная программа должна заканчиваться катушкой **Абсолютное окончание обработки**.

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

Операнды: нет

Все программы пользователя должны завершать главную программу операциями “Абсолютное окончание обработки”. Операция “Условное окончание обработки” используется, когда обработка должна завершаться до операции “Абсолютное окончание обработки”.

STOP

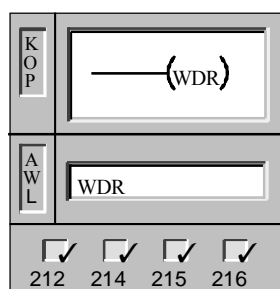


Операция **STOP** заканчивает обработку программы пользователя немедленно, переводя CPU из режима работы RUN в режим работы STOP.

Операнды: нет

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

Сброс времени контроля



Посредством операции **Сброс времени контроля** можно перезапустить время контроля CPU. Благодаря этому удлиняется максимально допустимое время цикла без сообщения об ошибке времени контроля.

Операнды: нет

Указания по сбросу времени контроля с помощью операции WDR

Используйте операцию “Сброс времени контроля” с осторожностью. Если Вы посредством программных циклов препятствуете завершению цикла работы контроллера или чрезмерно замедляете его, то до завершения цикла не могут выполняться следующие процессы :

- Коммуникация (кроме свободно программируемой коммуникации)
- Актуализация входов и выходов (кроме случая прямого управления входами и выходами)
- Актуализация принудительно установленных значений
- Актуализация специальных меркеров (SM0, SM5 - SM29 не актуализируются)
- Диагностика во время исполнения
- Таймеры с разрешающей способностью 10 мс и 100 мс не будут надлежащим образом накапливать время для циклов длительностью больше 25 секунд
- Операция STOP в программе обработки прерываний

Указание

Если Вы ожидаете, что время цикла превысит 300 мс, или значительного повышения активности прерываний, так что главный цикл прерывается дольше, чем на 300 мс, то Вам следует перезапускать время контроля с помощью операции WDR.

CPU в течение 1,4 секунды переходит в режим работы STOP, если Вы переводите переключатель CPU в положение STOP.

Примеры операций STOP, END и WDR

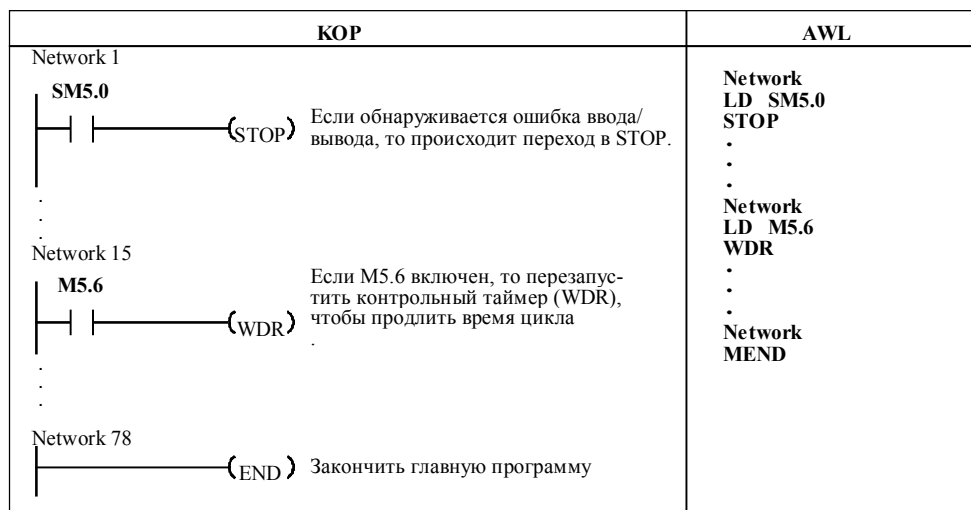
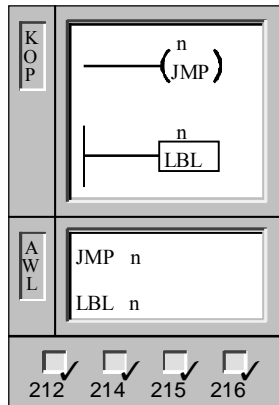


Рис. 9-17. Примеры операций STOP, END и WDR в форме KOP и AWL

Переход на метку, определение метки перехода



Операция **Переход на метку** выполняет ответвление программы к заданной метке перехода (n).

Операция **Определение метки перехода** указывает целевой пункт (n), в который нужно перейти.

Операнды:n: 0 - 255

Операция перехода и относящаяся к ней метка перехода должны обе находиться либо в главной программе, либо в одной подпрограмме, либо в одной программе обработки прерываний. Вы не можете перейти из главной программы на метку, расположенную в подпрограмме или в программе обработки прерываний. Вы также не можете из подпрограммы или программы обработки прерываний перейти на метку, расположенную вне соответствующей подпрограммы или программы обработки прерываний.

На рис. 9–18 показаны примеры операций “Переход на метку перехода” и “Определение метки перехода”.

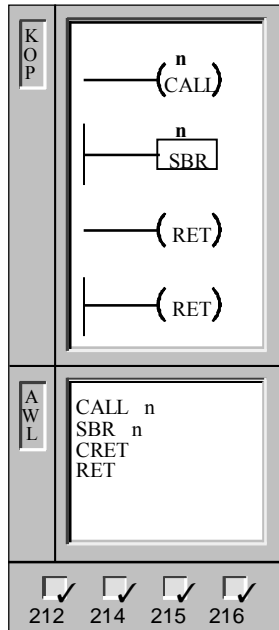
Если переход выполняется, то в вершине стека всегда находится ”1”. Поэтому Вы можете в сегменте, следующем за блоком LBL, подключить выходы и блоки непосредственно к левой шине тока. В AWL операция загрузки, следующая за операцией LBL, может опускаться.

Пример операции перехода на метку перехода

KOP		AWL
<p>Network 14</p> <p>·</p> <p>·</p> <p>Network 33</p>	<p>Если реланентные данные не потеряны, то перейти на LBL 4.</p> <p>Можно использовать JMP и LBL в главной программе, в подпрограммах или в программах обработки прерываний. Операция перехода и относящаяся к ней метка перехода должны всегда находиться в в одной и той же части кода (обе в главной программе, обе в подпрограмме или обе в программе обработки прерываний).</p>	<p>Network LDN SM0.2 JMP 4 · · Network LBL 4</p>

Рис. 9-18. Примеры операций “Переход на метку перехода” и “Определение метки перехода” в форме KOP и AWL

Вызов, начало и окончание подпрограммы



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

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

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

Любая подпрограмма должна заканчиваться операцией **Абсолютное окончание подпрограммы**.

Операнды:n: от 0 до 63

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

Вы можете производить вложение подпрограмм (внутри одной подпрограммы вызывать другую подпрограмму) с глубиной вложенности, равной максимум восьми уровням. Допустима рекурсия (подпрограмма вызывает саму себя), однако Вам следует применять рекурсию лишь с осторожностью.

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

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

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

На рис. 9–19 показаны примеры операций над подпрограммами.

Пример операций над подпрограммами

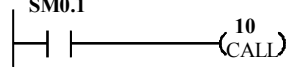
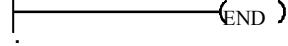
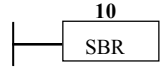
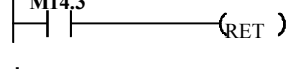
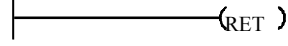
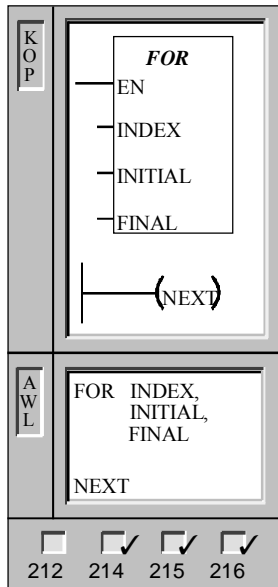
KOP	AWL
Network 1  В первом цикле: вызов SBR10 для инициализации.	Network LD SM0.1 CALL 10 . . .
Network 39  Все подпрограммы нужно располагать после операции END.	Network MEND . . .
Network 50  Начало подпрограммы 10.	Network SBR 10 . . .
Network 65  Подпрограмма 10 может заканчиваться условно ? RET)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

Рис. 9-19. Примеры операций над подпрограммами в форме KOP и AWL

FOR и NEXT



Операция **Программный цикл с FOR** выполняет операции между FOR и NEXT. Вы должны задать текущее значение счетчика программного цикла (INDEX), начальное значение (INITIAL) и конечное значение (FINAL).

Операция **Конец программного цикла с FOR (NEXT)** отмечает конец программного цикла с FOR и устанавливает вершину стека в "1".

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

Например, если значение INITIAL равно 1 и значение FINAL равно 10, то операции между FOR и NEXT выполняются десять раз, причем счетчик INDEX каждый раз увеличивается на "1": 1, 2, 3, ... 10.

Если начальное значение больше, чем конечное значение, то цикл не выполняется. После каждого выполнения операций между FOR и NEXT значение INDEX увеличивается на "1", и результат сравнивается с конечным значением. Если INDEX больше, чем конечное значение, то цикл заканчивается.

С помощью операций FOR и NEXT Вы можете управлять программными циклами, которые повторяются определенное количество раз. Каждая операция FOR требует операции NEXT. Вы можете производить взаимное вложение программных циклов с FOR и NEXT (внутри одного программного цикла с FOR и NEXT обрабатывать другой программный цикл с FOR и NEXT) с глубиной вложенности, равной максимум восьми уровням.

На рис. 9–20 показаны примеры операций FOR и NEXT.

Примеры операций FOR и NEXT в форме KOP и AWL

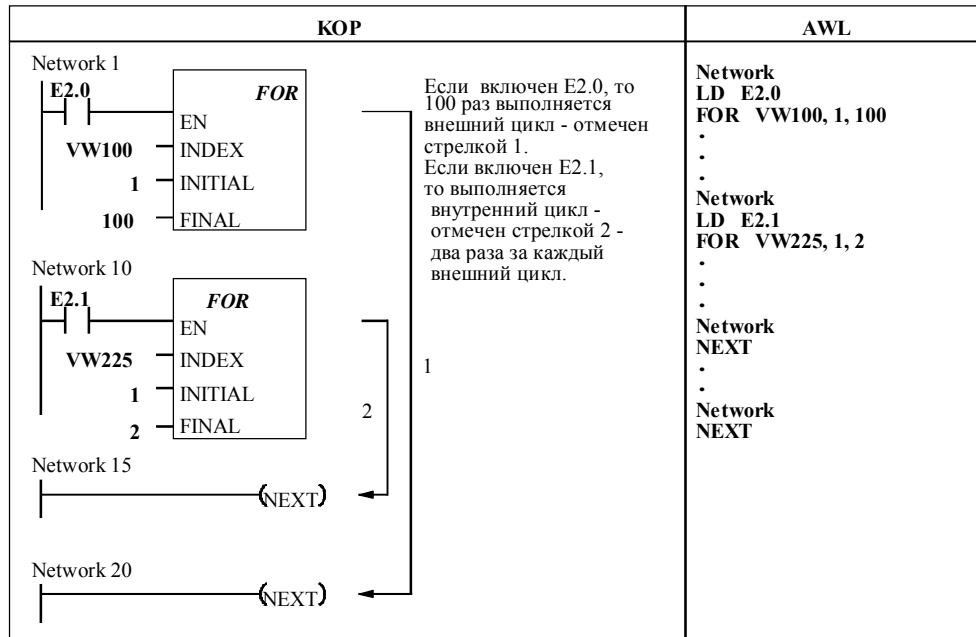
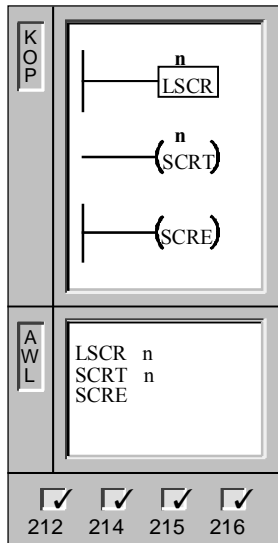


Рис. 9-20. Примеры операций FOR и NEXT в форме KOP и AWL

Операции для реле шагового управления



Операция **Загрузка реле шагового управления** отмечает начало сегмента SCR. Если $n = 1$, то разблокируется поток сигнала к сегменту SCR. Сегмент SCR должен заканчиваться операцией SCRE.

Операция **Фронт реле шагового управления** отмечает SCR-бит, который должен разблокироваться (следующий S-бит, который должен быть установлен). Если поток сигнала поступает к катушке, то включается указанный S-бит и выключается S-бит операции LSCR (который разблокировал данный сегмент SCR).

Операция **Конец реле шагового управления** отмечает конец сегмента SCR.

Операнды: n: S

Описание операций реле шагового управления

В KOP и AWL реле шагового управления (SCR) используются для организации операций или шагов, выполняемых установкой, в эквивалентные программные сегменты. Реле шагового управления подразделяют управляющую программу на логические сегменты.

Операция LSCR загружает значение S-бита, заданного в операции, в стек реле шагового управления и в логический стек. Сегмент SCR активизируется или деактивируется результирующим значением стека SCR. Вершина логического стека загружается в заданный S-бит, так что блоки и катушки могут подключаться к левой шине тока непосредственно без промежуточного контакта. На рис. 9-21 показаны S-стек и логический стек и последствия операции LSCR.

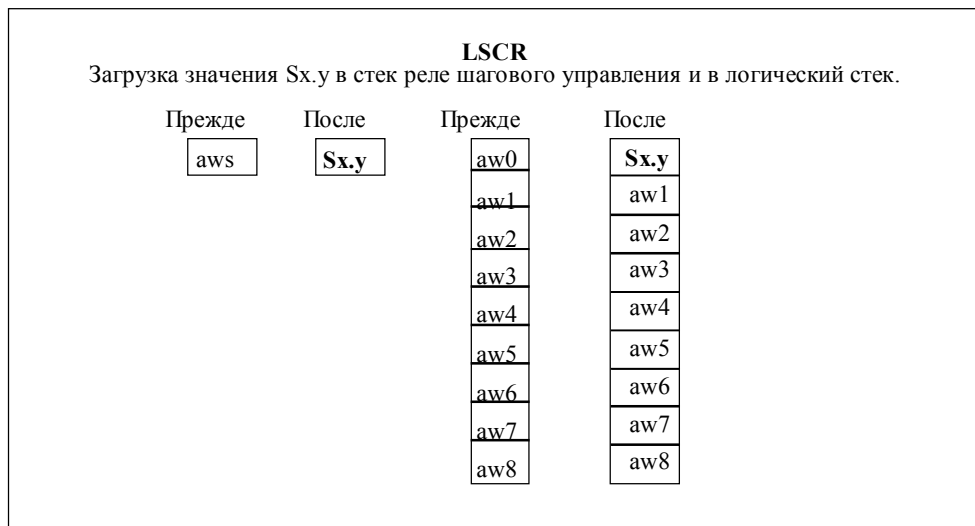


Рис. 9-21. Влияние операции LSCR на логический стек

При операциях с реле шагового управления необходимо учитывать следующее:

- Все операции между операцией LSCR и операцией SCRE образуют сегмент SCR и в отношении выполнения зависят от значения S-стека. Логика программы между операцией SCRE и следующей операцией LSCR не зависит от значения S-стека.
- Операция SCRT устанавливает S-бит, разблокирующий следующее реле шагового управления, и сбрасывает S-бит, который был загружен для того, чтобы разблокировать данный раздел сегмента SCR.

Указание

Вы можете использовать реле шагового управления в главной программе, однако их нельзя вставлять в подпрограммы или программы обработки прерываний.

С помощью операций перехода можно совершать переходы внутри сегментов SCR, а также перескакивать через сегменты SCR. Однако нельзя совершать переходы снаружи внутрь сегмента SCR или изнутри сегмента SCR наружу.

Пример для реле шагового управления

На рис. 9–22 показан пример, поясняющий принцип работы реле шагового управления

- В этом примере с помощью специального меркера SM0.1 (меркер первого цикла) устанавливается S0.1. В первом цикле S0.1 находится на активном шаге 1.
- После 2-секундной задержки T37 вызывает переключение на следующий шаг 2. Это переключение деактивирует сегмент SCR для шага 1 (S0.1) и активизирует сегмент SCR для шага 2 (S0.2).

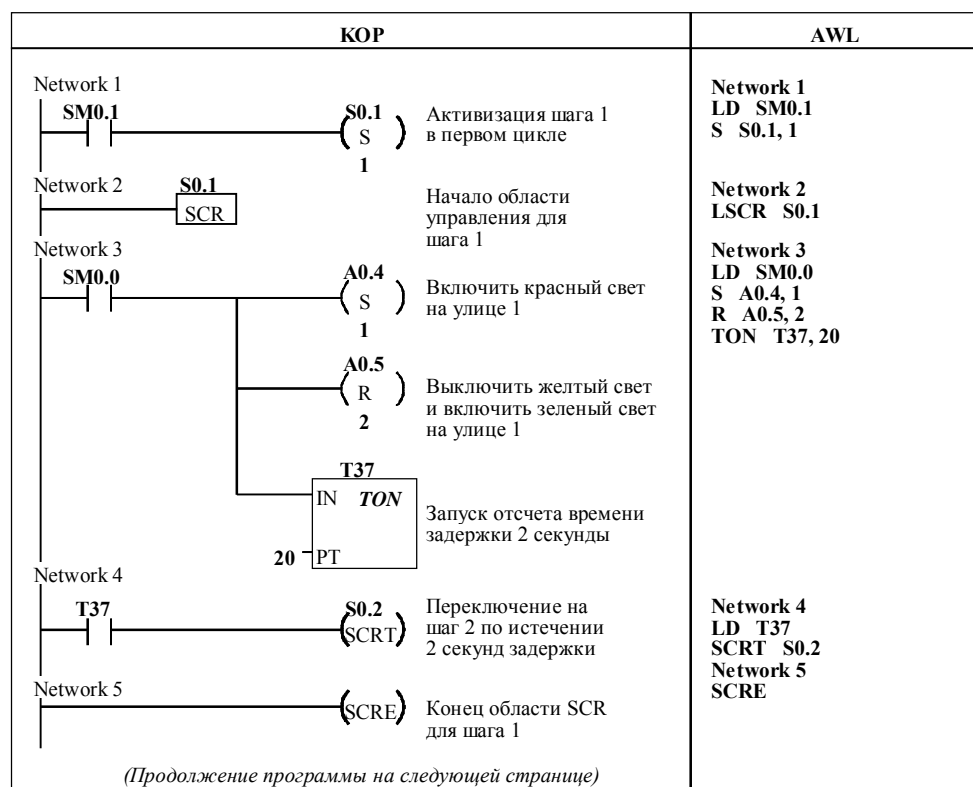


Рис. 9-22. Пример для реле шагового управления (SCR)

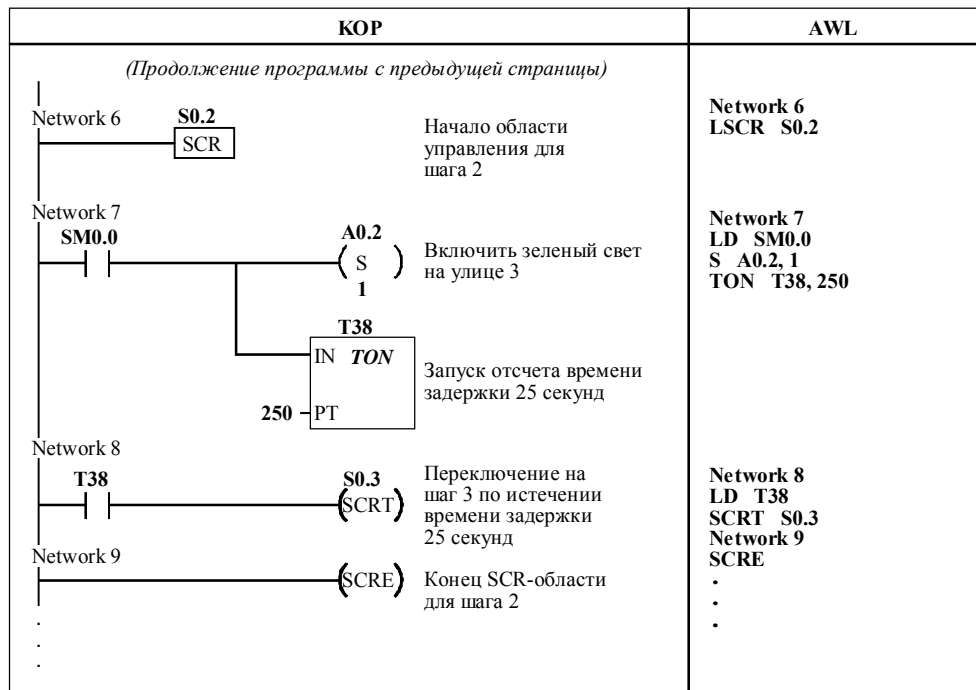


Рис. 9-22. Пример для реле шагового управления (SCR), продолжение

Разделение потоков управления

Во многих приложениях необходимо разделять единый поток последовательных шагов на два или более отдельных потоков. Если поток управления расчленяется на несколько потоков, то все выходящие потоки должны активизироваться одновременно. Этот процесс показан на рисунке 9-23.

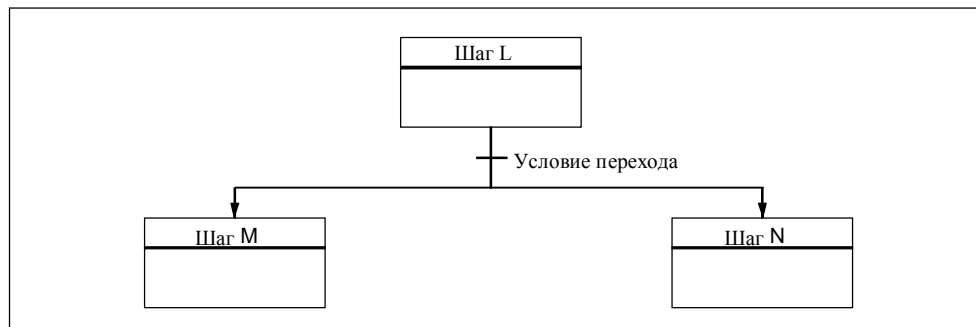


Рис. 9-23. Разделение потока управления

Разделение потоков управления можно реализовать в программе SCR посредством того, что несколько операций SCRT активизируются одним и тем же условием перехода (см. рис. 9–24).

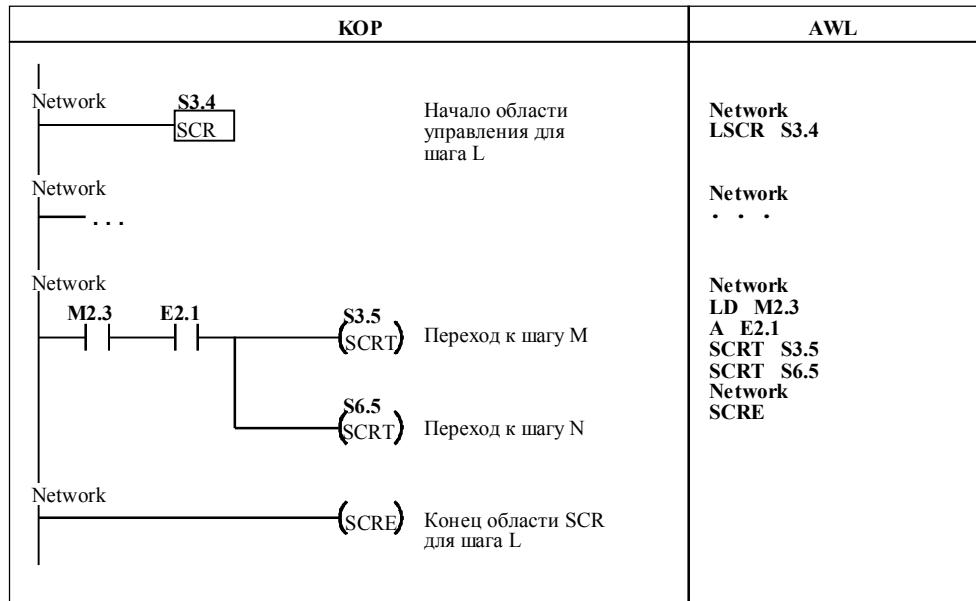


Рис. 9-24. Пример разделения потока управления

Сведение потоков управления

Подобная ситуация возникает, когда два или более потоков последовательных состояний должны соединиться в один поток. Если два или более потоков управления сливаются в один поток, то это называют сведением. При сведении потоков управления все входящие потоки должны быть завершены до того, как начнет выполняться следующий шаг. На рис. 9–25 показано сведение двух потоков управления.

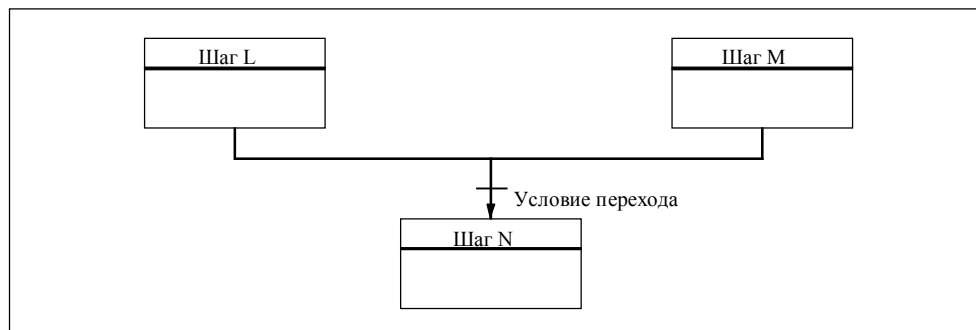


Рис. 9-25. Сведение потоков управления

Сведение потоков управления можно реализовать в программе SCR посредством того, что происходит переключение с шага L на шаг L' и с шага M на шаг M'. Когда оба SCR-бита, представляющие L' и M', истинны, то может быть активизирован шаг N, как показано ниже.

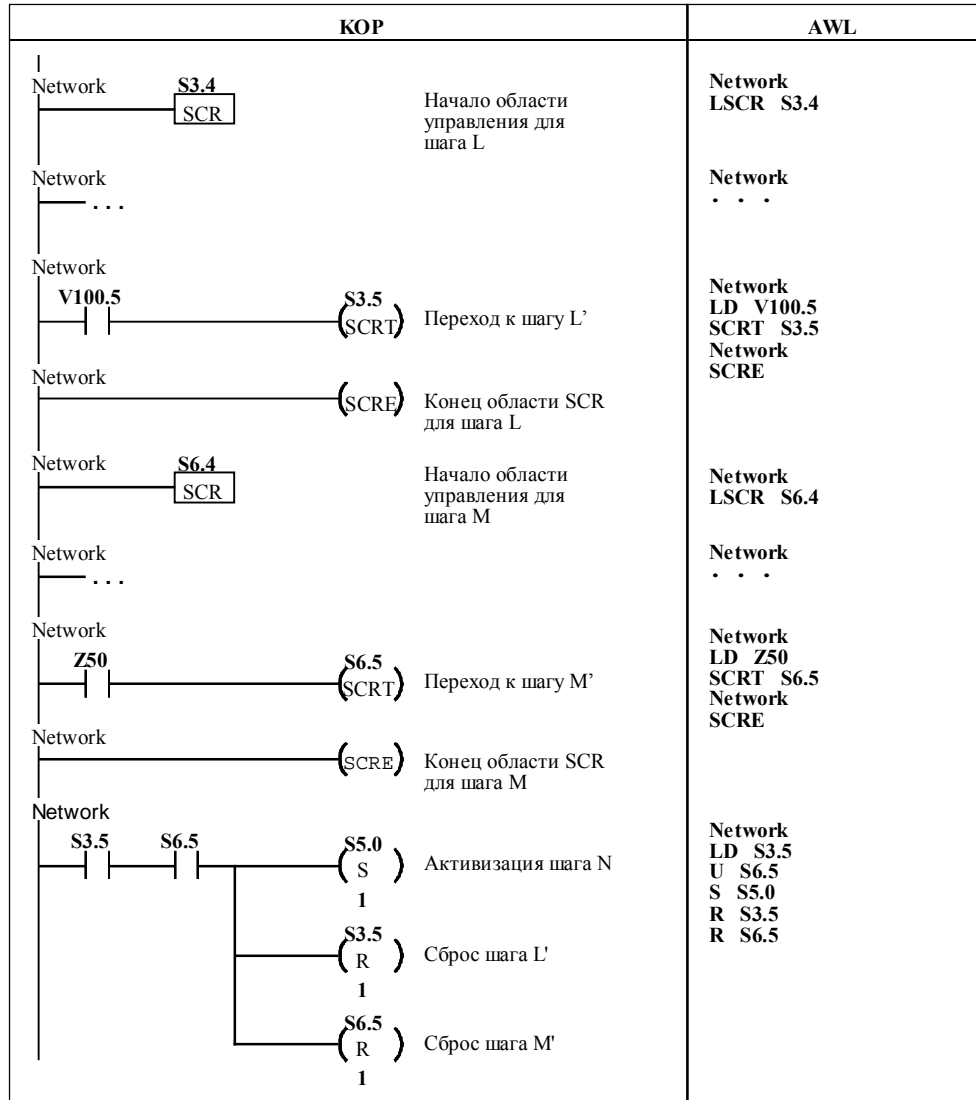


Рис. 9-26. Пример сведения потоков управления

Набор операций

В других ситуациях поток управления может быть направлен в один из нескольких возможных потоков управления в зависимости от того, какое условие перехода на следующий шаг первым станет истинным. Этот случай представлен на рисунке 9–27.

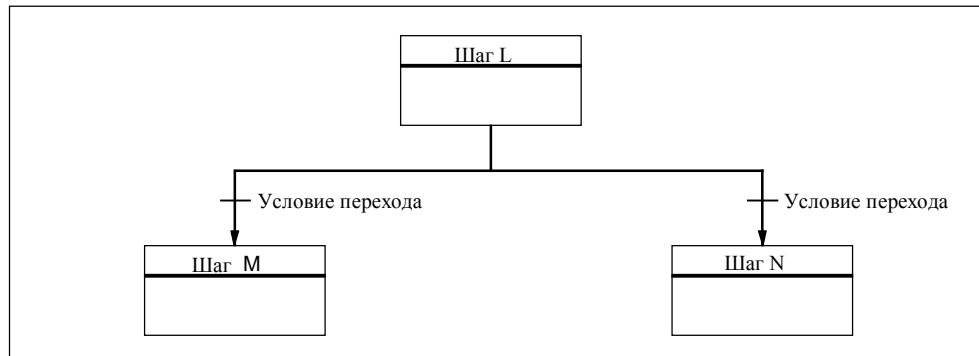


Рис. 9-27. Перенаправление потока управления в зависимости от условия перехода

Эквивалентная программа SCR показана на рисунке 9–28.

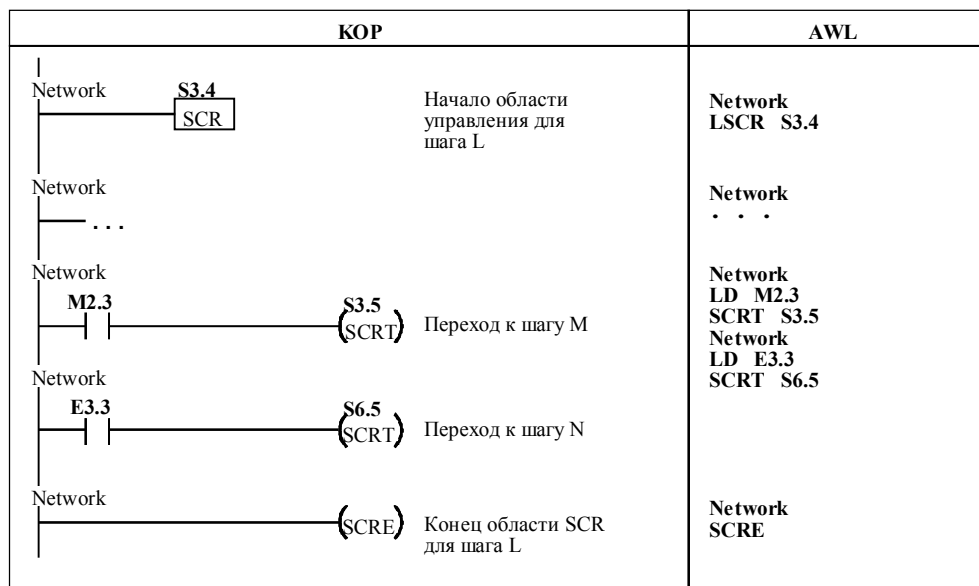


Рис. 9-28. Пример условий перехода на следующий шаг

Нуль-операция

K O P	\xrightarrow{N} (NOP)
A W L	NOP N
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
212	214
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
215	216

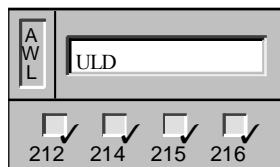
Нуль-операция не влияет на обработку программы пользователя. Операнд N является целым числом в диапазоне от 0 до 255.

операнды: N: от 0 до 255

Нуль-операция должна располагаться внутри главной программы, в подпрограмме или в программе обработки прерываний.

9.9 Стековые операции

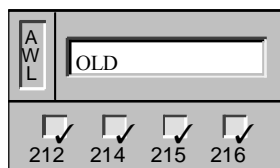
Логическое сопряжение через И первого и второго уровней стека



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

Операнды: нет

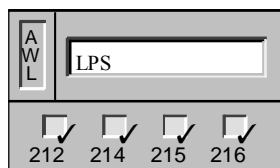
Логическое сопряжение через ИЛИ первого и второго уровней стека



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

Операнды: нет

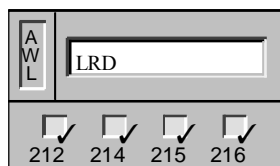
Дублирование вершины стека



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

Операнды: нет

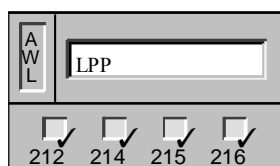
Копирование второго значения стека



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

Операнды: нет

Выталкивание самого верхнего значения стека



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

Операнды: нет

Стековые операции

На рис. 9–29 показаны операции “Логическое сопряжение через И первого и второго уровней стека” и “Логическое сопряжение через ИЛИ первого и второго уровней стека”.

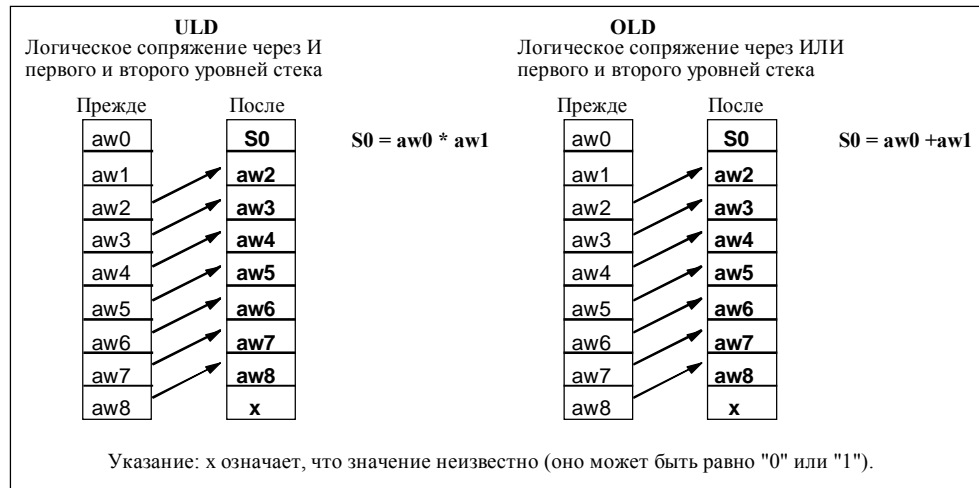


Рис. 9-29. Стековые операции ULD и OLD

На рис. 9–30 показаны операции “Дублирование вершины стека”, “Копирование второго значения стека” и “Выталкивание самого верхнего значения стека”.

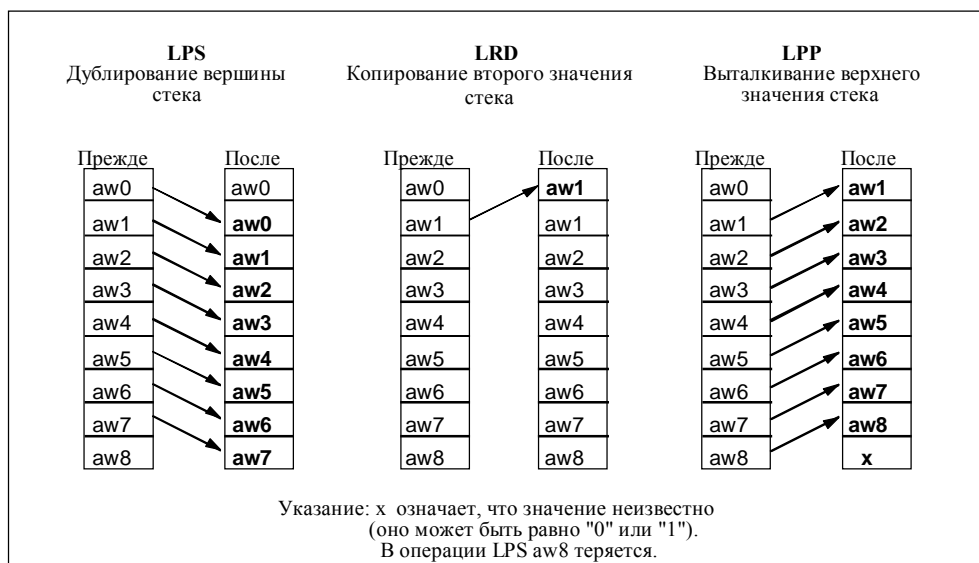


Рис. 9-30 Стековые операции LPS, LRD и LPP

Примеры стековых операций

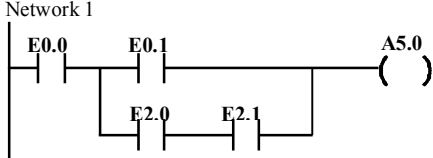
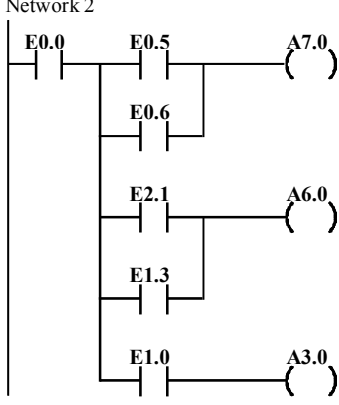
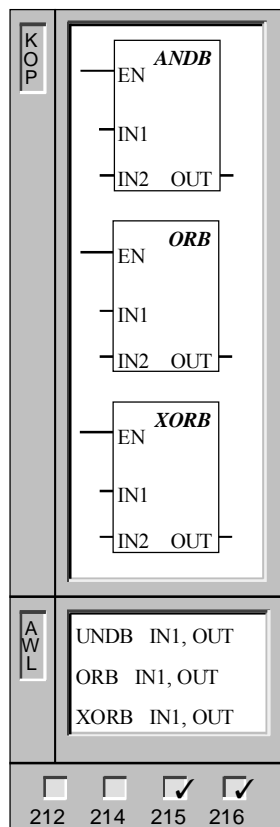
KOP	AWL
<p>Network 1</p>  <p>Network 2</p> 	<pre> NETWORK LD E0.0 LD E0.1 LD E2.0 U E2.1 OLD ULD = A5.0 NETWORK LD E0.0 LPS LD E0.5 O E0.6 ULD = A7.0 LRD LD E2.1 O E1.3 ULD = A6.0 LPP U E1.0 = A3.0 </pre>

Рис. 9-31. Примеры стековых операций в форме KOP и AWL

9.10 Логические операции

Логическое сопряжение байтов через И, ИЛИ и ИСКЛЮЧАЮЩЕЕ ИЛИ



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

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

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

Операнды: IN1, IN2: VB, EB, AB, MB, SMB, SB,
AC, константа, *VD, *AC, SB

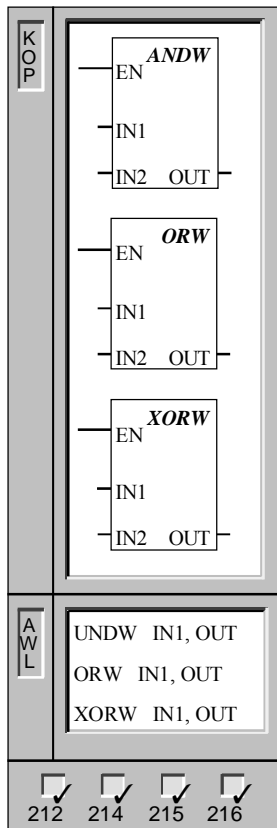
OUT: VB, EB, AB, MB, SMB, SB,
AC, *VD, *AC, SB

Указание: При программировании в KOP Вы можете указать, что IN1 совпадает с OUT. Таким образом, Вы экономите место в памяти.

Эти операции влияют на следующие специальные меркеры:

SM1.0 (нуль)

Логическое сопряжение слов через И, ИЛИ и ИСКЛЮЧАЮЩЕЕ ИЛИ



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

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

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

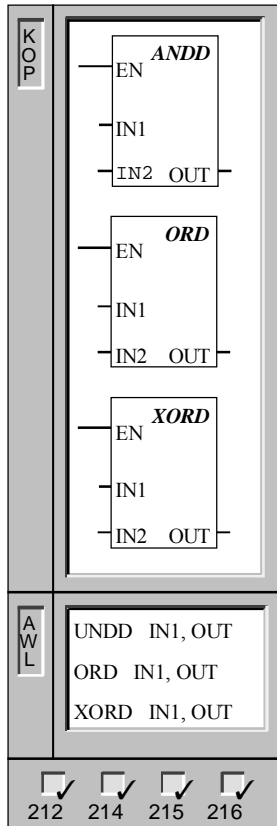
Операнды: IN1, IN2: VW, T, Z, EW, AW, MW, SMW, AC, AEW, константа, *VD, *AC, SW

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

Указание: При программировании в KOP Вы можете указать, что IN1 совпадает с OUT. Таким образом, Вы экономите место в памяти.

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

Логическое сопряжение двойных слов через И, ИЛИ и ИСКЛЮЧАЮЩЕЕ ИЛИ



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

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

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

Операнды: IN1, IN2: VD, ED, AD, MD, SMD, AC, HC,
 константа, *VD, *AC, SD

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

Указание: При программировании в KOP Вы можете указать, что IN1 совпадает с OUT. Таким образом, Вы экономите место в памяти.

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

Примеры логических операций

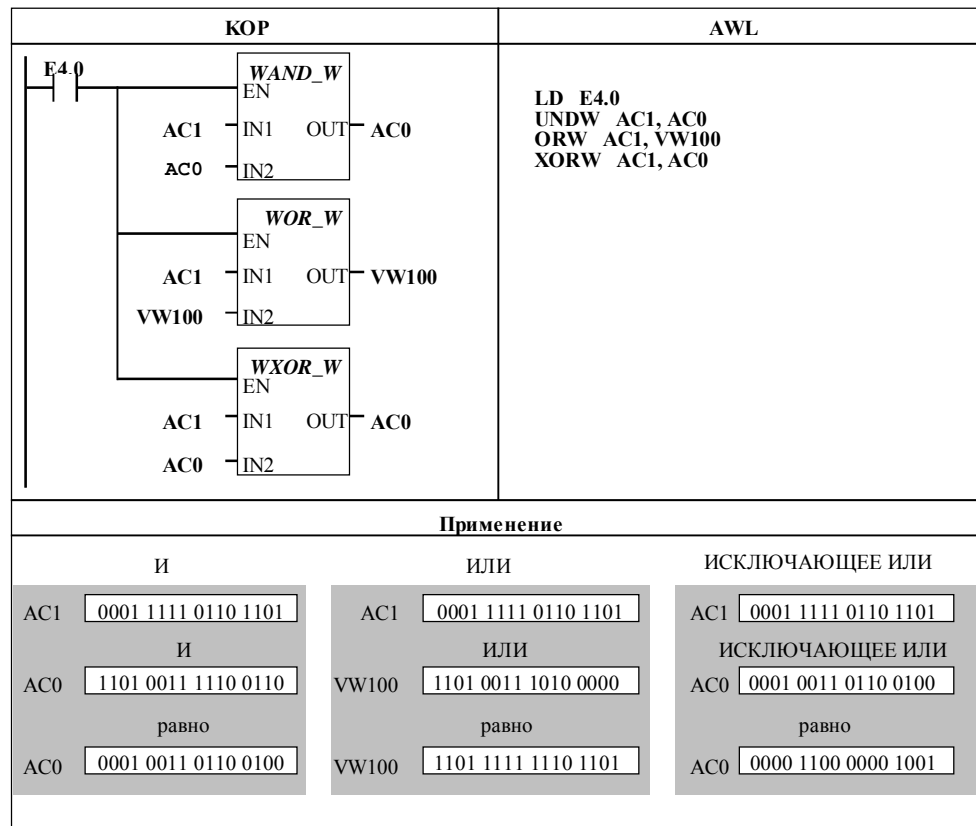
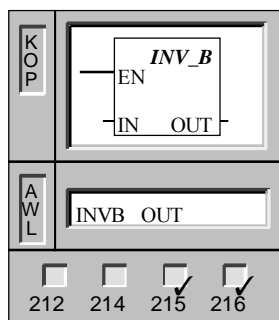


Рис. 9-32. Примеры логических операций

Образование дополнения до единицы для байта



Операция **Образование дополнения до единицы для байта** образует дополнение до единицы для значения входного байта и загружает результат в байт (OUT).

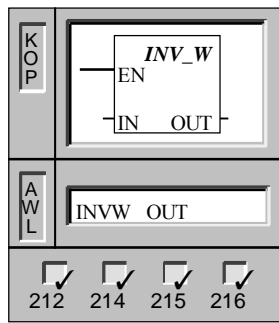
Операнды: IN:VB, EB, AB, MB, SMB, SB, AC, *VD, *AC, SB
 OUT:VB, EB, AB, MB, SMB, SB, AC, *VD, *AC, SB

Указание: При программировании в KOP Вы можете указать, что IN1 совпадает с OUT. Таким образом, Вы экономите место в памяти.

Эти операции влияют на следующие специальные меркеры:

SM1.0 (нуль)

Образование дополнения до единицы для целого числа (16 бит)



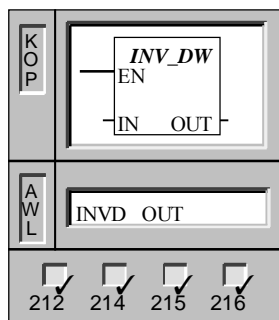
Операция **Образование дополнения до единицы для целого числа (16 бит)** образует дополнение до единицы для значения входного слова и загружает результат в слово (OUT).

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

Указание: При программировании в KOP Вы можете указать, что IN совпадает с OUT. Таким образом Вы экономите место в памяти.

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

Образование дополнения до единицы для целого числа (32 бита)



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

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

Указание: При программировании в KOP Вы можете указать, что IN совпадает с OUT. Таким образом, Вы экономите место в памяти.

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

Пример операции дополнения до единицы

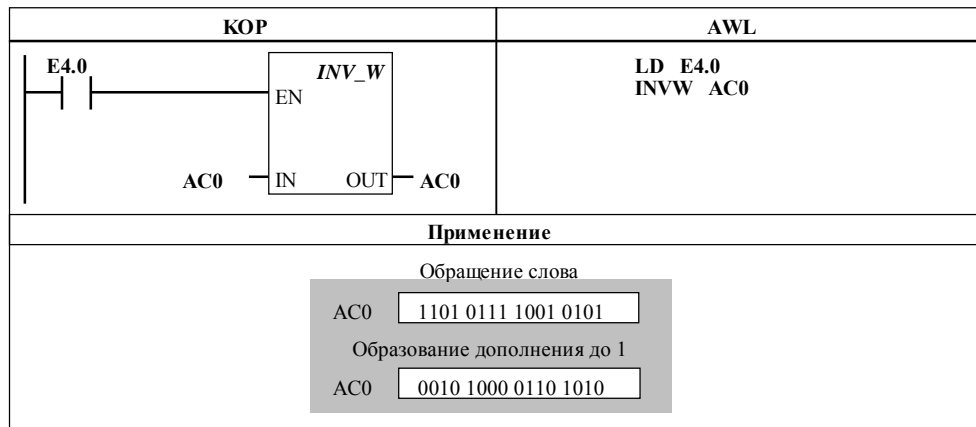
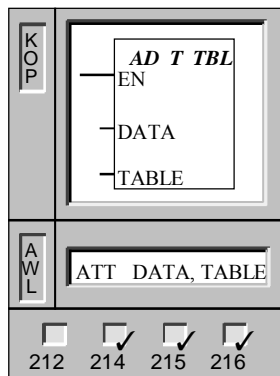


Рис. 9-33. Пример операции образования дополнения до единицы в KOP и AWL

9.11 Табличные операции и операции поиска

Запись значения в таблицу



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

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

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

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

Эта операция влияет на следующие специальные меркеры:

SM1.4 устанавливается в "1", если Вы пытаетесь записать в таблицу слишком много значений.

Пример операции "Запись значения в таблицу"

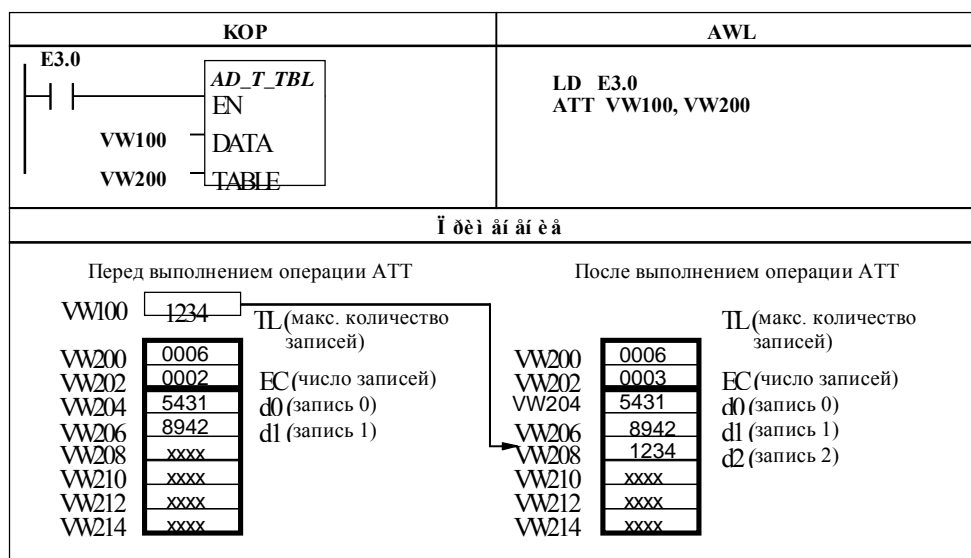
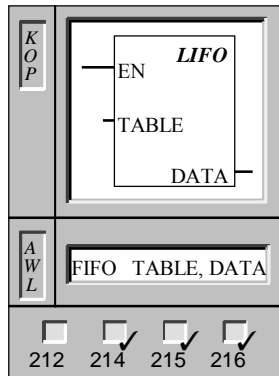


Рис. 9-34. Пример операции "Запись значения в таблицу"

LIFO



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

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

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

Эта операция влияет на следующие специальные меркеры:

SM1.5 (устанавливается в "1", если Вы пытаетесь стереть запись в пустой таблице).

Пример операции LIFO

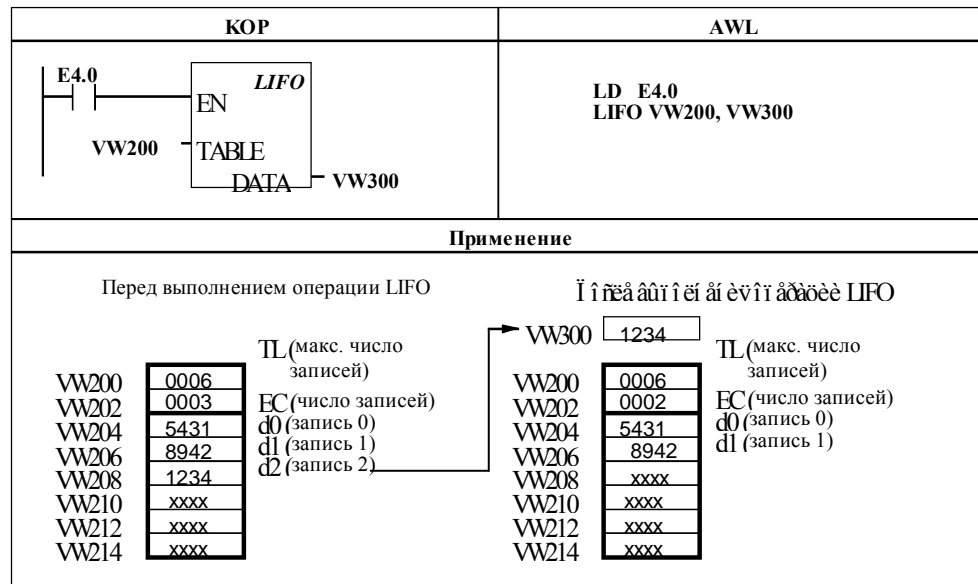
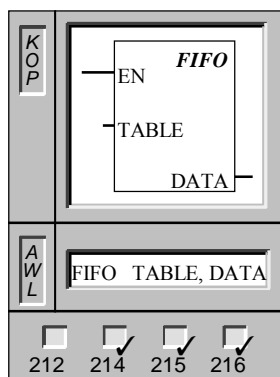


Рис. 9-35. Пример операции LIFO в форме KOP и AWL

FIFO



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

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

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

Эта операция влияет на следующие специальные меркеры:

SM1.5 устанавливается в "1", если Вы пытаетесь стереть запись в пустой таблице.

Пример операции FIFO

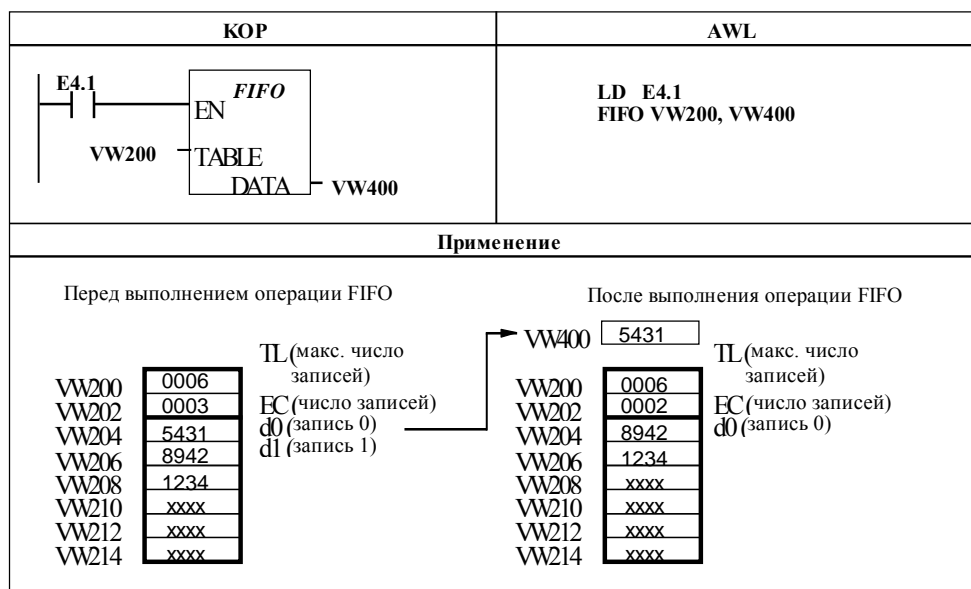


Рис. 9-36. Пример операции FIFO в форме KOP и AWL

Поиск значения в таблице

KOP	TBL_FIND — EN — SRC — PATRN — INDX — CMD
AWL	FND= SRC, PATRN, FND<> SRC, PATRN, INDX FND< SRC, PATRN, FND> SRC, PATRN,
	<input type="checkbox"/> 212 <input checked="" type="checkbox"/> 214 <input checked="" type="checkbox"/> 215 <input checked="" type="checkbox"/> 216

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

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

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

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

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

CMD: 1 (=) 2 (<>) 3 (<) 4 (>)

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

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

задающие максимальную длину таблицы и фактическое количество записей в таблице.

Указание

Если Вы используете операции поиска в таблицах, составленных с помощью операций ATT, LIFO и FIFO, то количество записей и записи данных имеют прямое соответствие. В отличие от операций ATT, LIFO и FIFO, где максимальное количество записей задается в слове, операции поиска не используют данное слово. Поэтому операнд SRC операции поиска располагается на один адрес слова (два байта) выше, чем операнд таблицы, соответствующей операции ATT, LIFO или FIFO, как показано на рис. 9-37.

Формат таблицы для ATT, LIFO и FIFO			Формат таблицы для TBL_FIND		
VW200	0006	TL (i àèñ. ÷èñèí çàí èñáé)	VW202	0006	TL (макс. число зап.)
VW202	0006	EC (число записей)	VW204	xxxx	d0 (данные 0)
VW204	xxxx	d0 (данные 0)	VW206	xxxx	d1 (данные 1)
VW206	xxxx	d1 (данные 1)	VW208	xxxx	d2 (данные 2)
VW208	xxxx	d2 (данные 2)	VW210	xxxx	d3 (данные 3)
VW210	xxxx	d3 (данные 3)	VW212	xxxx	d4 (данные 4)
VW212	xxxx	d4 (данные 4)	VW214	xxxx	d5 (данные 5)
VW214	xxxx	d5 (данные 5)			

Рис. 9-37. Различные форматы таблицы в операциях поиска и операциях ATT, LIFO и FIFO

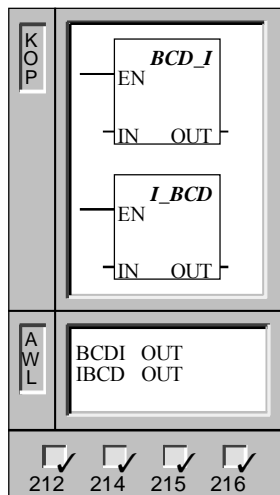
Пример операции поиска

KOP	AWL																					
<table style="border: 1px solid black; padding: 5px;"> <tr> <td style="border-right: 1px solid black; padding: 2px;">E2.1</td> <td style="padding: 2px;">TBL_FIND</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px;">VW202</td> <td style="padding: 2px;">EN</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px;">16#3130</td> <td style="padding: 2px;">SRC</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px;">AC1</td> <td style="padding: 2px;">PATRN</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px;">1</td> <td style="padding: 2px;">INDX</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px;"></td> <td style="padding: 2px;">CMD</td> </tr> </table> <p>Если активизирован E2.1 то в таблице ищется значение, равное 3130 HEX.</p>	E2.1	TBL_FIND	VW202	EN	16#3130	SRC	AC1	PATRN	1	INDX		CMD	<p>ED 12.1 FND= VW202, 16#3130, AC1</p>									
E2.1	TBL_FIND																					
VW202	EN																					
16#3130	SRC																					
AC1	PATRN																					
1	INDX																					
	CMD																					
Применение																						
<p>Это таблица, которую Вы просматриваете. Если таблица была создана с помощью операции ATT, LIFO или FIFO, то VW200 содержит максимально допустимое число записей и не требуется для операций поиска.</p> <table style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <tr> <td style="padding: 2px;">VW202</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">0006</td> <td style="padding: 2px;">EC (число записей)</td> </tr> <tr> <td style="padding: 2px;">VW204</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">3133</td> <td style="padding: 2px;">d0 (запись 0)</td> </tr> <tr> <td style="padding: 2px;">VW206</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">4142</td> <td style="padding: 2px;">d1 (запись 1)</td> </tr> <tr> <td style="padding: 2px;">VW208</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">3130</td> <td style="padding: 2px;">d2 (запись 2)</td> </tr> <tr> <td style="padding: 2px;">VW210</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">3030</td> <td style="padding: 2px;">d3 (запись 3)</td> </tr> <tr> <td style="padding: 2px;">VW212</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">3130</td> <td style="padding: 2px;">d4 (запись 4)</td> </tr> <tr> <td style="padding: 2px;">VW214</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">4541</td> <td style="padding: 2px;">d5 (запись 5)</td> </tr> </table> <p>AC1 <input style="width: 50px;" type="text" value="0"/> AC1 нужно сбросить в "0", чтобы вести поиск с самой верхней записи таблицы.</p> <p style="text-align: center;">Прмотр таблицы</p> <p>AC1 <input style="width: 50px;" type="text" value="2"/> AC1 содержит номер первой записи, соответствующей критерию поиска.</p> <p>AC1 <input style="width: 50px;" type="text" value="3"/> Увеличить INDX на "1" перед просмотром остальных записей таблицы.</p> <p style="text-align: center;">Прмотр таблицы</p> <p>AC1 <input style="width: 50px;" type="text" value="4"/> AC1 содержит номер второй записи, соответствующей критерию поиска.</p> <p>AC1 <input style="width: 50px;" type="text" value="5"/> Увеличить INDX на "1" перед просмотром остальных записей таблицы.</p> <p style="text-align: center;">Прмотр таблицы</p> <p>AC1 <input style="width: 50px;" type="text" value="6"/> AC1 содержит значение, соответствующее числу записей в Tabelle таблице. Вся таблица просмотрена, дальнейшие подходящие записи не найдены.</p> <p>AC1 <input style="width: 50px;" type="text" value="0"/> Для получения возможности нового поиска в таблице нужно сбросить значение INDX в "0".</p>		VW202	0006	EC (число записей)	VW204	3133	d0 (запись 0)	VW206	4142	d1 (запись 1)	VW208	3130	d2 (запись 2)	VW210	3030	d3 (запись 3)	VW212	3130	d4 (запись 4)	VW214	4541	d5 (запись 5)
VW202	0006	EC (число записей)																				
VW204	3133	d0 (запись 0)																				
VW206	4142	d1 (запись 1)																				
VW208	3130	d2 (запись 2)																				
VW210	3030	d3 (запись 3)																				
VW212	3130	d4 (запись 4)																				
VW214	4541	d5 (запись 5)																				

Рис. 9-38. Пример операций поиска в форме KOP и AWL

9.12 Операции преобразования

Преобразование BCD в целое число и преобразование целого числа в BCD



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

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

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

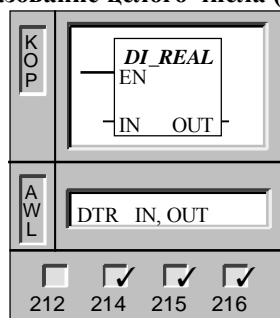
OUT: VW, T, Z, EW, AW, MW, SMW, AC, *VD, *AC, SW

Указание: При программировании в KOP Вы можете указать, что IN совпадает с OUT. Таким образом, Вы экономите место в памяти.

Эти операции влияют на следующие специальные меркеры:

SM1.6 (недействительное BCD-значение)

Преобразование целого числа (32 бита) в действительное число

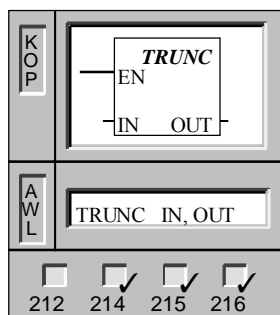


Операция **Преобразование целого числа (32 бита) в действительное число** преобразует целое число (32 бита) со знаком (IN) в действительное число (32 бита) (OUT).

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

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

Преобразование действительного числа в целое число (32 бита)



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

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

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

Эти операции влияют на следующие специальные меркеры:

SM1.1 (переполнение)

Пример преобразования действительного числа

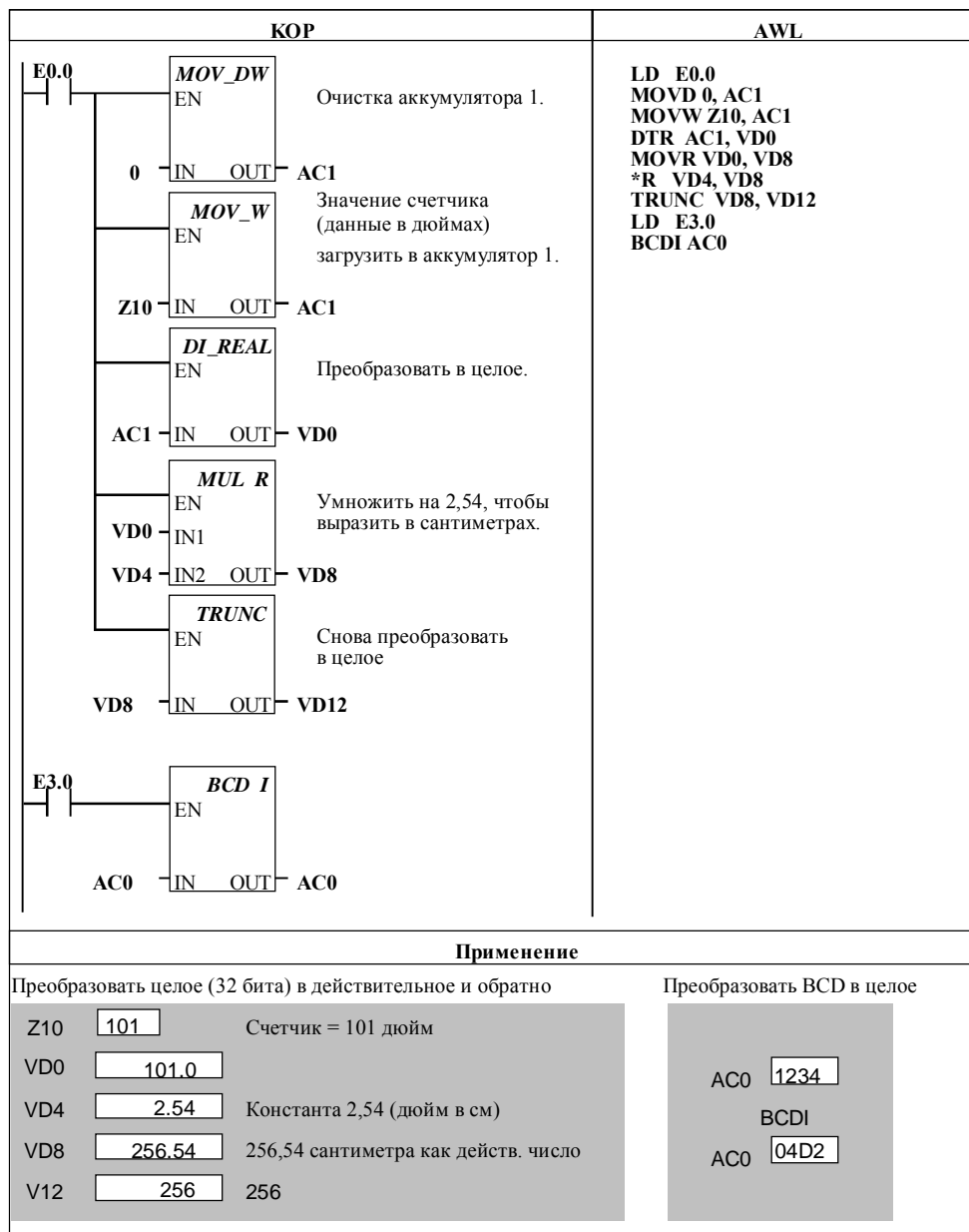
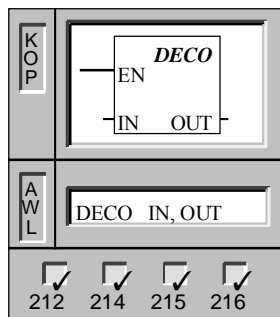


Рис. 9-39. Пример преобразования действительного числа

Преобразование бита в шестнадцатичное число

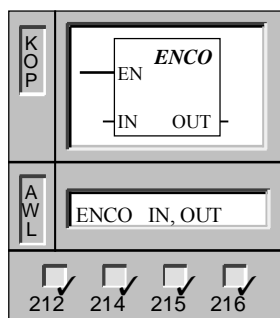


Операция **Преобразование бита в шестнадцатичное число** устанавливает в выходном слове (OUT) бит, номер которого (бит #) соответствует тому, который представлен младшим полубайтом (4 бита) входного байта (IN). Остальные биты выходного слова устанавливаются в “0”.

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

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

Преобразование шестнадцатичного числа в бит

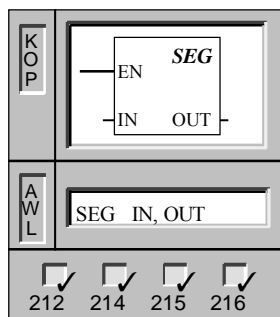


Операция **Преобразование шестнадцатичного числа в бит** записывает номер младшего значащего бита (бит #) входного слова (IN) в младший полубайт (4 бита) выходного байта (OUT).

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

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

Образование битовой комбинации для семисегментного индикатора



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

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

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

На рис. 9–40 показано кодирование, используемое данной операцией для подсветки семисегментного индикатора.

(IN) LSD	Отображ. сегментов	(OUT)					Отображ. сегментов	(OUT)					
		- g	f	e	d c b a			- g	f	e	d c b a		
0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	1	1	0				
2	2	0	1	0	1	1	0	1	1	1	1	1	1
3	3	0	1	0	0	1	1	1	1	1	1	1	1
4	4	0	1	1	0	0	1	1	0				
5	5	0	1	1	0	1	1	0	1				
6	6	0	1	1	1	1	1	0	1				
7	7	0	0	0	0	0	1	1	1				
										8	8	0	1
										9	9	0	1
										A	A	0	1
										B	B	0	1
										C	C	0	0
										D	D	0	1
										E	E	0	1
										F	F	0	1

Рис. 9-40. Кодирование семисегментного индикатора

Примеры преобразования шестнадцатиричных чисел

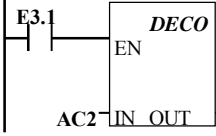
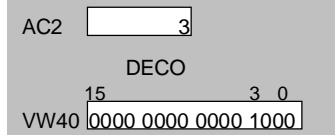
KOP	AWL
 <p>Установка бита, соответствующего коду ошибки в аккумуляторе 2.</p>	<p>LD E3.1 DECO AC2, VW40</p>
Применение	
<p>Аккумулятор 2 содержит код ошибки 3. Операция DECO устанавливает в VW40 бит, соответствующий этому коду.</p>	

Рис. 9-41. Пример установки бита ошибки с помощью операции DECO

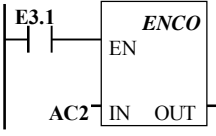
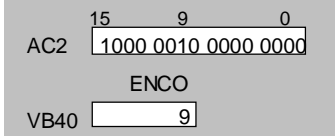
KOP	AWL
 <p>Преобразовать бит ошибки в аккумуляторе 2 в код ошибки в VB40.</p>	<p>LD E3.1 ENCO AC2, VB40</p>
Применение	
<p>Аккумулятор 2 содержит бит ошибки. Операция ENCO преобразует младший значащий бит в код ошибки, который записывается в VB40.</p>	

Рис. 9-42. Пример преобразования бита ошибки в код ошибки с помощью ENCO

Пример образования битовой комбинации для семисегментного индикатора

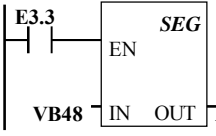
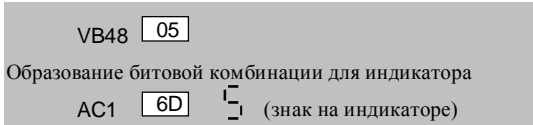
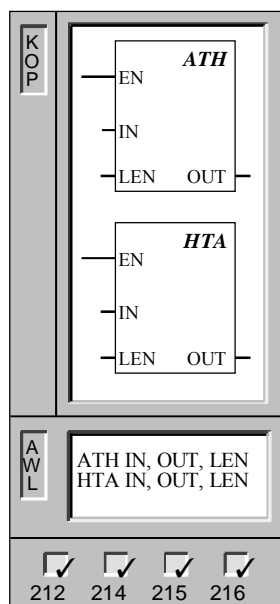
KOP	AWL
	<p>LD E3.3 SEG VB48, AC1</p>
Применение	
	

Рис. 9-43. Пример операции “Образование битовой комбинации для семисегментного индикатора”

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



Операция **Преобразование строки символов ASCII-кода в шестнадцатиричное число** Преобразует строку длиной LEN символов, начиная с символа IN, в шестнадцатиричные цифры, которые начинаются с адреса OUT. Строка символов может быть длиной максимум 255 символов.

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

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

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

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

Эти операции влияют на следующие специальные меркеры: SM1.7 (недопустимый ASCII-символ)

Примеры преобразования ASCII в HEX

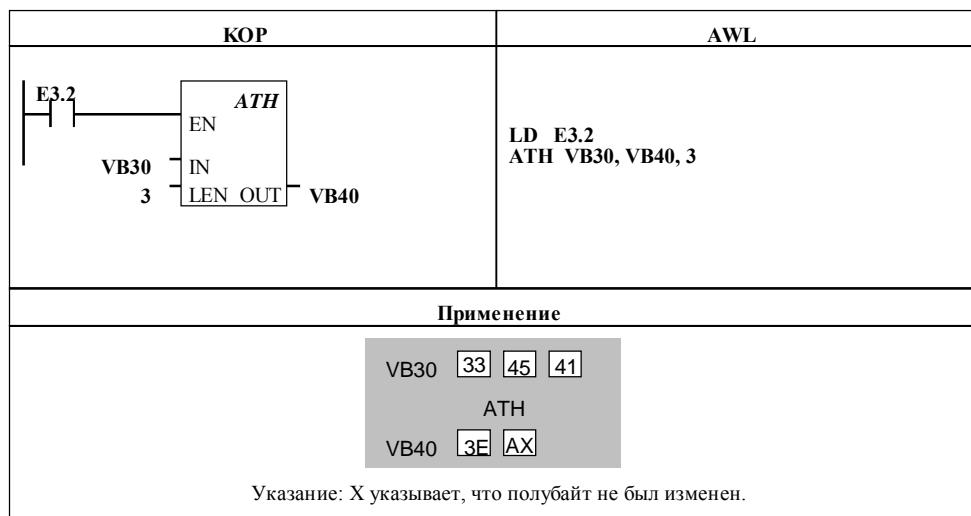
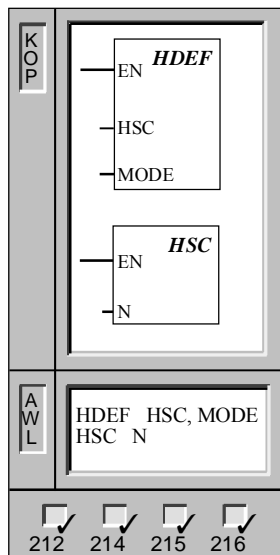


Рис. 9-44. Пример преобразования ASCII-строки в шестнадцатиричные цифры

9.13 Операции с быстрыми счетчиками

Определение режима для быстрых счетчиков, активизация быстрых счетчиков



Операция **Определение режима для быстрых счетчиков** назначает заданному быстрому счетчику (HSC) режим счета (MODE) (см. табл. 9–6).

Операция **Активизация быстрых счетчиков** конфигурирует и управляет режимом работы быстрых счетчиков через сигнальные состояния битов специальных меркеров HSC. Параметр N задает номер быстрого счетчика.

Вы можете использовать для каждого счетчика только один блок HDEF.

Операнды: HSC: от 0 до 2

MODE: 0 (HSC0)
от 0 до 11 (HSC1 или 2)

N: от 0 до 2

Описание операций над быстрыми счетчиками

Быстрые счетчики подсчитывают быстрые события, которыми невозможно управлять с частотой циклической обработки программы контроллера.

- HSC0 представляет собой реверсивный счетчик, который поддерживает тактовый вход. Ваша программа управляет направлением счета (вперед или назад) через бит управления направлением. Максимальная частота счета данного счетчика составляет 2 кГц.
- HSC1 и HSC2 представляют собой универсальные счетчики, которые можно конфигурировать согласно одному из двенадцати различных режимов счета. Различные режимы счета приведены в таблице 9–6. Максимальная частота счета счетчиков HSC1 и HSC2 определяется Вашим CPU (см. приложение F).

Каждый счетчик имеет в своем распоряжении особые входы, поддерживающие такие функции, как датчик тактовых импульсов, управление направлением счета, сброс и запуск. Для двухфазных счетчиков оба датчика тактовых импульсов могут работать с максимальной частотой. В случае A/B–счетчиков (в квадратурных режимах) Вы можете выбирать однократную или четырехкратную скорость счета. HSC1 и HSC2 полностью не зависят друг от друга и не влияют на другие быстрые операции. Оба счетчика работают с максимальной частотой, не оказывая друг на друга отрицательного воздействия.

На рис. 9–52 показан пример инициализации HSC1.

Использование быстрых счетчиков

Быстрые счетчики в типовой ситуации используются в качестве привода счетных механизмов, в которых вал, вращающийся с постоянным числом оборотов, снабжен шаговым датчиком угла. Шаговый датчик угла обеспечивает определенное количество счетных значений за оборот, а также один импульс сброса за оборот. Датчики тактовых импульсов и импульс сброса шагового датчика угла подают входные сигналы для быстрых счетчиков. В быстрый счетчик загружается первое из нескольких предварительно установленных значений. Желаемые выходы активизируются в промежутки времени, в котором текущее значение счетчика меньше, чем предварительно установленное значение. Счетчик организуется таким образом, что возникает сигнал прерывания, когда текущее значение счетчика равно предварительно установленному значению или когда счетчик сбрасывается.

Если текущее значение счетчика равно предварительно установленному значению и возникает сигнал прерывания, то загружается новое предварительно установленное значение и устанавливается следующее состояние сигналов для выходов. Если событие прерывания возникает потому, что счетчик сбрасывается, то устанавливаются первое предварительно установленное значение и первые состояния сигналов выходов и повторяется цикл.

Так как прерывания появляются с частотой, намного меньшей, чем частота, с которой работает счетчик, то можно реализовать точное управление быстрыми операциями с относительно низким влиянием на общий цикл контроллера. Так как Вы можете назначать прерывания определенным программам обработки прерываний, то каждая новая предварительная установка может загружаться в отдельной программе обработки прерываний, позволяя легко управлять состоянием и без труда отслеживать программу. Естественно, Вы можете обрабатывать в отдельной программе обработки прерываний также все события прерываний. Подробную информацию по этому вопросу Вы найдете в разделе по операциям обработки прерываний.

Импульсные диаграммы быстрых счетчиков

Приведенные ниже импульсные диаграммы (рисунки 9–45, 9–46, 9–47 и 9–48) показывают, как работает каждый счетчик соответственно своему классу. Работа входов запуска и сброса представлена на отдельной импульсной диаграмме и имеет силу для всех счетчиков, использующих данные входы. На диаграммах для входов сброса и запуска активность входов запрограммирована как “высокая”.

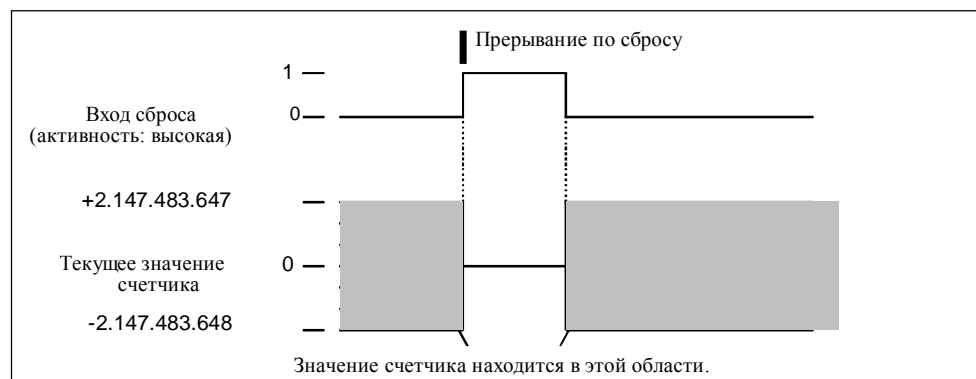


Рис. 9-45. Пример работы счетчика с входом сброса и без входа запуска

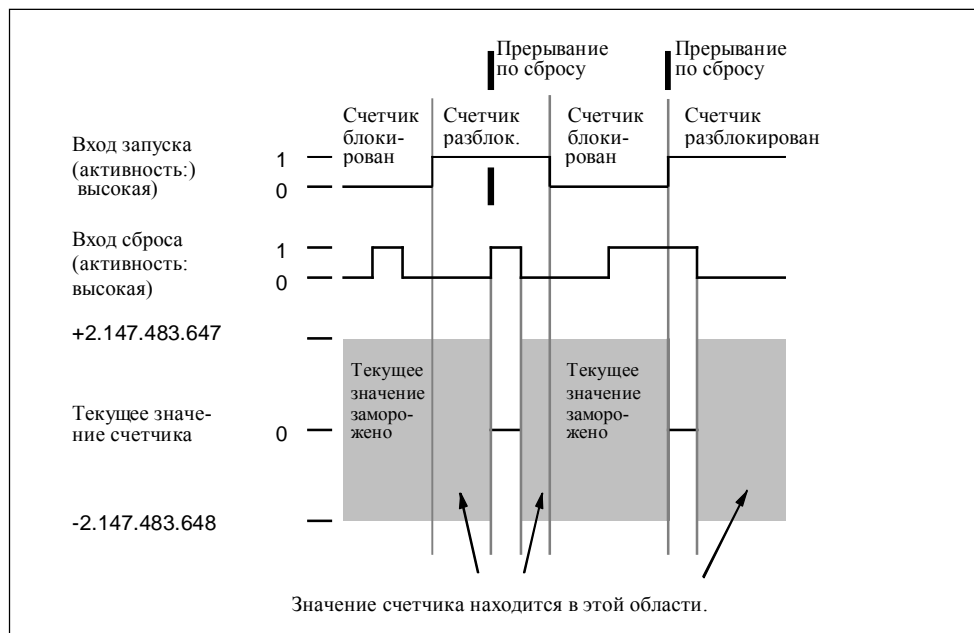


Рис. 9-46. Пример работы счетчика с входами сброса и запуска

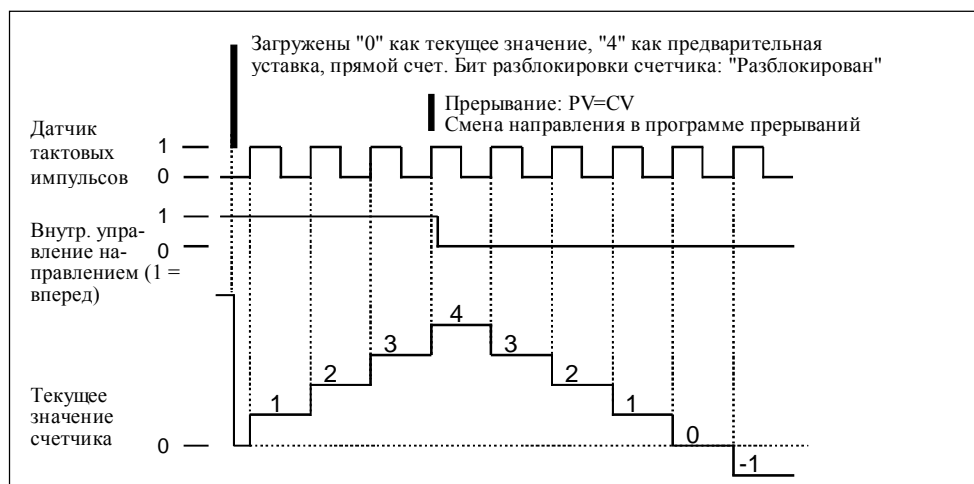


Рис. 9-47. Пример работы HSC0 в счетном режиме 0 и HSC1 или HSC2 в одном из счетных режимов 0, 1 или 2

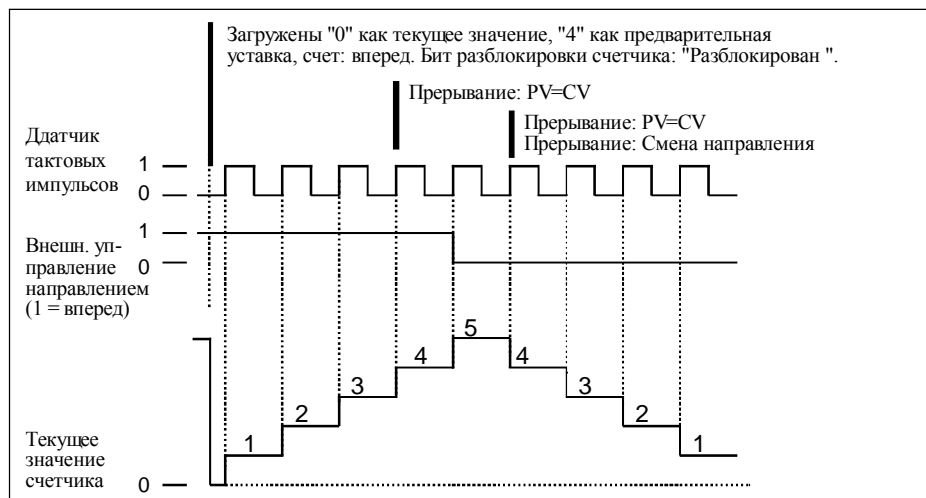


Рис. 9-48. Пример работы HSC1 или HSC2 в одном из режимов счета 3, 4 или 5

Если Вы используете для HSC1 или HSC2 один из режимов счета 6, 7 или 8 и при этом в течение 0,3 микросекунды появляется нарастающий фронт как на входе прямого счета, так и на входе обратного счета, то быстрый счетчик может интерпретировать оба этих события как одновременные. В этом случае текущее значение не изменяется, а также не отображается смена направления счета. Если между появлением нарастающего фронта на входе прямого счета и появлением нарастающего фронта на входе обратного счета прошло более 0,3 микросекунд, то быстрый счетчик воспринимает оба этих события раздельно. Ни в одном из этих двух случаев не возникает ошибка, и счетчик сохраняет правильное счетное значение (см. рисунки 9-50, 9-51 и 9-52).

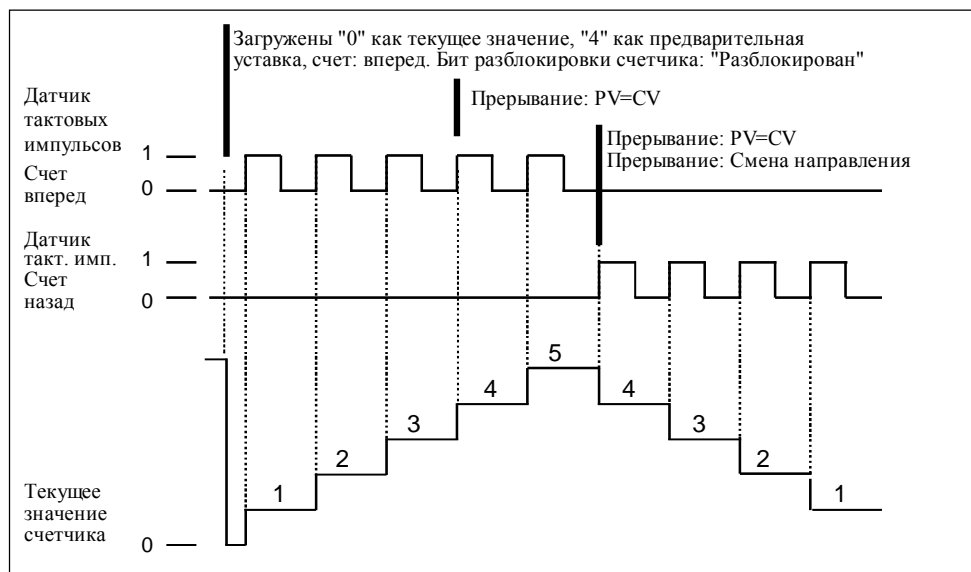


Рис. 9-49. Пример работы HSC1 или HSC2 в одном из режимов счета 6, 7 или 8

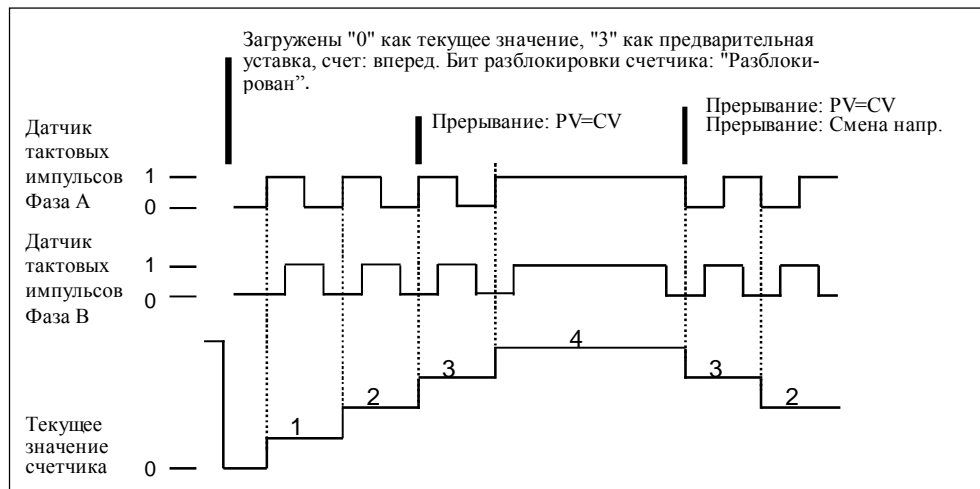


Рис. 9-50. Пример работы HSC1 или HSC2 в одном из режимов счета 9, 10 или 11 (А/В-счетчик, однократная скорость)

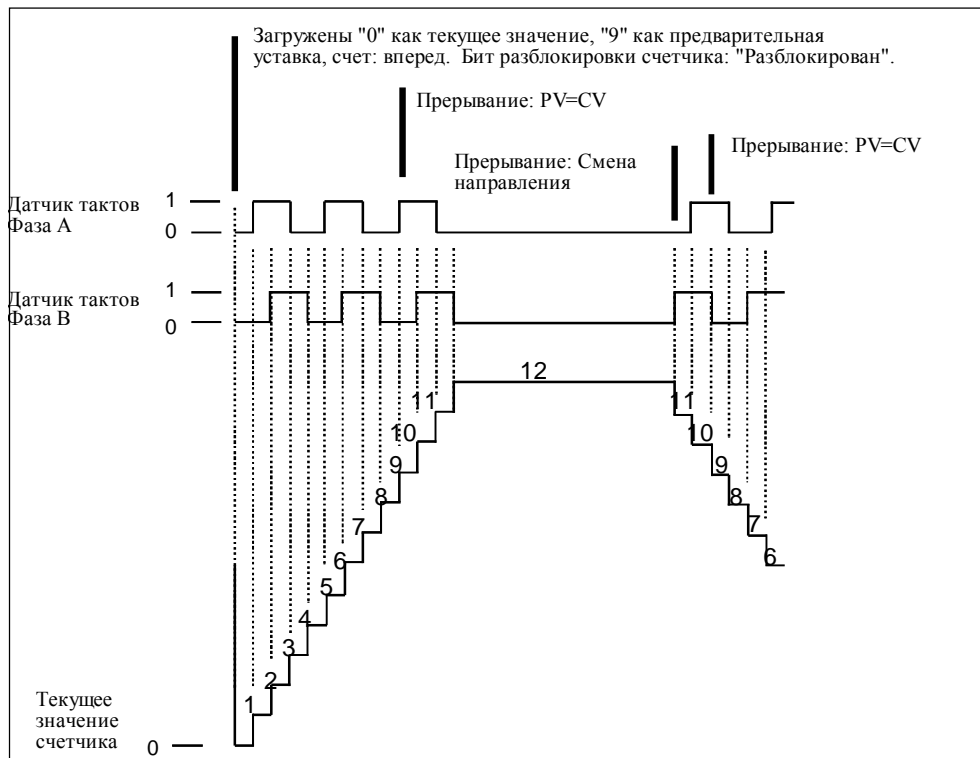


Рис. 9-51. Пример работы HSC1 или HSC2 в одном из режимов счета 9, 10 или 11 (А/В-счетчик, четырехкратная скорость)

Монтаж проводных соединений входов быстрых счетчиков

Таблица 9–5 показывает входы быстрых счетчиков, которые могут использоваться для таких функций, как датчик тактовых импульсов, управление направлением счета, сброс и запуск. Функции входов описаны в таблице 9–6.

Таблица 9–5. Входы быстрых счетчиков

Быстрый счетчик	Используемые входы
HSC0	E0.0
HSC1	E0.6, E0.7, E1.0, E1.1
HSC2	E1.2, E1.3, E1.4, E1.5

Таблица 9–6. Режимы быстрых счетчиков

HSC0					
Режим	Описание	E0.0			
0	Однофазный реверсивный счетчик с внутренним управлением направлением счета SM37.3 = 0, обратный счет SM37.3 = 1, прямой счет	Датчик тактовых импульсов			
HSC1					
Режим	Описание	E0.6	E0.7	E1.0	E1.1
0	Однофазный реверсивный счетчик с внутренним управлением направлением счета SM47.3 = 0, обратный счет SM47.3 = 1, прямой счет	Датчик такт. имп.		Сброс	
1					
2				Запуск	
3	Однофазный реверсивный счетчик с внешним управлением направлением счета E0.7 = 0, обратный счет E0.7 = 1, прямой счет	Датчик такт. имп.	Направление	Сброс	
4					
5				Запуск	
6	Двухфазный счетчик с тактовыми входами для прямого и обратного счета	Датчик такт. имп. (вперед)	Датчик такт. имп. (назад)	Сброс	
7					
8				Запуск	
9	A/B–счетчик Фаза А опережает В на 90 градусов по часовой стрелке	Датчик такт. имп. (фаза А)	Датчик такт. имп. (фаза В)	Сброс	
10	Фаза В опережает А на 90 градусов против часовой стрелки				
11	Запуск				
HSC2					
Режим	Описание	E1.2	E1.3	E1.4	E1.5
0	Однофазный реверсивный счетчик с внутренним управлением направлением счета SM 57.3 = 0, обратный счет SM 57.3 = 1, прямой счет	Датчик такт. имп.		Сброс	
1					
2				Запуск	
3	Однофазный реверсивный счетчик с внешним управлением направлением счета E1.3 = 0, обратный счет E1.3 = 1, прямой счет	Датчик такт. имп.	Направление счета	Сброс	
4					
5				Запуск	
6	Двухфазный счетчик с тактовыми входами для прямого и обратного счета	Датчик такт. имп. (вперед)	Датчик такт. имп. (назад)	Сброс	
7					
8				Запуск	
9	A/B–счетчик	Датчик тактовых импульсов (фаза А)	Датчик тактовых импульсов (фаза В)	Сброс	
10	Фаза А опережает В на 90 градусов по часовой стрелке				
11	Фаза В опережает А на 90 градусов против часовой стрелки			Запуск	

Описание различных быстрых счетчиков (HSC0, HSC1, HSC2)

Все счетчики (HSC0, HSC1 и HSC2) в одном и том же режиме работают одинаково. Для HSC1 и HSC2 имеются четыре основных режима счета □ см. табл. 9–6). Каждый счетчик можно использовать следующим образом: без входов сброса и запуска, со входом сброса, но без входа запуска, либо со входами сброса и запуска.

Если Вы активизируете вход сброса, то он сбрасывает текущее значение. Текущее значение остается сброшенным до тех пор, пока Вы не деактивизируете вход сброса. Если Вы активизируете вход запуска, то счетчик начинает считать. Если вход запуска деактивизируется, то текущее значение счетчика сохраняется постоянным, а тактовые события игнорируются. Если вход сброса активизируется в то время, когда вход запуска неактивен, то сброс игнорируется, а текущее значение не изменяется. Вход запуска остается деактивизированным. Если вход запуска активизируется в то время, когда вход сброса активен, то текущее значение сбрасывается.

Вы должны выбрать режим счета до того, как сможете использовать быстрый счетчик. Для этого используется операция HDEF (определение режима для быстрого счетчика). HDEF назначает режим счета быстрому счетчику (HSC0, HSC1 или HSC2). Для каждого быстрого счетчика можно выполнить только одну операцию HDEF. Для этого определите быстрый счетчик, вызвав с помощью меркера первого цикла SM0.1 (этот бит включается на время первого цикла, а потом выключается) подпрограмму, содержащую операцию HDEF.

Выбор активного состояния и однократной или четырехкратной скорости

HSC1 и HSC2 имеют в своем распоряжении три управляющих бита, с помощью которых можно конфигурировать активное состояние входов сброса и запуска и выбирать однократную или четырехкратную скорость счета (только в A/B–счетчиках). Эти биты находятся в управляющем байте соответствующего счетчика и используются только тогда, когда выполняется операция HDEF. Эти биты описываются в таблице 9–7.

Прежде чем выполнять операцию HDEF, нужно сначала установить управляющие биты в желаемое состояние. В противном случае счетчик принимает для выбранного счетного режима конфигурацию по умолчанию. Установленная по умолчанию активность входов сброса и запуска для HSC1 и HSC2 имеет значение “высокая”. В A/B–счетчиках по умолчанию установлена четырехкратная скорость счета (четырекратная частота датчика тактовых импульсов). Если операция HDEF выполнена, то Вы больше не сможете изменить настройку счетчика, если Вы сначала не переводите CPU в режим STOP.

Таблица 9–7. Управление уровнем активности входов сброса и запуска; биты выбора однократной или четырехкратной скорости счета HSC1 и HSC2

HSC1	HSC2	Описание (только при выполнении HDEF)
SM47.0	SM57.0	Бит управления активностью входа сброса: 0 - высокая, 1 - низкая.
SM47.1	SM57.1	Бит управления активностью входа запуска: 0 - высокая, 1 - низкая.
SM47.2	SM57.2	Скорость счета A/B–счетчиков: 0 - четырехкратная скорость счета, 1 - однократная скорость счета.

Управляющий байт

После определения счетчика и режима счета Вы можете программировать динамические параметры счетчика. Каждый быстрый счетчик имеет управляющий байт, который активизирует или блокирует счетчик, а также устанавливает, в каком направлении должен происходить счет (только режимы 0, 1 и 2). Управляющий байт задает также начальное направление счета для всех других режимов счета, а также текущее и предварительно установленное значения, которые должны загружаться. Управляющий байт и назначенные текущее и предварительно установленное значения проверяются при выполнении операции HSC. В табл. 9–8 описаны отдельные управляющие биты.

Таблица 9–8. Управляющие биты для HSC0, HSC1 и HSC2

HSC0	HSC1	HSC2	Описание
SM37.0	SM47.0	SM57.0	После выполнения HDEF не используются (в HSC0 никогда).
SM37.1	SM47.1	SM57.1	После выполнения HDEF не используются (в HSC0 никогда).
SM37.2	SM47.2	SM57.2	После выполнения HDEF не используются (в HSC0 никогда).
SM37.3	SM47.3	SM57.3	Бит управления направлением счета: 0 - назад, 1 - вперед.
SM37.4	SM47.4	SM57.4	Запись направления счета в HSC: 0 - не актуализировать, 1 - актуализировать направление.
SM37.5	SM47.5	SM57.5	Запись нового предварительно установленного значения в HSC: 0 - не актуализировать, 1 - актуализировать предварительно установленное значение.
SM37.6	SM47.6	SM57.6	Запись нового текущего значения в HSC: 0 - не актуализировать, 1 - актуализировать текущее значение.
SM37.7	SM47.7	SM57.7	Разблокировка HSC: 0 - HSC блокировать, 1 - HSC активизировать.

Установка текущего и предварительно установленного значений

Каждый быстрый счетчик имеет в своем распоряжении текущее значение и предварительно установленное значение размером 32 бита в каждом случае. Оба значения являются целыми числами со знаком. Чтобы загрузить новое текущее или предварительно установленное значение в быстрый счетчик, Вам нужно настроить управляющий байт и байты специальных меркеров, которые содержат текущие и/или предварительно установленные значения. Потом выполните операцию HSC, чтобы передать новые значения в быстрые счетчики. Таблица 9–9 описывает байты специальных меркеров, которые содержат новые текущие и предварительно установленные значения.

Дополнительно к управляющим байтам и байтам, содержащим новые текущие и предварительно установленные значения, можно считывать текущее значение быстрого счетчика также путем задания области памяти HC (текущее значение быстрого счетчика) и номера счетчика (0, 1 или 2). Таким способом, Вы можете считывать текущее значение непосредственно. Однако для записи Вам нужно использовать описанную выше операцию HSC.

Таблица 9–9. Текущее и предварительно установленное значения HSC0, HSC1 и HSC2
Текущее значение HSC0, HSC1 и HSC2

HSC0	HSC1	HSC2	Описание
SM38	SM48	SM58	Самый старший байт нового текущего значения (32 бита).
SM39	SM49	SM59	Второй по старшинству байт нового текущего значения (32 бита).
SM40	SM50	SM60	Третий по старшинству байт нового текущего значения (32 бита).
SM41	SM51	SM61	Младший байт нового текущего значения (32 бита).
Предварительно установленное значение HSC0, HSC1 и HSC2			
HSC0	HSC1	HSC2	Описание
SM42	SM52	SM62	Самый старший байт нового предварительно установленного значения (32 бита).
SM43	SM53	SM63	Второй по старшинству байт нового предварительно установленного значения (32 бита).
SM44	SM54	SM64	Третий по старшинству байт нового предварительно установленного значения (32 бита).
SM45	SM55	SM65	Младший байт нового предварительно установленного значения (32 бита).

Байт состояния

Каждый быстрый счетчик имеет байт состояния, предоставляющий в распоряжение меркеры состояния. Эти биты состояния задают текущее направление счета. Кроме того, они указывают, является ли текущее значение равным предварительно установленному значению или превышает его. Таблица 9–10 описывает биты состояния быстрых счетчиков.

Таблица 9–10. Биты состояния HSC0, HSC1 и HSC2

HSC0	HSC1	HSC2	Описание
SM36.0	SM46.0	SM56.0	Не используются.
SM36.1	SM46.1	SM56.1	Не используются.
SM36.2	SM46.2	SM56.2	Не используются.
SM36.3	SM46.3	SM56.3	Не используются.
SM36.4	SM46.4	SM56.4	Не используются.
SM36.5	SM46.5	SM56.5	Бит состояния: Текущее направление счета: 0 - назад, 1 - вперед.
SM36.6	SM46.6	SM56.6	Бит состояния: Текущее значение равно предварительно установленному значению: 0 - не равно, 1 - равно.
SM36.7	SM46.7	SM56.7	Бит состояния: Текущее значение больше предварительно установленного значения: 0 - меньше или равно, 1 - больше.

Указание

Биты состояния HSC0, HSC1 и HSC2 действительны только во время обработки программы прерываний для быстрых счетчиков. Когда Вы контролируете состояния быстрых счетчиков, Вы можете разблокировать прерывания для событий, влияющих на обрабатываемую операцию.

HSC–прерывания

HSC0 поддерживает одно условие прерывания: прерывание возникает, когда текущее значение равно предварительно установленному значению. HSC1 и HSC2 имеют в своем распоряжении три условия прерывания: прерывание возникает, когда текущее значение равно предварительно установленному значению, когда извне активизируется вход сброса или когда изменяется направление счета. Каждое из этих условий прерывания может разблокироваться или блокироваться отдельно. Подробную информацию по использованию прерываний Вы найдете в разделе об операциях прерывания.

Приведенные ниже описания инициализации и порядка обработки должны подробнее пояснить Вам принцип функционирования быстрых счетчиков. В описаниях в качестве примера используется HSC1. Объяснения инициализации исходят из того, что S7–200 был только что переведен в режим RUN и поэтому меркер первого цикла имеет значение “истина”. Если это не так, помните, что операция HDEF может выполняться для каждого быстрого счетчика только один раз после того, как система была переведена в режим RUN. Если Вы выполняете операцию HDEF для данного быстрого счетчика во второй раз, то возникает ошибка во время исполнения, а установки счетчика остаются такими, какими они были настроены для данного счетчика с помощью первой операции HDEF.

Инициализация счетных режимов 0, 1 и 2

Чтобы инициализировать HSC1 как однофазный реверсивный счетчик с внутренним управлением направлением счета (режим 0, 1 или 2), действуйте следующим образом:

1. Вызовите с помощью меркера первого цикла подпрограмму, в которой выполняется инициализация. Если Вы вызываете эту подпрограмму, то последующие циклы ее больше не вызывают, за счет чего сокращается время цикла и программа имеет более наглядную структуру.
2. В подпрограмме инициализации загрузите SM47 желаемыми установками. Например:
SM47 = 16#F8, активизирует счетчик
 записывает новое текущее значение
 устанавливает предварительно установленное значение
 устанавливает прямой счет
 устанавливает активность входов запуска и сброса на значение
 “высокая”.
3. Выполните операцию HDEF. При этом вход HSC установлен в “1”, а вход MODE установлен либо в “0”, если нет внешнего сброса или запуска, либо в “1”, если есть внешний сброс и нет запуска, либо в “2”, если есть внешний сброс и запуск.
4. Загрузите желаемое текущее значение в SM48 (двойное слово). (Если Вы загружаете значение “0”, то меркер сбрасывается).
5. Загрузите желаемое предварительно установленное значение в SM52 (двойное слово).
6. Если Вы хотите распознавать момент времени, когда текущее значение равно предварительно установленному значению, то запрограммируйте прерывание, назначив событие прерывания PV = CV (событие 13) программе обработке прерывания. Подробную информацию по обработке прерываний Вы найдете в разделе об операциях прерывания в данной главе.
7. Если Вы хотите распознавать внешний сброс, то запрограммируйте прерывание, назначив событие прерывания “Внешний сброс” (событие 15) программе обработке прерывания.
8. Выполните операцию “Разблокировка всех событий прерывания (ENI)”, чтобы разблокировать прерывания для HSC1.
9. Потом выполните операцию HSC с тем, чтобы S7–200 запрограммировал счетчик HSC1.
10. Закончите подпрограмму.

Инициализация счетных режимов 3, 4 и 5

Чтобы инициализировать HSC1 как однофазный реверсивный счетчик с внешним управлением направлением счета (режимы 3, 4 или 5), действуйте следующим образом:

1. Вызовите с помощью меркера первого цикла подпрограмму, в которой выполняется инициализация. Если Вы вызываете эту подпрограмму, то последующие циклы ее больше не вызывают, за счет чего сокращается время цикла и программа имеет более наглядную структуру.
2. В подпрограмме инициализации загрузите SM47 желаемыми установками. Например:
SM47 = 16#F8, активизирует счетчик
 записывает новое текущее значение
 устанавливает предварительно установленное значение
 устанавливает прямой счет
 устанавливает активность входов запуска и сброса на значение
 “высокая”.
3. Выполните операцию HDEF. При этом вход HSC установлен в “1”, а вход MODE установлен либо в “3”, если нет внешнего сброса или запуска, либо в “4”, если есть внешний сброс и нет запуска, либо в “5”, если есть внешний сброс и запуск.
4. Загрузите желаемое текущее значение в SM48 (двойное слово). (Если Вы загружаете значение “0”, то меркер сбрасывается).
5. Загрузите желаемое предварительно установленное значение в SM52 (двойное слово).
6. Если Вы хотите распознавать момент времени, когда текущее значение равно предварительно установленному значению, то запрограммируйте прерывание, назначив событие прерывания PV = CV (событие 13) программе обработке прерываний. Подробную информацию по обработке прерываний Вы найдете в разделе об операциях прерывания в данной главе.
7. Если Вы хотите распознавать смену направления счета, то запрограммируйте прерывание, назначив событие прерывания “Смена направления счета” (событие 14) программе обработке прерываний.
8. Если Вы хотите распознавать внешний сброс, то запрограммируйте прерывание, назначив событие прерывания “Внешний сброс” (событие 15) программе обработке прерывания.
9. Выполните операцию “Разблокировка всех событий прерывания (ENI)”, чтобы разблокировать прерывания для HSC1.
10. Потом выполните операцию HSC с тем, чтобы S7-200 запрограммировал счетчик HSC1.
11. Закончите подпрограмму.

Инициализация счетных режимов 6, 7 и 8

Для того, чтобы инициализировать HSC1 как двухфазный реверсивный счетчик с датчиками тактовых импульсов прямого/обратного счета (режим 6, 7 или 8), действуйте следующим образом:

1. Вызовите с помощью меркера первого цикла подпрограмму, в которой выполняется инициализация. Если Вы вызываете эту подпрограмму, то последующие циклы ее больше не вызывают, за счет чего сокращается время цикла и программа имеет более наглядную структуру.
2. В подпрограмме инициализации загрузите SM47 желаемыми установками. Например:
SM47 = 16#F8, активизирует счетчик
 записывает новое текущее значение
 записывает новое предварительно установленное значение
 устанавливает прямой счет
 устанавливает активность входов запуска и сброса на значение
 “высокая”.
3. Выполните операцию HDEF. При этом вход HSC установлен в “1”, а вход MODE установлен либо в “6”, если нет внешнего сброса или запуска, либо в “7”, если есть внешний сброс и нет запуска, либо в “8”, если есть внешний сброс и запуск.
4. Загрузите желаемое текущее значение в SM48 (двойное слово). (Если Вы загружаете значение “0”, то меркер сбрасывается).
5. Загрузите желаемое предварительно установленное значение в SM52 (двойное слово).
6. Если Вы хотите распознавать момент времени, когда текущее значение равно предварительно установленному значению, то запрограммируйте прерывание, назначив событие прерывания PV = CV (событие 13) программе обработке прерываний. Подробную информацию по обработке прерываний Вы найдете в разделе об операциях прерывания в данной главе.
7. Если Вы хотите распознавать смену направления счета, то запрограммируйте прерывание, назначив событие прерывания “Смена направления счета” (событие 14) программе обработке прерываний.
8. Если Вы хотите распознавать внешний сброс, то запрограммируйте прерывание, назначив событие прерывания “Внешний сброс” (событие 15) программе обработке прерываний.
9. Выполните операцию “Разблокировка всех событий прерывания (ENI)”, чтобы разблокировать прерывания для HSC1.
10. Потом выполните операцию HSC с тем, чтобы S7-200 запрограммировал счетчик HSC1.
11. Закончите подпрограмму.

Инициализация счетных режимов 9, 10 и 11

Для того, чтобы инициализировать HSC1 как A/B-счетчик (режим 9, 10 или 11), действуйте следующим образом:

1. Вызовите с помощью меркера первого цикла подпрограмму, в которой выполняется инициализация. Если Вы вызываете эту подпрограмму, то последующие циклы ее больше не вызывают, за счет чего сокращается время цикла и программа имеет более наглядную структуру.
2. В подпрограмме инициализации загрузите SM47 желаемыми установками.
Пример (однократная скорость счета):
SM47 = 16#FC, активизирует счетчик
 записывает новое текущее значение
 записывает новое предварительно установленное значение
 устанавливает прямой счет
 устанавливает активность входов запуска и сброса на значение “высокая”.
Пример (четырёхкратная скорость счета):
SM47 = 16#F8, активизирует счетчик
 записывает новое текущее значение
 записывает новое предварительно установленное значение
 устанавливает прямой счет
 устанавливает активность входов запуска и сброса на значение “высокая”.
3. Выполните операцию HDEF. При этом вход HSC установлен в “1”, а вход MODE установлен либо в “9”, если нет внешнего сброса или запуска, либо в “10”, если есть внешний сброс и нет запуска, либо в “11”, если есть внешний сброс и запуск.
4. Загрузите желаемое текущее значение в SM48 (двойное слово). (Если Вы загружаете значение “0”, то меркер сбрасывается).
5. Загрузите желаемое предварительно установленное значение в SM52 (двойное слово).
6. Если Вы хотите распознавать момент времени, когда текущее значение равно предварительно установленному значению, то запрограммируйте прерывание, назначив событие прерывания PV = CV (событие 13) программе обработке прерываний. Подробную информацию по обработке прерываний Вы найдете в разделе об операциях прерывания в данной главе.
7. Если Вы хотите распознавать смену направления счета, то запрограммируйте прерывание, назначив событие прерывания “Смена направления счета” (событие 14) программе обработке прерывания.
8. Если Вы хотите распознавать внешний сброс, то запрограммируйте прерывание, назначив событие прерывания “Внешний сброс” (событие 15) программе обработке прерывания.
9. Выполните операцию “Разблокировка всех событий прерывания (ENI)”, чтобы разблокировать прерывания для HSC1.
10. Потом выполните операцию HSC с тем, чтобы S7–200 запрограммировал счетчик HSC1.
11. Закончите подпрограмму.

Смена направления в режиме 0, 1 или 2

Для того, чтобы конфигурировать направление счета для HSC1 в качестве однофазного счетчика с внутренним управлением направлением счета (режим 0, 1 или 2), действуйте следующим образом:

1. Загрузите SM47, чтобы установить желаемое направление счета.
SM47 = 16#90, активизирует счетчик
устанавливает направление счета HSC на обратный счет
SM47 = 16#98, активизирует счетчик
устанавливает направление счета HSC на прямой счет.
2. Выполните операцию HSC с тем, чтобы S7–200 запрограммировал счетчик HSC1.

Загрузка нового текущего значения (произвольный режим)

Чтобы изменить текущее значение HSC1 (произвольный режим), действуйте следующим образом:

Когда Вы изменяете текущее значение, счетчик автоматически блокируется. Пока он заблокирован, он не считает и не генерирует прерываний.

1. Загрузите SM47, чтобы ввести желаемое текущее значение.
SM47 = 16#C0, активизирует счетчик.
записывает новое текущее значение.
2. Загрузите желаемое текущее значение в SM48 (двойное слово). (Если Вы загружаете значение "0", то меркер сбрасывается).
3. Выполните операцию HSC с тем, чтобы S7–200 запрограммировал счетчик HSC1.

Загрузка нового предварительно установленного значения (произвольный режим)

Чтобы изменить предварительно установленное значение HSC1 (произвольный режим), действуйте следующим образом:

1. Загрузите SM47, чтобы ввести желаемое предварительно установленное значение.
SM47 = 16#A0, активизирует счетчик.
записывает новое предварительно установленное значение.
2. Загрузите желаемое предварительно установленное значение в SM52 (двойное слово).
3. Выполните операцию HSC с тем, чтобы S7–200 запрограммировал счетчик HSC1.

Блокировка HSC (произвольный режим)

Чтобы блокировать быстрый счетчик HSC1 (произвольный режим), действуйте следующим образом:

1. Загрузите SM47, чтобы блокировать быстрый счетчик.
SM47 = 16#00, блокирует счетчик.
2. Выполните операцию HSC с тем, чтобы S7–200 запрограммировал счетчик HSC1.

Описанные выше способы действий показывают Вам, как отдельно изменить направление счета, текущее значение или предварительно установленное значение. Однако Вы можете изменить также несколько или все установки в приведенной выше последовательности, настраивая значение SM47 соответствующим образом и выполняя затем операцию HSC.

Пример быстрого счетчика

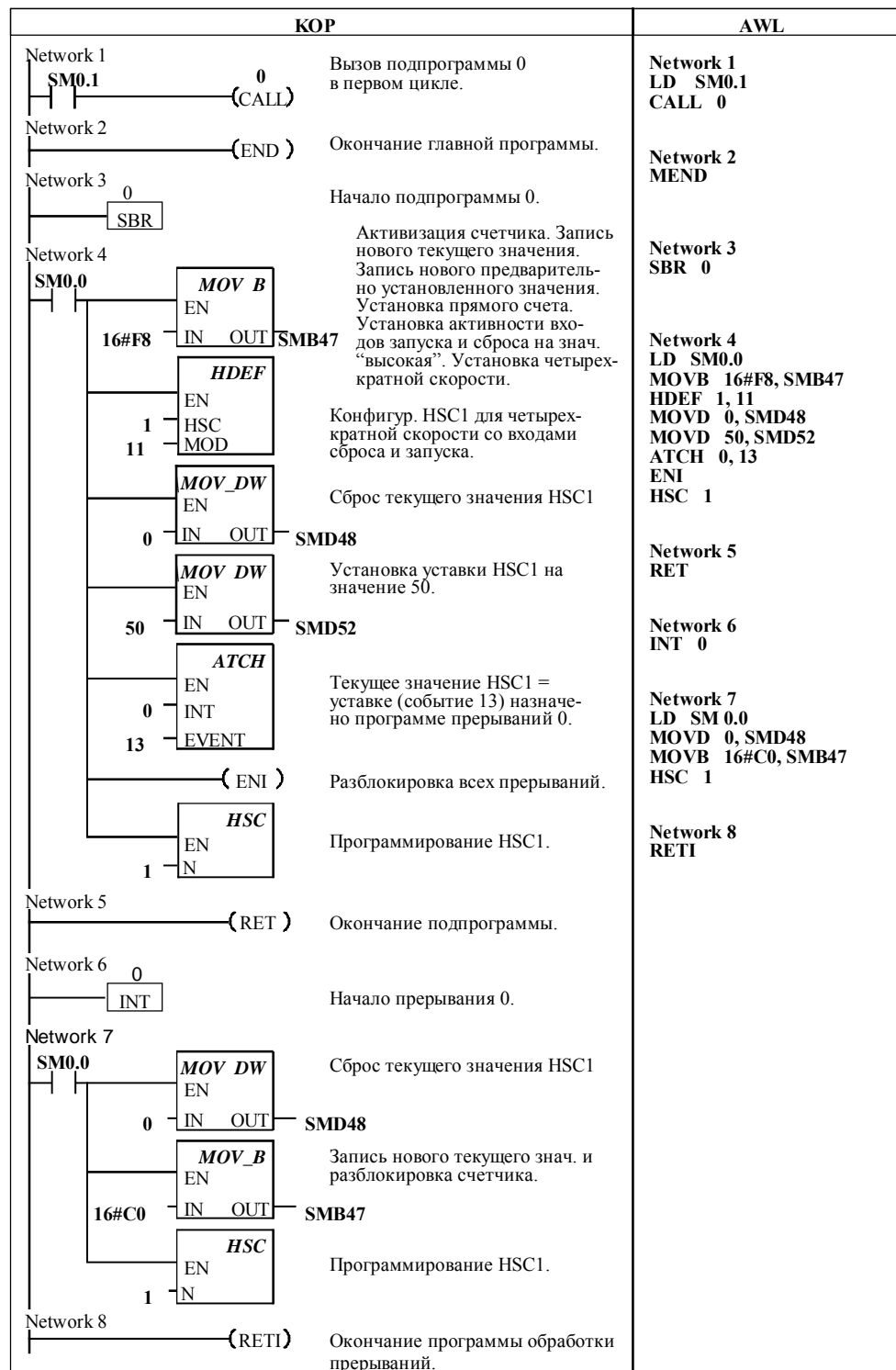
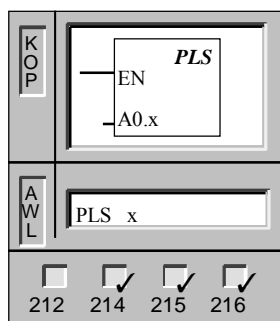


Рис. 9-52. Пример инициализации HSC1 в форме KOP и AWL

9.14 Операции с быстрыми выходами

Вывод импульсов



Операция **Вывод импульсов** проверяет специальные меркеры для этого импульсного выхода (x). Потом вызывается импульсная операция, определенная в специальных меркерах.

Операнды: x: от 0 до 1

Описание операций с быстрыми выходами для S7-200

Некоторые CPU могут через выходы A0.0 и A0.1 либо порождать быстрые последовательности импульсов (PTO = pulse train output), либо управлять широтно-импульсной модуляцией (PWM = pulse width modulation). Функция “Последовательность импульсов” обеспечивает на выходе последовательность прямоугольных импульсов (относительная длительность включения 50%) с определенным количеством импульсов и фиксированным периодом следования. Количество импульсов может лежать в диапазоне от 1 до 4.294.967.295. Период может задаваться в микросекундах (от 250 до 65.535) или в миллисекундах (от 2 до 65.535). Нечетное число микросекунд или миллисекунд вызывает искажение относительной длительности включения.

Функция PWM (ШИМ) обеспечивает фиксированный период следования импульсов с переменной относительной длительностью включения. Период следования и длительность импульсов могут задаваться в микро- или миллисекундах. Период следования лежит в диапазоне от 250 до 65.535 микросекунд или в диапазоне от 2 до 65.535 миллисекунд. Длительность импульсов лежит в диапазоне от 0 до 65.535 микросекунд или в диапазоне от 0 до 65.535 миллисекунд. Если длительность импульсов и период следования равны, то относительная длительность включения составляет 100%, и выход является постоянно включенным. Если длительность импульсов равна нулю, то относительная длительность включения составляет 0%, и выход выключается.

Если период следования задается меньшим, чем две единицы времени, то период следования принимает значение по умолчанию, равное двум единицам времени.

Указание

В функциях PTO и PWM времена переключения выходов различны для переходов от включенного состояния к выключенному и наоборот. Такое различие во временах переключения проявляется в виде искажения относительной длительности включения. Подробные данные по временам переключения Вы найдете в приложении F. Выходы PTO/PWM должны иметь минимальную нагрузку, не более 10% от номинальной, чтобы иметь возможность обеспечивать четкие переходы (“ВКЛ/ВЫКЛ” и “ВЫКЛ/ВКЛ”).

Изменение длительности импульса

Функция PWM выполняется непрерывно. Изменение длительности импульсов вызывает кратковременное блокирование функции PWM на время актуализации. Это происходит асинхронно по отношению к циклу PWM и могло бы вызвать нежелательное дрожание импульсов в управляемой установке. Если необходима синхронная актуализация длительности импульсов, то импульсный выход заводится на один из входов прерываний (E0.0 - E0.4). Вы можете синхронизировать цикл PWM (см. рис. 9–57) за счет того, что в момент времени, когда должна изменяться длительность импульсов, разблокируется прерывание по нарастающему фронту на входе, на который заведен импульсный выход.

Длительность импульсов изменяется в программе обработки прерывания, а событие прерывания отделяется от программы или блокируется. Благодаря этому прерывания возникают только во время изменения длительности импульсов.

Вызов функций РТО/PWM

Каждый генератор РТО/PWM имеет в своем распоряжении управляющий байт (8 битов), по мере надобности одно значение для периода следования и для длительности импульсов (оба являются 16–битными значениями без знака) и значение счетчика импульсов (32–битное значение без знака). Эти значения хранятся в определенных специальных меркерах. После конфигурирования специальных меркеров для специфической функции, Вы можете вызвать эту функцию с помощью операции “Вывод импульсов” (PLS). Когда эта операция выполняется, контроллер S7–200 считывает специальные меркеры и программирует генератор РТО/PWM в соответствии с конфигурацией.

Конвейер РТО

В дополнение к управляющей информации операция РТО использует два бита состояния, которые показывают, было ли сгенерировано заданное количество импульсов и не возникло ли переполнение конвейера.

Функция РТО допускает связь двух определенных импульсных выходов в цепочку или их организацию по принципу конвейера. За счет этого может поддерживаться непрерывность следующих друг за другом импульсных последовательностей на выходах. Для загрузки конвейера вначале формируется первое определение РТО, а затем выполняется операция PLS. Непосредственно после выполнения операции PLS Вы можете сформировать второе определение и снова выполнить операцию PLS.

Если третье определение создается прежде, чем закончилась первая операция РТО (т.е. до того, как было сгенерировано полное количество выходных импульсов первого определения), то устанавливается в “1” бит “Переполнение конвейера РТО” (SM66.6 или SM76.6). При переходе в режим RUN этот бит сбрасывается в “0”. После обнаружения переполнения этот бит должен сбрасываться в “0” программой, чтобы можно было обнаруживать последующие переполнения.

В табл. 9–11 показаны специальные меркеры для импульсных выходов 0 и 1.

Указание

По умолчанию управляющие биты, период следования, длительность импульсов и значения счетчика импульсов установлены в ноль.

Таблица 9–11. Специальные меркеры для конвейерного соединения двух импульсных выходов с помощью функций PTO и PWM

A 0.0	A 0.1	Биты состояния для импульсных выходов
SM66.6	SM76.6	Переполнение конвейера PTO 0 - нет переполнения, 1 - переполнение
SM66.7	SM76.7	Холостой ход PTO обработка, 1 - холостой ход
A 0.0	A 0.1	Биты управления для выходов PTO/PWM
SM67.0	SM77.0	PTO/PWM: актуализация периода следования 0 - нет актуализации, 1 - актуализация периода следования
SM67.1	SM77.1	PWM: актуализация длительности импульсов 0 - нет актуализации, 1 - актуализация длительности импульсов
SM67.2	SM77.2	PTO: актуализация значения счетчика импульсов актуализации, 1 - актуализация счетчика импульсов
SM67.3	SM77.3	PTO/PWM: выбор базы времени
SM67.4	SM77.4	Не используется.
SM67.5	SM77.5	Не используется.
SM67.6	SM77.6	PTO/PWM: выбор функции
SM67.7	SM77.7	PTO/PWM: разблокировка - блокировка PTO/PWM, 1 - разблокировка PTO/PWM
A 0.0	A 0.1	Значения периода следования для выходов PTO/PWM (диапазон: 2 - 65.535)
SM68	SM78	Старший байт периода следования PTO/PWM
SM69	SM79	Младший байт периода следования PTO/PWM
A 0.0	A 0.1	Значения длительности импульсов для выходов PWM (диапазон: 0 - 65.535)
SM70	SM80	Старший байт длительности импульсов PWM
SM71	SM81	Младший байт длительности импульсов PWM
A 0.0	A 0.1	Значения счетчика импульсов для импульсных выходов (диапазон: 1 - 4.294.967.295)
SM72	SM82	Самый старший байт значения счетчика импульсов PTO
SM73	SM83	Второй по старшинству байт значения счетчика импульсов PTO
SM74	SM84	Третий по старшинству байт значения счетчика импульсов PTO
SM75	SM85	Младший байт значения счетчика импульсов PTO

С помощью таблицы 9–12 Вы можете быстро установить значение, которое Вам нужно записать в управляющий регистр PTO/PWM, чтобы вызвать желаемую операцию. Используйте SMB67 для PTO/PWM0 и SMB77 для PTO/PWM1. Если Вы хотите загрузить новое значение счетчика импульсов (SMD72 или SMD82), длительность импульса (SMW70 или SMW80) или период следования (SMW68 или SMW78), то Вы должны загрузить эти значения точно так же, как управляющий регистр перед выполнением операции PLS.

Таблица 9–12. Справочная таблица по шестнадцатиричным значениям PTO/PWM						
Управляющий регистр (16-ричное значение)	Результат выполнения операции PLS					
	Разблокировка	Функция	База времени	Значение счетчика импульсов	Длительность импульсов	Период следования
16#81	Да	PTO	1 мкс/такт			Загрузка
16#84	Да	PTO	1 мкс/такт	Загрузка		
16#85	Да	PTO	1 мкс/такт	Загрузка		Загрузка
16#89	Да	PTO	1 мкс/такт			Загрузка
16#8C	Да	PTO	1 мкс/такт	Загрузка		
16#8D	Да	PTO	1 мкс/такт	Загрузка		Загрузка
16#C1	Да	PWM	1 мкс/такт			Загрузка
16#C2	Да	PWM	1 мкс/такт		Загрузка	
16#C3	Да	PWM	1 мкс/такт		Загрузка	Загрузка
16#C9	Да	PWM	1 мкс/такт			Загрузка
16#CA	Да	PWM	1 мкс/такт		Загрузка	Загрузка
16#CB	Да	PWM	1 мкс/такт		Загрузка	Загрузка

Инициализация и выполнение функций PTO/PWM

В следующих разделах описываются способы действий по инициализации и организации функций PTO и PWM. Они помогут Вам понять их принцип действия. В этих описаниях используется выход A0.0 и предполагается, что перед этим S7–200 был переведен в режим RUN и поэтому меркер первого цикла имеет значение “истина”. Если это не так или функция PTO/PWM должна быть инициализирована снова, то Вы можете вызвать программу инициализации с помощью другого условия, а не с помощью меркера первого цикла.

Инициализация функции PWM

Для инициализации функции PWM на выходе A0.0 действуйте следующим образом:

1. Установите выход в “1” с помощью меркера первого цикла и вызовите подпрограмму, в которой выполняется инициализация. Если Вы вызываете эту подпрограмму, то последующие циклы ее больше не вызывают за счет чего сокращается время цикла и программа имеет более наглядную структуру.
2. Загрузите значение 16#C3 в SM67 в подпрограмме инициализации, указывая этим для функции PWM, что инкрементирование должно производиться в микросекундах (Вы можете загрузить также значение 16#CB, если хотите в функции PWM инкрементировать в миллисекундах). Эти шестнадцатиричные значения устанавливают управляющий байт, с помощью которого разблокируется операция PTO/PWM и выбирается функция PWM. Кроме того, управляющий байт указывает, что инкрементирование производится в микро- или миллисекундах и что должны актуализироваться значения для длительности импульса и периода следования.
3. Загрузите желаемый период следования в SM68 (слово).
4. Загрузите желаемую длительность импульса в SM70 (слово).
5. Выполните операцию PLS, чтобы S7–200 запрограммировал генератор PTO/PWM.
6. Загрузите значение 16#C2 в SM67, чтобы инкрементирование происходило в микросекундах (или 16#CA в случае миллисекунд). Этот сбрасывает также задание на актуализацию длительности следования в управляющем байте и позволяет изменять длительность импульса. Загружается новое значение для длительности импульса. Затем выполняется операция PLS без изменения управляющего байта.
7. Закончите подпрограмму.

Необязательные действия для синхронной актуализации. Если требуется синхронная актуализация, то действуйте следующим образом:

1. Выполните операцию “Разблокировка всех событий прерываний” (ENI).
2. С помощью условия, посредством которого Вы актуализируете длительность импульса, поставьте в соответствие прерыванию по нарастающему фронту программу обработки прерываний (ATCH). Условие, используемое для сопоставления события, может быть активным в течение только одного цикла.
3. Добавьте программу обработки прерываний, которая актуализирует длительность импульса, а затем блокирует прерывание по нарастающему фронту.

Указание

Необязательные действия для синхронной актуализации требуют, чтобы выход PWM был подключен по цепи обратной связи к одному из входов прерываний.

Изменение длительности импульсов для выходов PWM

Для изменения длительности импульсов для выходов PWM в подпрограмме действуйте следующим образом:

1. Вызовите подпрограмму, чтобы загрузить желаемую длительность импульсов в SM70 (слово).
2. Выполните операцию PLS, чтобы S7–200 запрограммировал генератор PTO/PWM.
3. Закончите подпрограмму.

Инициализация функции РТО

Для инициализации функции РТО действуйте следующим образом:

1. Сбросьте выход в “0” с помощью меркера первого цикла и вызовите подпрограмму, в которой выполняется инициализация. Если Вы вызываете эту подпрограмму, то последующие циклы ее больше не вызывают за счет чего сокращается время цикла и программа имеет более наглядную структуру.
2. Загрузите в подпрограмме инициализации значение 16#85 в SM67, указывая этим для функции РТО, что инкрементирование должно производиться в микросекундах (Вы можете загрузить также значение 16#8D, если Вы хотите в функции РТО инкрементировать в миллисекундах). Эти шестнадцатиричные значения устанавливают управляющий байт, с помощью которого разблокируется операция РТО/PWM и выбирается функция РТО. Кроме того, управляющий байт указывает, что инкрементирование производится в микро- или миллисекундах и что должны обновляться значения счетчика и периода следования импульсов.
3. Загрузите желаемый период следования в SM68 (слово).
4. Загрузите желаемое количество импульсов в SM72 (слово).
5. Этот шаг является необязательным: если Вы хотите после выполнения операции “Последовательность импульсов” выполнить поставленную в соответствие функцию, то Вы можете запрограммировать прерывание, сопоставляя событие “Последовательность импульсов закончена” (класс прерываний 19) программе обработки прерываний и выполняя операцию “Разблокировка всех событий прерываний”. Подробную информацию по обработке прерываний Вы найдете в разделе 1.17.
6. Выполните операцию PLS, чтобы S7-200 запрограммировал генератор РТО/PWM.
7. Закончите подпрограмму.

Изменение периода следования в функции РТО

Для изменения периода следования в программе обработки прерываний или подпрограмме действуйте следующим образом:

1. Загрузите значение 16#81 в SM67, указывая этим для функции РТО, что инкрементирование должно производиться в микросекундах (Вы можете загрузить также значение 16#89, если Вы хотите инкрементировать в миллисекундах). Эти шестнадцатиричные значения устанавливают управляющий байт, с помощью которого разблокируется операция РТО/PWM и выбирается функция РТО. Кроме того, управляющий байт указывает, что инкрементирование производится в микро- или миллисекундах и что должно обновляться значение для периода следования.
2. Загрузите желаемый период следования в SM68 (слово).
3. Выполните операцию PLS, чтобы S7-200 запрограммировал генератор РТО/PWM.
4. Закончите программу обработки прерываний или подпрограмму. (Подпрограммы не могут вызываться из программ обработки прерываний.)

Изменение значения счетчика в функции РТО

Для изменения значения счетчика в программе обработки прерываний или подпрограмме действуйте следующим образом:

1. Загрузите значение 16#84 в SM67, указывая этим для функции РТО, что инкрементирование должно производиться в микросекундах (Вы можете загрузить также значение 16#8C, если Вы хотите инкрементировать в миллисекундах). Эти шестнадцатиричные значения устанавливают управляющий байт, с помощью которого разблокируется операция РТО/PWM и выбирается функция РТО. Кроме того, управляющий байт указывает, что инкрементирование производится в микро- или миллисекундах и что должно обновляться значение счетчика.
2. Загрузите желаемое значение счетчика в SM72 (двойное слово).
3. Выполните операцию PLS, чтобы S7-200 запрограммировал генератор РТО/PWM.
4. Закончите программу обработки прерываний или подпрограмму. (Подпрограммы не могут вызываться из программ обработки прерываний.)

Изменение периода следования и значения счетчика импульсов в функции РТО

Для изменения периода следования и значения счетчика импульсов в программе обработки прерываний или подпрограмме действуйте следующим образом:

1. Загрузите значение 16#85 в SM67, указывая этим для функции РТО, что инкрементирование должно производиться в микросекундах (Вы можете загрузить также значение 16#8D, если Вы хотите инкрементировать в миллисекундах). Эти шестнадцатиричные значения устанавливают управляющий байт, с помощью которого разблокируется операция РТО/PWM и выбирается функция РТО. Кроме того, управляющий байт указывает, что инкрементирование производится в микро- или миллисекундах и что должно актуализироваться значение для периода следования и значения счетчика.
2. Загрузите желаемый период следования в SM68 (слово).
3. Загрузите желаемое значение счетчика в SM72 (слово).
4. Выполните операцию PLS, чтобы S7-200 запрограммировал генератор РТО/PWM.
5. Закончите программу обработки прерываний или подпрограмму. (Подпрограммы не могут вызываться из программ обработки прерываний.)

Активные функции РТО/PWM

Если одна из функций РТО и PWM активна на выходе A0.0 или A0.1, то соответствующий выход блокируется. Ни значения, записанные для этого выхода в области отображения процесса, ни принудительно установленные значения не передаются на выход, пока активна одна из функций РТО или PWM. Операция РТО активна, когда она разблокирована и не завершена. Прямые операции над выходами, которые записывают в эти выходы в то время, когда функция РТО или PWM активна, не вызывают искажения формы сигнала в обеих этих функциях.

Влияние на выходы

Функция РТО/PWM и область отображения процесса используют выходы A0.0 и A0.1 совместно. Начальное и конечное состояния форм сигналов функций РТО и PWM испытывают влияние со стороны соответствующей области отображения процесса. Если A0.0 или A0.1 выводят последовательность импульсов, то область отображения процесса определяет начальное и конечное состояния выходов и побуждает импульсный выход запускаться на высоком или низком уровне.

При изменениях в конвейере РТО и в длительности импульсов PWM функции РТО и PWM кратковременно блокируются. Вследствие этого может произойти небольшой разрыв в форме сигналов выходов. Чтобы минимизировать вредное влияние таких разрывов, сбрасывайте в "0" бит функции РТО и устанавливайте в "1" бит функции PWM в области отображения процесса. На рис. 9-54 показаны результирующие формы сигналов функций РТО и PWM. Обратите внимание на то, что в случае функции РТО в точке изменения последний полупериод сокращается на длительность импульса величиной примерно 120 микросекунд. Если в функции PWM используются необязательные действия для синхронной актуализации, то первый импульс после перехода продлится примерно на 120 микросекунд.



Рис. 9-53. Пример формы импульсных последовательностей на A0.0 или A0.1

Пример последовательности импульсов (PTO)

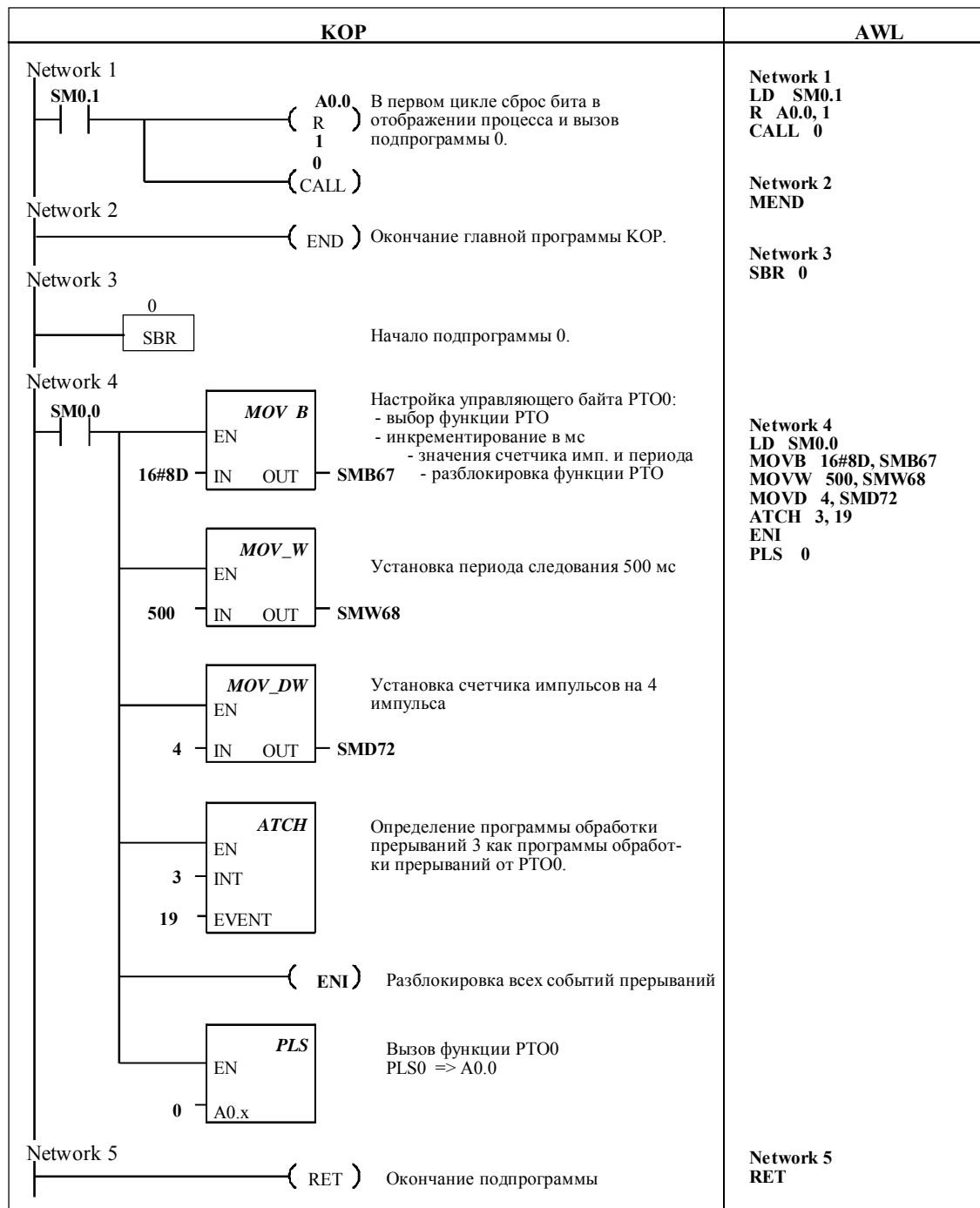


Рис. 9-54. Пример последовательности импульсов

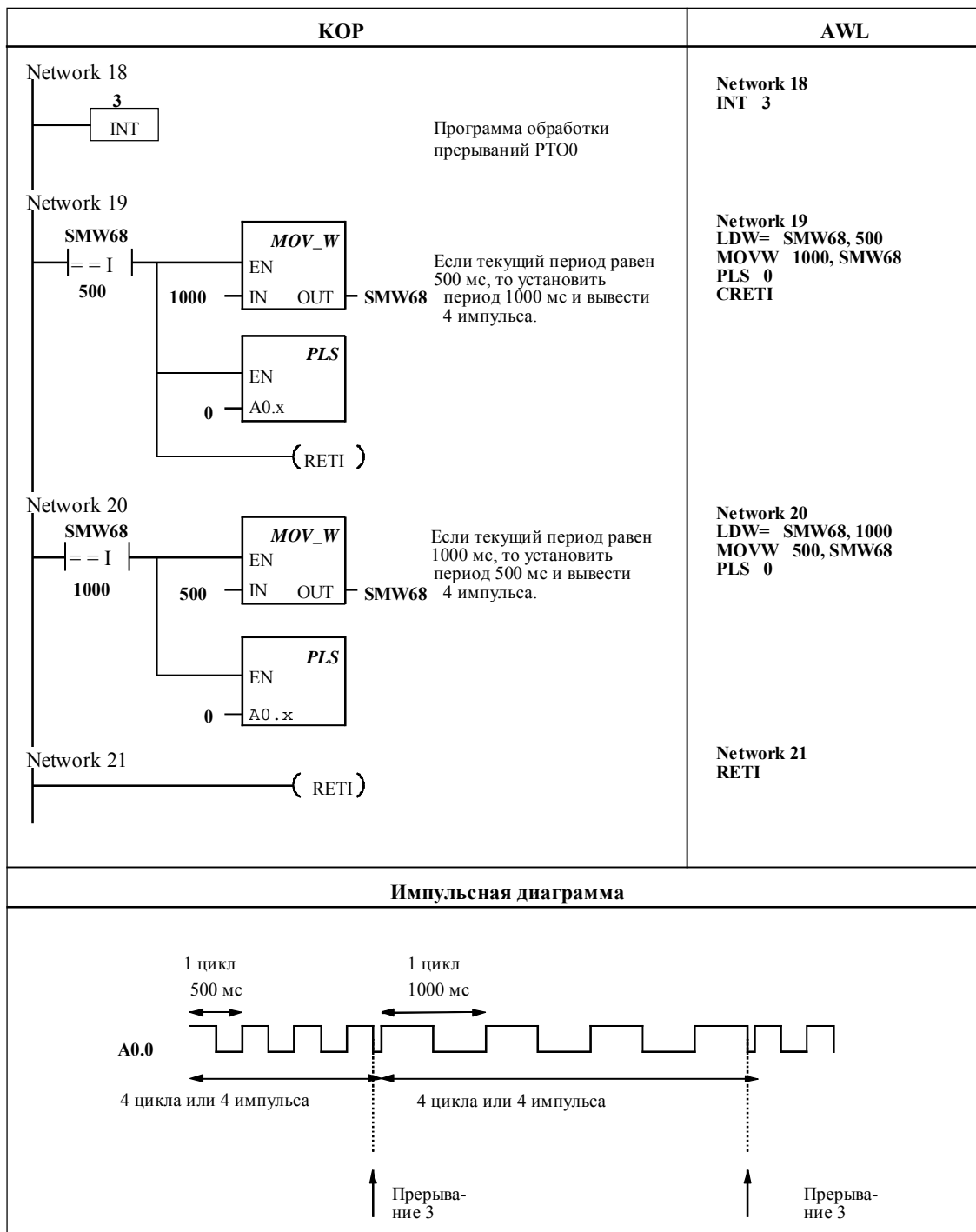


Рис. 9-54. Пример последовательности импульсов, продолжение

Пример широтно-импульсной модуляции

На рис. 9–57 показан пример широтно-импульсной модуляции. При изменении длительности импульсов функция PWM кратковременно блокируется, чтобы можно было изменить значение. Это происходит асинхронно по отношению к циклу PWM. Поэтому в управляемой установке может возникнуть нежелательное дрожание импульсов. Если длительность импульсов должна актуализироваться синхронно, то импульсный выход заводится по цепи обратной связи на вход прерываний (E0.0). Если длительность импульсов должна измениться, то разблокируется вход прерываний, так что при следующем нарастающем фронте на E0.0 длительность импульсов изменяется синхронно по отношению к циклу PWM. Собственно говоря, длительность импульсов изменяется в программе обработки прерываний. Событие прерывания блокируется или отделяется в программе обработки прерываний. Благодаря этому прерывание возникает только лишь тогда, когда должна измениться длительность импульсов.

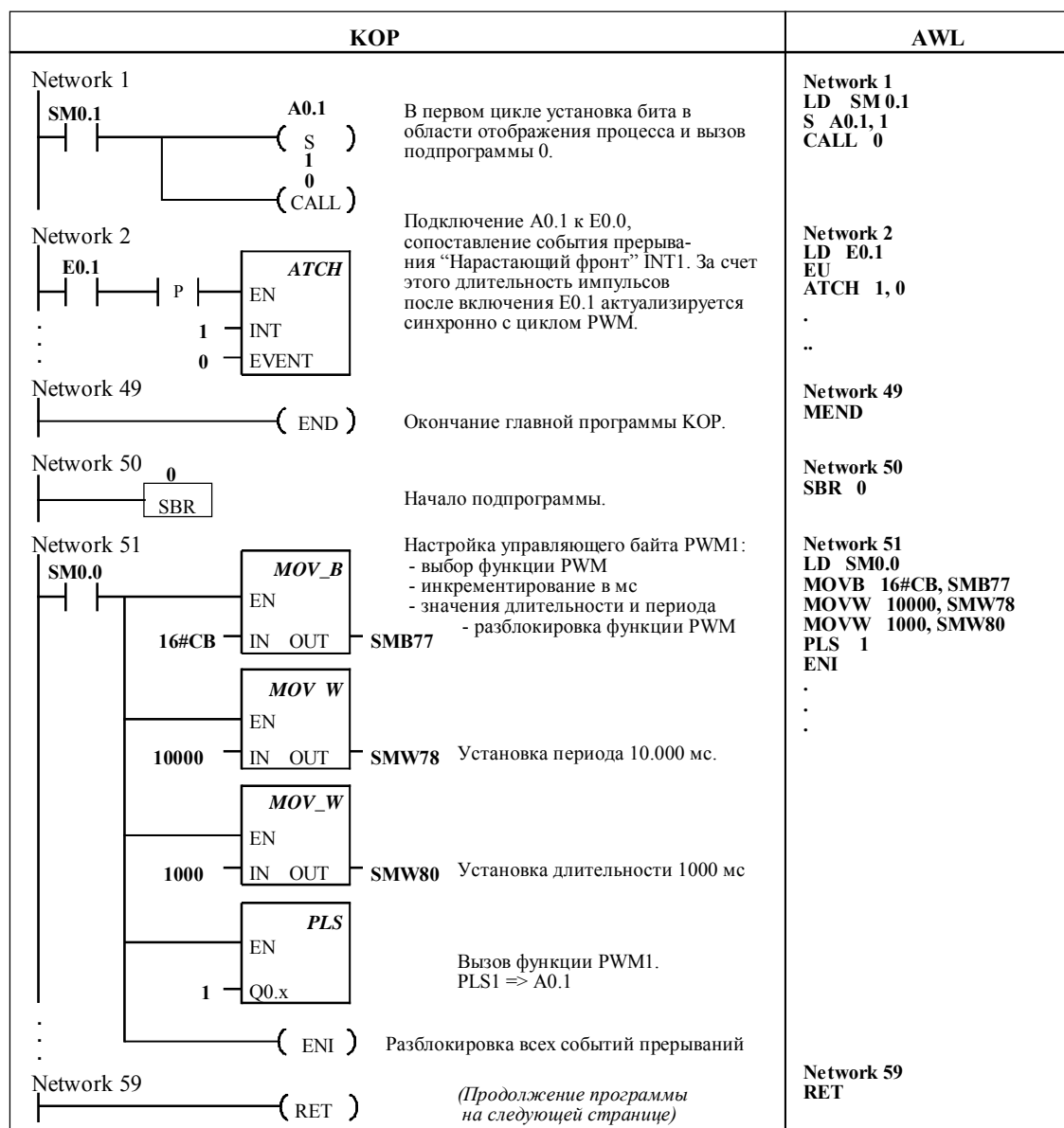


Рис. 9-55. Пример быстрого выхода с широтно-импульсной модуляцией

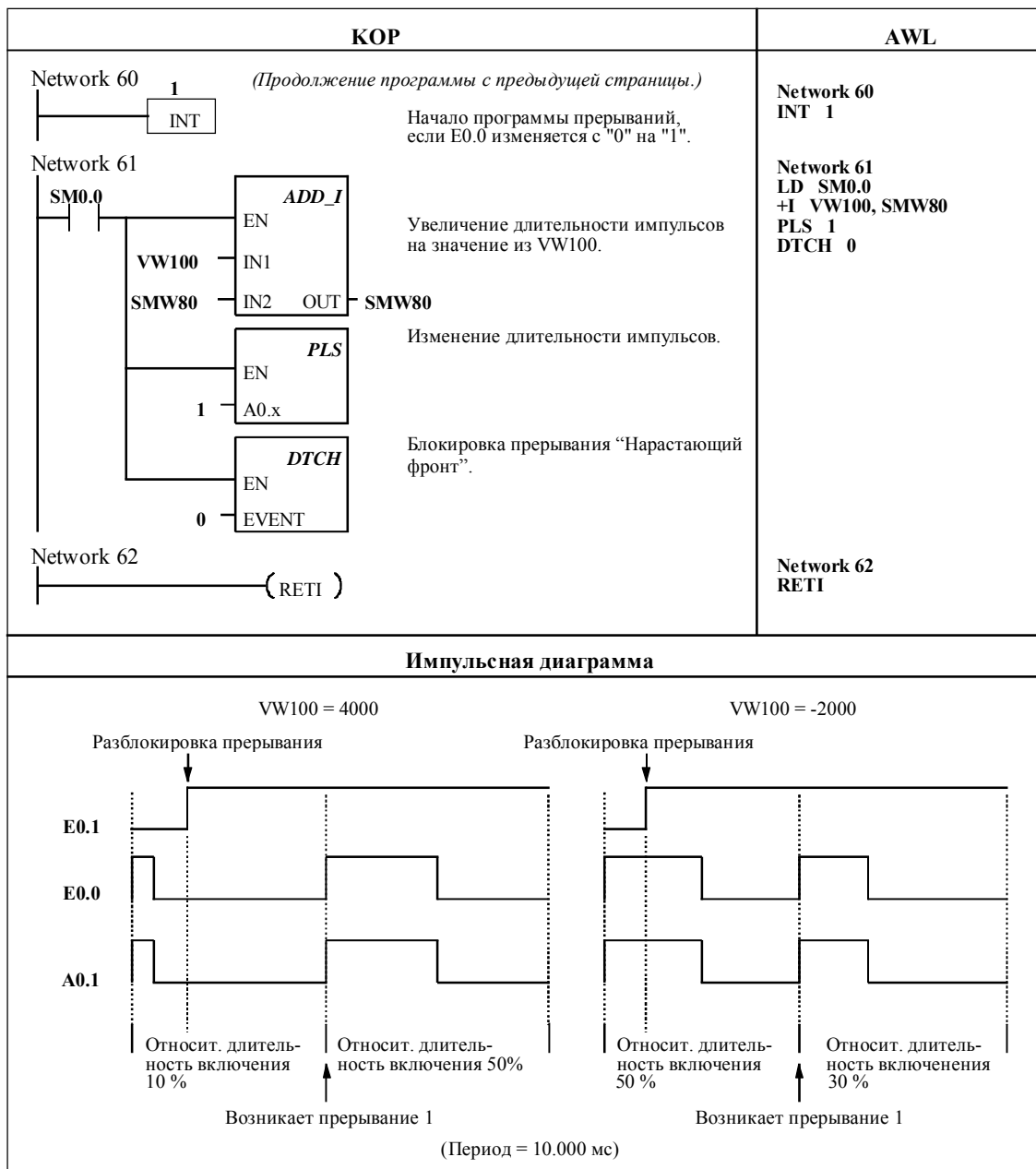
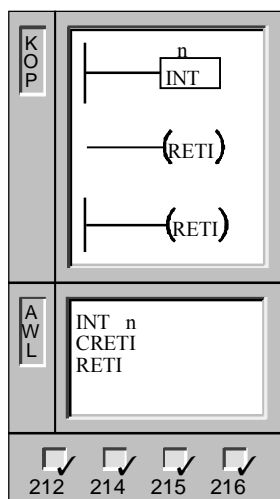


Рис. 9-55. Пример быстрого выхода с широтно-импульсной модуляцией, продолжение

9.15 Прерывания

**Начало программы обработки прерываний и
обработки прерываний**

окончание программы



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

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

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

Операнды: n: от 0 до 127

Программы обработки прерываний

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

Указания по использованию программ обработки прерываний

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

Ограничения

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

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

Системная поддержка прерываний

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

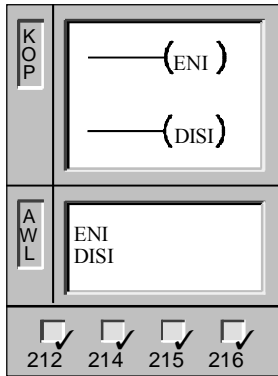
Совместное использование данных в главной программе и программе обработки прерываний

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

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

- Для программы на AWL, которая совместно использует одну единственную переменную: если совместно используется только одна переменная в виде байта, слова или двойного слова и Ваша программа написана в форме AWL, то промежуточные результаты операций с совместно используемыми данными следует записывать только по адресам памяти или в аккумуляторах, которые не используются совместно.
- Для программы на KOP, которая совместно использует одну единственную переменную: если совместно используется только одна переменная в виде байта, слова или двойного слова и Ваша программа написана в форме KOP, то обращение к совместно используемым адресам памяти следует производить только с помощью операций передачи (MOV_B, MOV_W, MOV_DW, MOV_R). Многие операции в KOP соответствуют последовательностям команд в AWL, которые могут прерываться. Однако каждая из этих операций передачи соответствует одной единственной команде AWL, обработка которой не может испытывать воздействия событий прерываний.
- Для программ на AWL или KOP, которые совместно используют несколько переменных: если совместно используются нескольких взаимосвязанных байтов, слов или двойных слов, то исполнением программы обработки прерываний можно управлять посредством операций “Блокировка всех событий прерываний” (DISI) и “Разблокировка всех событий прерываний” (ENI). В том месте главной программы, где Вы располагаете операции, выполняющие доступ к совместной памяти, Вы должны блокировать события прерываний. После того, как выполнены все операции, работающие с совместной памятью, Вы должны снова разблокировать события прерываний. В течение времени, когда события прерываний заблокированы, программы обработки прерываний не могут выполняться и не имеют доступа к совместной памяти. Однако, такой способ программирования может вызывать замедленную реакцию на события прерывания.

Разблокировка и блокировка всех событий прерываний



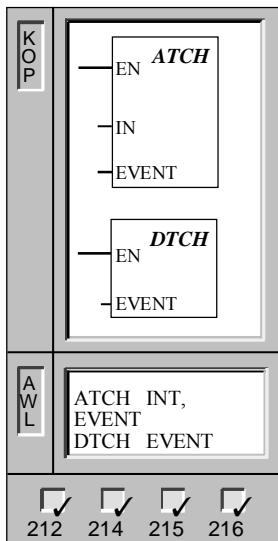
Операция **Разблокировка всех событий прерываний** разблокирует обработку всех назначенных событий прерываний.

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

Операнды: нет

При переходе в режим RUN Вы блокируете прерывания. Если CPU находится в режиме RUN, то Вы можете с помощью операции ENI разблокировать все события прерываний. Команда “Блокировка всех событий прерываний” допускает постановку прерываний в очередь, но не разрешает вызывать программы обработки прерываний.

Назначение прерывания и отделение прерывания



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

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

Операнды: INT : от 0 до 127

EVENT: от 0 до 20

Описание операций назначения прерываний и отделения прерываний

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

Если Вы сопоставляете событие прерывания программе обработки прерываний, то это событие автоматически разблокируется. Если Вы выполняете операцию “Блокировка всех событий прерываний”, то все возникающие прерывания становятся в очередь до тех пор, пока Вы снова не отмените блокировку прерываний с помощью операции “Разблокировка всех событий прерываний”.

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

В табл. 9–13 приведены различные события прерываний.

Таблица 9–13. Описание событий прерываний

Номер события	Описание прерываний	CPU 212	CPU 214	CPU 215	CPU 216
0	Нарастающий фронт, E0.0*	Да	Да	Да	Да
1	Спадающий фронт, E0.0*	Да	Да	Да	Да
2	Нарастающий фронт, E0.1		Да	Да	Да
3	Спадающий фронт, E0.1		Да	Да	Да
4	Нарастающий фронт, E0.2		Да	Да	Да
5	Спадающий фронт, E0.2		Да	Да	Да
6	Нарастающий фронт, E0.3		Да	Да	Да
7	Спадающий фронт, E0.3		Да	Да	Да
8	Порт 0: Принят символ	Да	Да	Да	Да
9	Порт 0: Закончена передача	Да	Да	Да	Да
10	Управляемое временем прерывание 0	Да	Да	Да	Да
11	Управляемое временем прерывание 1		Да	Да	Да
12	HSC0: CV = PV (текущее значение = предварительно установленному значению)*	Да	Да	Да	Да
13	HSC1: CV = PV (текущее значение = предварительно установленному значению)*		Да	Да	Да
14	HSC1: Смена направления счета		Да	Да	Да
15	HSC1: Внешний сброс		Да	Да	Да
16	HSC2: CV = PV (текущее значение = предварительно установленному значению)		Да	Да	Да
17	HSC2: Смена направления счета		Да	Да	Да
18	HSC2: Внешний сброс		Да	Да	Да
19	PLS0: Отсчет импульсов завершен		Да	Да	Да
20	PLS1: Отсчет импульсов завершен		Да	Да	Да
21	Таймер T32: Прерывание CT = PT			Да	Да
22	Таймер T96: Прерывание CT = PT			Да	Да
23	Порт 0: Закончен прием сообщения			Да	Да
24	Порт 1: Закончен прием сообщения				Да
25	Порт 1: Принят символ				Да
26	Порт 1: Закончена передача				Да
* Если событие 12 (HSC0, PV = CV) поставлено в соответствие прерыванию, то события 0 и 1 не могут быть сопоставлены никакому прерыванию. Если одно из событий 0 и 1 поставлено в соответствие прерыванию, то событие 12 не может быть сопоставлено никакому прерыванию.					

Прерывания коммуникационных портов

Последовательным коммуникационным портом контроллера можно управлять с помощью программы на KOP или AWL. Коммуникация через такой порт называется свободно программируемой коммуникацией. В случае свободно программируемой коммуникации Ваша программа определяет скорость передачи данных, количество битов на символ, контроль четности и протокол. Прерывания передачи и приема облегчают коммуникацию с программным управлением. Подробную информацию по этому вопросу Вы найдете в разделе об операциях передачи и приема.

Прерывания от ввода/вывода

К прерываниям от ввода/вывода относятся прерывания при нарастающем или спадающем фронте, прерывания от быстрых счетчиков и прерывания от последовательности импульсов. CPU может создавать прерывание при нарастающем и/или спадающем фронте на входе. В табл. 9–14 приведены входы, доступные для прерываний в разных CPU. События “Нарастающий фронт” и “Спадающий фронт” могут восприниматься по каждому из этих входов. С помощью этих событий могут также отображаться сбойные ситуации, которые должны сразу приниматься во внимание при появлении события.

Таблица 9–14. Поддерживаемые прерывания по вводу/выводу

Прерывания от ввода/вывода	CPU 212	CPU 214	CPU 215	CPU 216
Входы и выходы	E0.0	от E0.0 до E0.3	от E0.0 до E0.3	от E0.0 до E0.3

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

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

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

Прерывания, управляемые временем

S7–200 может поддерживать одно или несколько прерываний, управляемых временем (см. табл. 9–15). С помощью прерываний, управляемых временем, Вы можете определять действия, которые должны выполняться периодически. Период задается с шагом 1 мс, значения лежат в диапазоне от 5 мс до 255 мс. Период для управляемого временем прерывания 0 запишите в SMB34, период для управляемого временем прерывания 1 запишите в SMB35.

Таблица 9–15. Поддерживаемые прерывания, управляемые временем

Прерывания, управляемые временем	CPU 212	CPU 214	CPU 215	CPU 216
Количество поддерживаемых прерываний, управляемых временем	1	2	2	2

Управляемое временем событие прерывания вызывает соответствующую программу обработки прерываний каждый раз, когда истекает время. В общем случае с помощью управляемых временем событий прерываний Вы управляете регулярным опросом аналоговых входов. Управляемое временем прерывание разблокируется и время начинает отсчитываться, когда Вы назначаете программу обработки прерываний управляемому временем событию прерывания. При этом система фиксирует период, чтобы последующие изменения не влияли на период. Если Вы хотите изменить период, то Вам нужно задать новое значение для периода и затем снова назначить программу обработки прерываний управляемому временем событию прерывания. При новом назначении эта функция стирает накопленное значение времени предыдущего назначения, и время начинает отсчитываться с новым значением периода.

После разблокировки прерывание, управляемое временем, функционирует непрерывно и обрабатывается каждый раз, когда истекает заданный интервал времени. Если Вы выходите из режима RUN или отделяете прерывание от программы обработки прерываний (DTCH), то управляемое временем прерывание блокируется. Если Вы выполняете операцию “Блокировка всех событий прерываний”, то управляемые временем прерывания в дальнейшем хотя и появляются, однако ставятся в очередь (до тех пор, пока либо прерывания снова не разблокируются, либо очередь не переполнится). На рис. 9–59 показан пример прерывания, управляемого временем.

Управляемые временем прерывания T32/T96 служат для управляемого временем реагирования на истечение заданного интервала времени. Эти прерывания поддерживаются только формирователями задержки включения (TON) с разрешающей способностью 1 мс - T32 и T96. В противном случае таймеры T32 и T96 имеют обычные функциональные возможности. Когда прерывание разблокировано, назначенная программа обработки прерываний выполняется, если при актуализации таймеров с разрешающей способностью 1 мс в цикле CPU окажется, что текущее значение активного таймера равно предварительно установленному значению таймера (см. раздел 9.4). Вы разблокируете эти прерывания, назначая программу обработки прерываний событию прерывания T32/T96.

Приоритеты прерываний и очереди

Приоритеты прерываний назначаются согласно следующей фиксированной схеме приоритетов:

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

Прерывания обрабатываются контроллером в пределах соответствующих им приоритетов в последовательности из появления. Всегда активна только одна программа обработки прерываний. Если в данный момент обрабатывается программа обработки прерываний, то эта программа доводится до конца. Она не может прерываться программой обработки прерываний, появляющейся позже, даже если приоритет этой программы выше. Прерывания, возникающие во время обработки другого прерывания, принимаются в очередь и обрабатываются позже. В табл. 9–16 показаны три очереди для прерываний и максимальное количество прерываний, которые могут приниматься в каждую очередь.

Таблица 9–16. Очереди для прерываний и максимальное количество записей на очередь.

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

В принципе может появиться больше прерываний, чем сможет принять очередь. Поэтому система имеет в своем распоряжении меркеры переполнения очередей, указывающие вид событий прерываний, которые не смогли быть приняты в очередь. Таблица 9–17 поясняет специальные меркеры, которые устанавливаются при переполнении очереди. Вы можете использовать эти биты 4.0, 4.1 и 4.2 только в одной программе обработки прерываний, так как они сбрасываются, когда очередь обработана и снова начинается обработка главной программы.

Таблица 9–17. Определения специальных меркеров переполнения очередей	
Описание (0 = нет переполнения, 1 = переполнение)	Специальный меркер
Переполнение очереди для коммуникационных прерываний	SM4.0
Переполнение очереди для прерываний от ввода/вывода	SM4.1
Переполнение очереди для прерываний, управляемых временем	SM4.2

В табл. 9–18 показаны событие прерывания, приоритет и назначенный номер прерывания.

Таблица 9–18. Описание событий прерываний

Номер события	Описание прерывания	Класс приоритета	Приоритет
8	Порт 0: Принят символ	Коммуникационные прерывания: высший класс приоритета	0
9	Порт 0: Закончена передача		0*
23	Порт 0: Закончен прием сообщения		0*
24	Порт 1: Закончен прием сообщения		1
25	Порт 1: Принят символ		1*
26	Порт 1: Закончена передача		1*
0	Нарастающий фронт, E0.0**		Прерывания от ввода/вывода: средний класс приоритета
2	Нарастающий фронт, E0.1	1	
4	Нарастающий фронт, E0.2	2	
6	Нарастающий фронт, E0.3	3	
1	Спадающий фронт, E0.0**	4	
3	Спадающий фронт, E0.1	5	
5	Спадающий фронт, E0.2	6	
7	Спадающий фронт, E0.3	7	
12	HSC0: CV = PV (текущее значение = предварительно установленному значению)**	0	
13	HSC1: CV = PV (текущее значение = предварительно установленному значению)	8	
14	HSC1: Смена направления счета	9	
15	HSC1: Внешний сброс	10	
16	HSC2: CV = PV (текущее значение = предварительно установленному значению)	11	
17	HSC2: Смена направления счета	12	
18	HSC2: Внешний сброс	13	
19	PLS0: Счет импульсов завершен	14	
20	PLS1: Счет импульсов завершен	15	
10	Управляемое временем прерывание 0	Управляемые временем прерывания: низший класс приоритета	0
11	Управляемое временем прерывание 1		1
21	Таймер T32: Прерывание CT = PT		2
22	Таймер T96: Прерывание CT = PT		3

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

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

Пример прерываний

На рис. 9–56 показан пример операций в программе обработки прерываний.

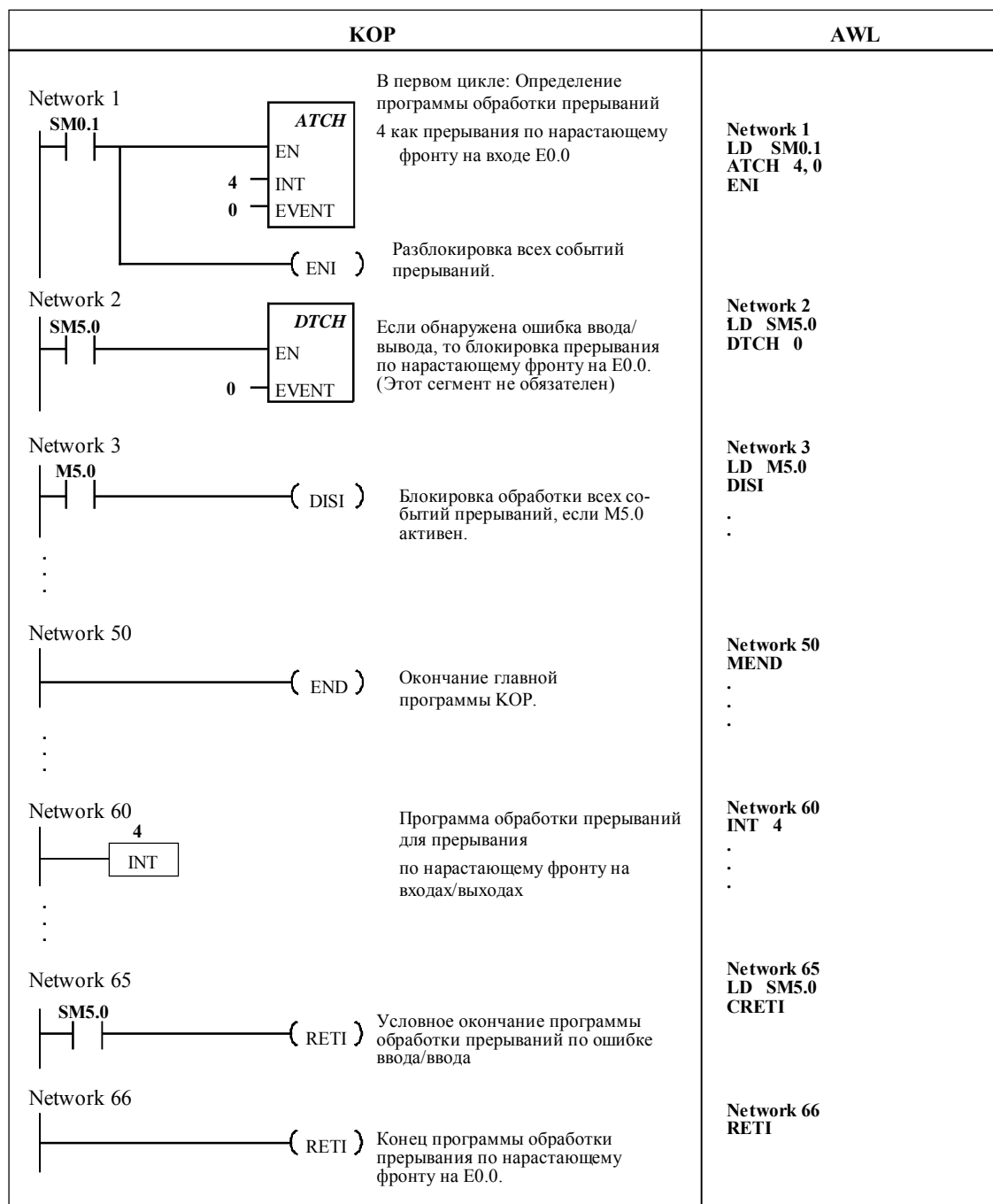


Рис. 9-56. Пример операций в программе обработки прерываний в форме KOP и AWL

На рис. 9-57 показано, как можно считывать значение аналогового входа с помощью прерывания, управляемого временем.

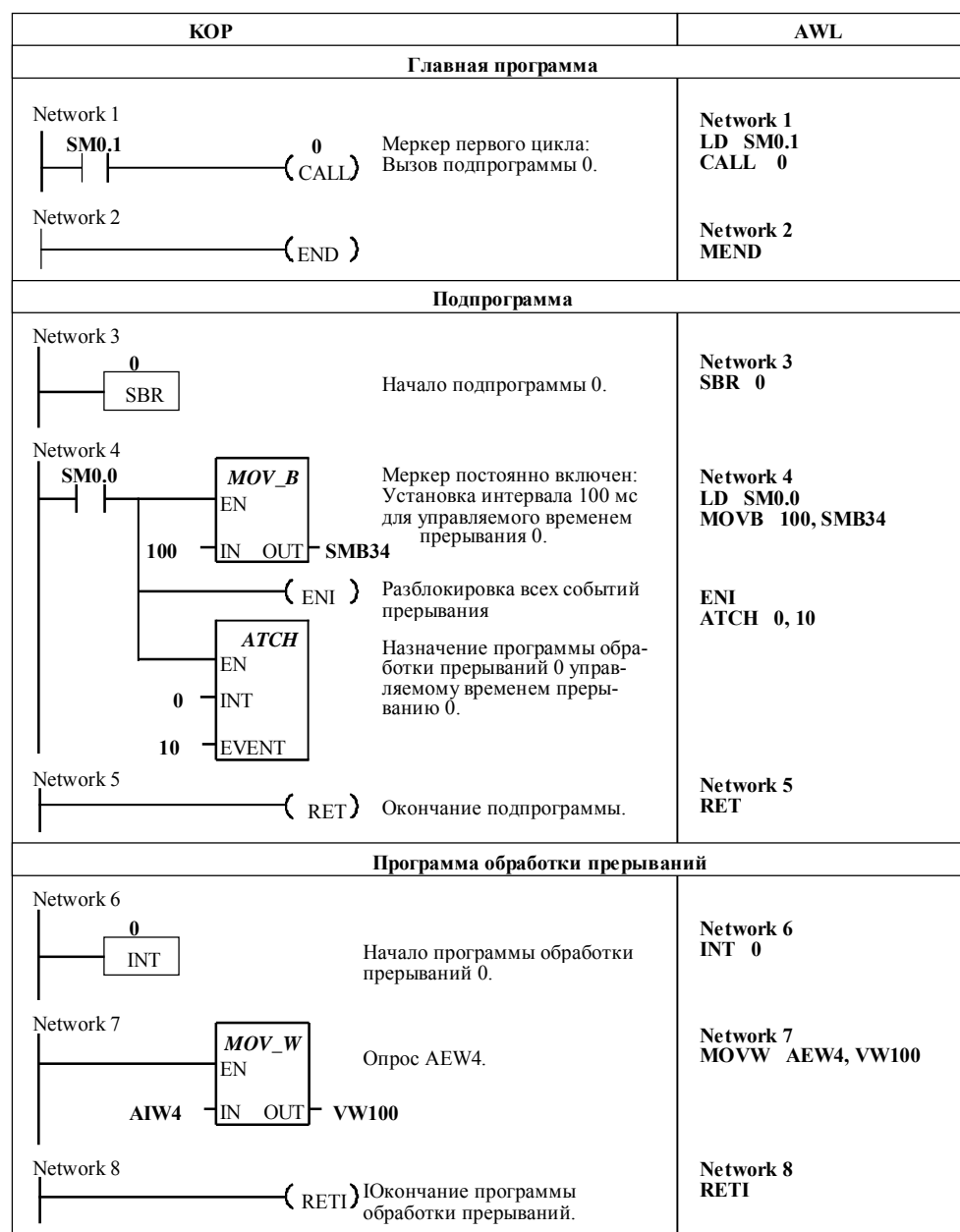
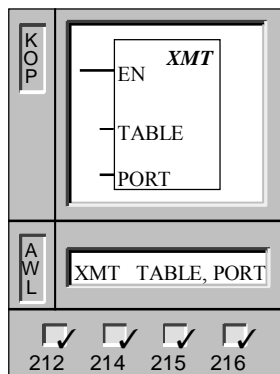


Рис. 9-57. Пример считывания аналогового входа с помощью прерывания, управляемого временем

9.16 Коммуникационные операции

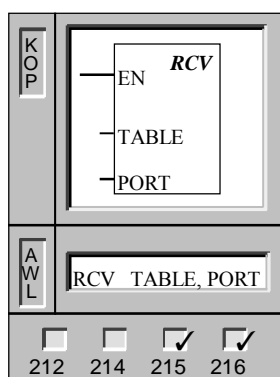
Передача сообщения из буферной памяти и прием сообщения в буферную память



Операция **Передача сообщения из буферной памяти** вызывает передачу из буферной памяти (TABLE). Первая запись в буфере указывает, сколько байтов должны передаваться. PORT задает коммуникационный порт, который должен использоваться для передачи.

Операнды: TABLE: VB, EB, AB, MB, SMB,
*VD, *AC, SB
PORT: 0 - 1

Операция XMT используется в случае свободно программируемой коммуникации для передачи данных через коммуникационный порт(ы).



Операция **Прием сообщения из буферной памяти** вызывает изменения в Setup, которые инициируют или завершают прием сообщения. Сообщения, принимаемые через заданный порт (PORT), записываются в буферную память (TABLE). Первая запись в буфере указывает количество принятых байтов.

Операнды: TABLE: VB, EB, AB, MB, SMB, SB,
*VD, *AC
PORT: 0 - 1

Операция RCV используется в случае свободно программируемой коммуникации для приема данных через коммуникационный порт(ы).

Описание свободно программируемой коммуникации

Последовательным коммуникационным портом CPU можно управлять через программу пользователя. Такой режим работы называется свободно программируемой коммуникацией. Если Вы выбираете режим свободно программируемой коммуникации, то KOP-программа управляет работой коммуникационного порта посредством прерываний приема, прерываний передачи, операции передачи (XMT) и операции приема (RCV). Во время свободно программируемой коммуникации коммуникационный протокол полностью управляется KOP-программой. SMB30 (для порта 0) и SMB130 (для порта 1, если Ваш CPU имеет в своем распоряжении два порта) служат для выбора скорости передачи данных и контроля четности.

Если CPU находится в режиме STOP, то свободно программируемая коммуникация блокируется и восстанавливается нормальная коммуникация (например, доступ через устройство программирования).

В простейшем случае Вы можете посылать сообщение в принтер или устройство индикации и использовать для этого только операцию XMT. Однако Вы можете передавать данные также на считыватель штрихового кода, весы или сварочный аппарат. В любом случае Вы должны написать Вашу программу таким образом, чтобы она поддерживала протокол устройства, с которым CPU желает обмениваться сообщениями в режиме свободно программируемой коммуникации.

Свободно программируемая коммуникация возможна только тогда, когда CPU находится в режиме RUN. Вы разблокируете режим свободно программируемой коммуникации, устанавливая значение "01" в поле выбора протокола в SMB30 (порт 0) или в SMB130 (порт 1). В режиме свободно программируемой коммуникации Вы не можете обмениваться сообщениями с устройством программирования.

Указание

Активизацией режима свободно программируемой коммуникации можно управлять с помощью специального меркера SM0.7. Этот специальный меркер представляет текущее положение переключателя режимов работы. Если SM0.7 = 0, то этот переключатель находится в положении TERM. Если SM0.7 = 1, то переключатель находится в положении RUN. Если Вы разблокируете режим свободно программируемой коммуникации только тогда, когда переключатель находится в положении RUN, то Вы можете контролировать работу CPU или управлять его работой с помощью устройства программирования, переводя переключатель в другое положение.

Инициализация свободно программируемой коммуникации

SMB30 и SMB130 конфигурируют для свободно программируемой коммуникации коммуникационные порты 0 и 1 соответственно. В этих специальных меркерах Вы устанавливаете скорость передачи данных, контроль четности и количество битов данных. Эти управляющие байты описаны в таблице 9–19.

Таблица 9–19. Специальные меркеры SMB30 и SMB130

Порт 0	Порт 1	Описание								
Формат SMB30	Формат SMB130	Управляющий байт для свободно программируемой коммуникации <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>π</td> <td>π</td> <td>δ</td> <td>β</td> <td>β</td> <td>β</td> <td>μ</td> <td>μ</td> </tr> </table>	π	π	δ	β	β	β	μ	μ
π	π	δ	β	β	β	μ	μ			
SM30.6 и SM30.7	SM130.6 и SM130.7	pp Выбор паритета 00 = нет паритета 01 = четный паритет 10 = нет паритета 11 = нечетный паритет								
SM30.5	SM130.5	d Количество битов данных на символ 0 = 8 битов на символ 1 = 7 битов на символ								
SM30.2 - SM30.4	SM130.2 - SM130.4	bbb Скорость передачи данных 000 = 38.400 Бод (в CPU 212: 19.200 Бод) 001 = 19.200 Бод 010 = 9.600 Бод 011 = 4.800 Бод 100 = 2.400 Бод 101 = 1.200 Бод 110 = 600 Бод 111 = 300 Бод								
SM30.0 и SM30.1	SM130.0 и SM130.1	mm Выбор протокола 00 = Протокол интерфейса "точка-к-точке" (PPI) или системный протокол 01 = Протокол свободно программируемой коммуникации 10 = PPI дает возможность использования операций NETR и NETW 11 = резервный (по умолчанию PPI/режим slave)								

Передача данных с помощью операции XMT

Операция XMT облегчает передачу данных. С помощью операции XMT Вы можете передать буфер с максимальной длиной 255 символов. После передачи последнего символа буфера генерируется прерывание (событие прерывания 9 для порта 0 и событие прерывания 26 для порта 1), если событию “Закончена передача” назначена программа обработки прерываний. Вы можете передавать данные также без прерывания (например, при передаче сообщения на принтер), контролируя SM4.5 на предмет окончания передачи.

Указание

Кабель PC/PPI содержит преобразователь RS-232 в RS-485. Если этот кабель используется для свободно программируемой коммуникации, то Вы должны обеспечить интервал в 2 символа между приемом и передачей сообщений.

Прием данных с помощью операции RCV

Операция RCV облегчает прием данных. С помощью операции RCV Вы можете принимать буфер максимальной длиной 255 символов. После поступления в буфер последнего символа генерируется прерывание (событие прерывания 9 для порта 0 и событие прерывания 26 для порта 1), если событию “Закончен прием” назначена программа обработки прерываний. Вы можете принимать данные также без прерывания, контролируя SM86.

С помощью операции RCV Вы можете выбирать условия начала и конца сообщения. Условия начала и конца сообщения описаны в таблице на рис. 9-58 (SM86 - SM94 для порта 0 и SM186 - SB194 для порта 1).

Порт 0	Порт 1	Описание
SMB86	SMB186	<div style="display: flex; justify-content: space-between; align-items: center;"> MSB 7 LSB 0 </div> <div style="display: flex; align-items: center; margin: 5px 0;"> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">n</div> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">0</div> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">e</div> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">0</div> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">0</div> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">t</div> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">c</div> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">p</div> </div> <p>Байт состояния приема сообщений</p> <p>n: 1 = прием сообщений блокирован пользователем e: 1 = принят символ конца t: 1 = прием сообщения закончен: истекло время c: 1 = прием сообщения закончен: достигнуто максим. число символов p: 1 = прием сообщения закончен: ошибка четности</p>
SMB87	SMB187	<div style="display: flex; justify-content: space-between; align-items: center;"> MSB 7 LSB 0 </div> <div style="display: flex; align-items: center; margin: 5px 0;"> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">n</div> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">x</div> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">y</div> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">z</div> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">m</div> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">t</div> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">0</div> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">0</div> </div> <p>Управляющий байт приема сообщений</p> <p>n: 0 = функция приема сообщений блокирована 1 = функция приема сообщений разблокирована Бит блокировки/разблокировки приема проверяется каждый раз, когда выполняется операция RCV. x: 0 = SMB88 или SMB188 не учитыв. 1 = с помощью SMB88 или SMB188 распознавать начало сообщения y: 0 = SMB89 или SMB189 игнорировать 1 = с помощью SMB89 или SMB189 распознавать конец сообщения z: 0 = SMW90 или SMB190 игнорировать 1 = с помощью SMB90 или SMB190 распознавать пустую строку m: 0 = датчик тактов отмеряет межсимвольный интервал 1 = датчик тактов отмеряет время сообщения t: 0 = SMW92 или SMB192 игнорировать 1 = закончить прием, если превышено время из SMW92 или SMW192. Эти биты определяют критерии идентификации сообщений (включая критерии начала и конца сообщения). Для распознавания начала сообщения разблокированные критерии начала сообщения логически сопрягаются через И и должны появляться последовательно (со стартовым символом после строки холостого хода). Для распознавания конца сообщения разблокированные критерии конца сообщения логически сопрягаются через ИЛИ. Уравнения критериев для начала и конца сообщения: начало сообщения = z * x конец сообщения = y + t + достижение максимального числа символов. Разблокированная функция приема сообщений заканчивается сразу автоматически при переполнении или ошибке четности</p>
SMB88	SMB188	Символ начала сообщения
SMB89	SMB189	Символ конца сообщения
SMB90 SMB91	SMB190 SMB191	Длительность пустой строки в мс. Первый символ, принимаемый по истечении времени пустой строки, отмечает начало нового сообщения. SM90 (или SM190) является старшим байтом, а SM91 (или SM191) является младшим байтом.
SMB92 SMB93	SMB192 SMB193	Значение для контроля времени в мс при измерении времени между символами и времени между сообщениями. Если это время превышено, то прием сообщений заканчивается. SM92 (или SM192) является старшим байтом, а SM93 (или SM193) является младшим байтом.
SMB94	SMB194	Максимальное число символов, которое может быть принято (1 - 255 байтов). Указание: Этот диапазон регулируется максимально ожидаемой длиной буфера, даже если не используется прерывание приема посредством функции подсчета символов.

Рис. 9-58. Специальные меркеры SM86 - SMB94 и SMB186 - SM194

Прием данных с помощью посимвольных прерываний

Для того, чтобы располагать большей гибкостью в поддерживаемых протоколах, Вы можете также принимать данные с управлением от прерываний. При этом каждый принятый символ порождает прерывание. Принятый символ записывается в SMB2, а состояние паритета (если активизировано) записывается в SM3.0. Это происходит непосредственно перед выполнением программы обработки прерываний, назначенной событию "Принят символ".

- SMB2 служит в качестве буфера для принимаемых символов в случае свободно программируемой коммуникации. Символы, принимаемые во время свободно программируемой коммуникации, записываются в этот буфер, чтобы программа пользователя могла быстро производить доступ к этим значениям.
- SMB3 используется в случае свободно программируемой коммуникации и содержит бит, который устанавливается, если в принимаемом символе обнаруживается ошибка паритета. Все другие биты этого байта являются резервными. С помощью данного бита Вы можете отбросить сообщение или создать отрицательное подтверждение.

Пример операции передачи сообщения из буферной памяти

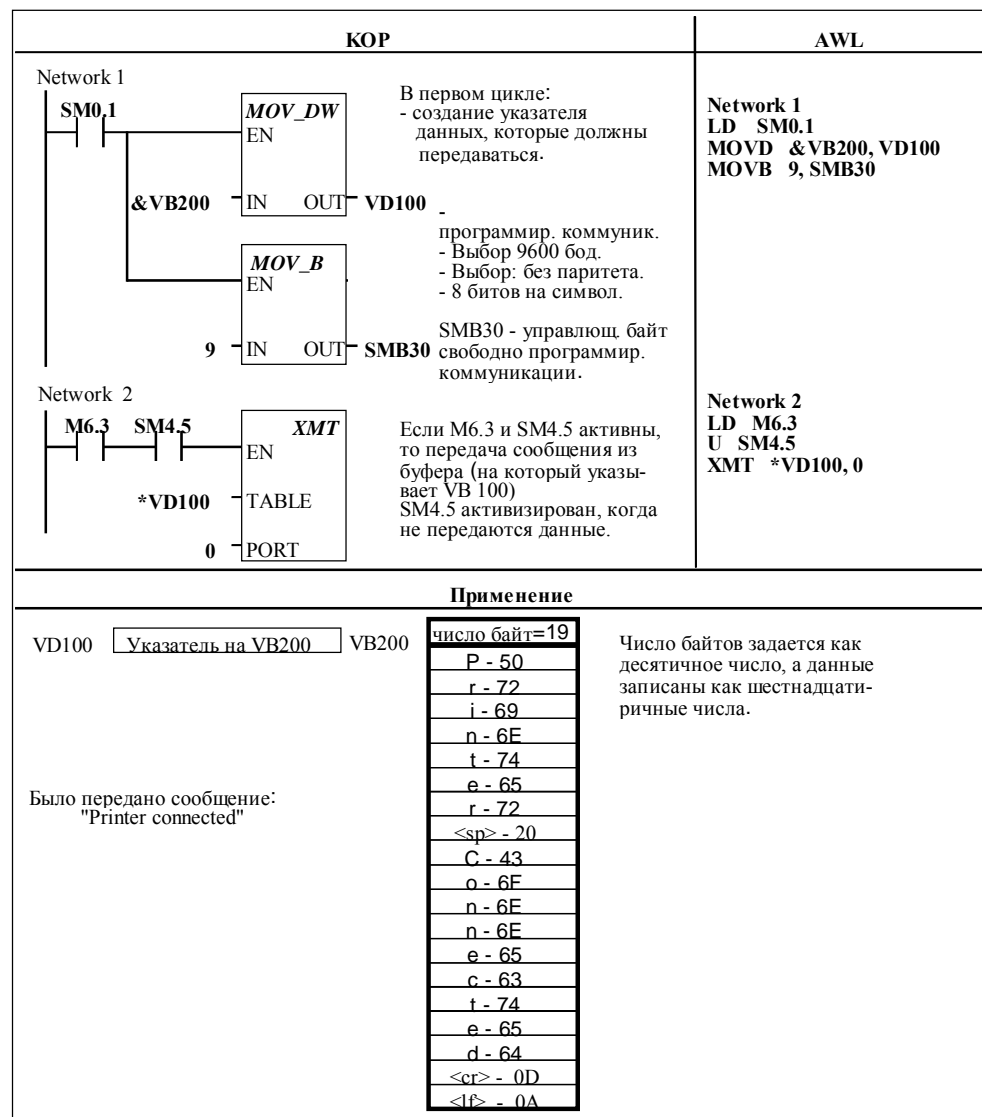
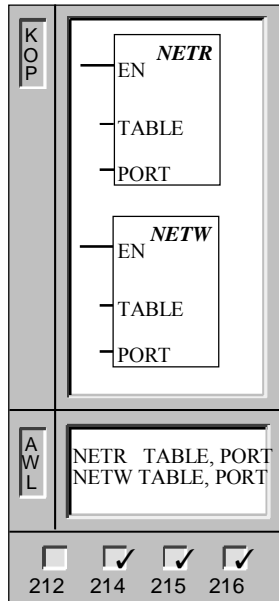


Рис. 9-59. Операция передачи сообщения из буфера в форме KOP и AWL

Чтение из сети и запись в сеть



Команда **Чтение из сети** запускает коммуникационную операцию, которая в соответствии с определением в таблице (TABLE) считывает данные из удаленного устройства через порт (PORT).

Команда **Запись в сеть** запускает коммуникационную операцию, которая в соответствии с определением в таблице (TABLE) записывает данные в удаленное устройство через порт (PORT).

Операнды: TABLE: VB, MB, *VD, *AC

PORT: константы 0 - 1

С помощью операции NETR Вы можете считывать максимум 16 байтов информации из удаленной станции. С помощью операции NETW Вы можете записать максимум 16 байтов информации в удаленную станцию. В S7-200 могут быть активизированы одновременно восемь операций NETR и NETW, например четыре операции NETR и четыре операции NETW или две операции NETR и шесть операций NETW.

Рисунок 9-60 определяет таблицу, на которую ссылается параметр TABLE в командах NETR и NETW.

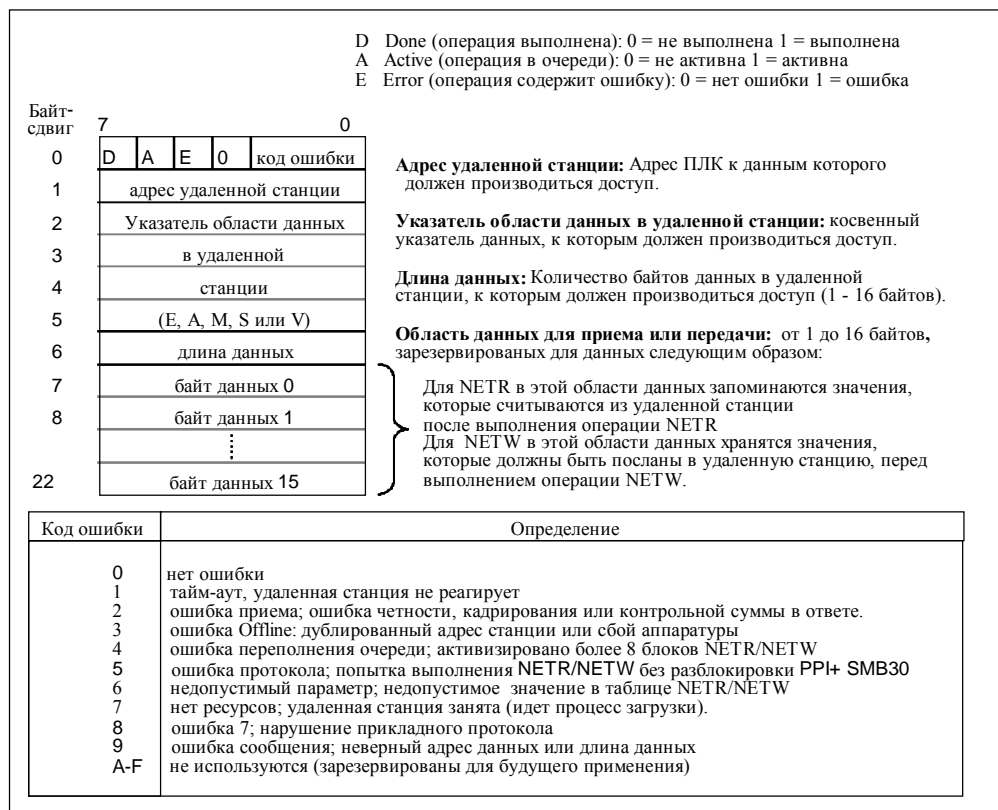


Рис. 9-60. Определение TABLE для операций NETR и NETW

Примеры операций NETR и NETW

На рис. 9–61 показан пример использования операций NETR и NETW. В этом примере речь идет о производственной линии, на которой сосуды наполняются маслом и передаются дальше на одну из четырех упаковочных машин. Упаковочная машина упаковывает по восемь сосудов с маслом в картонную коробку. Направляющее устройство управляет тем, к какой упаковочной машине направить отдельный сосуд с маслом. Четыре CPU 212 управляют четырьмя упаковочными машинами. Один CPU 214, оснащенный интерфейсом оператора TD 200, управляет направляющим устройством. На рис. 9–61 показана конструкция сети.

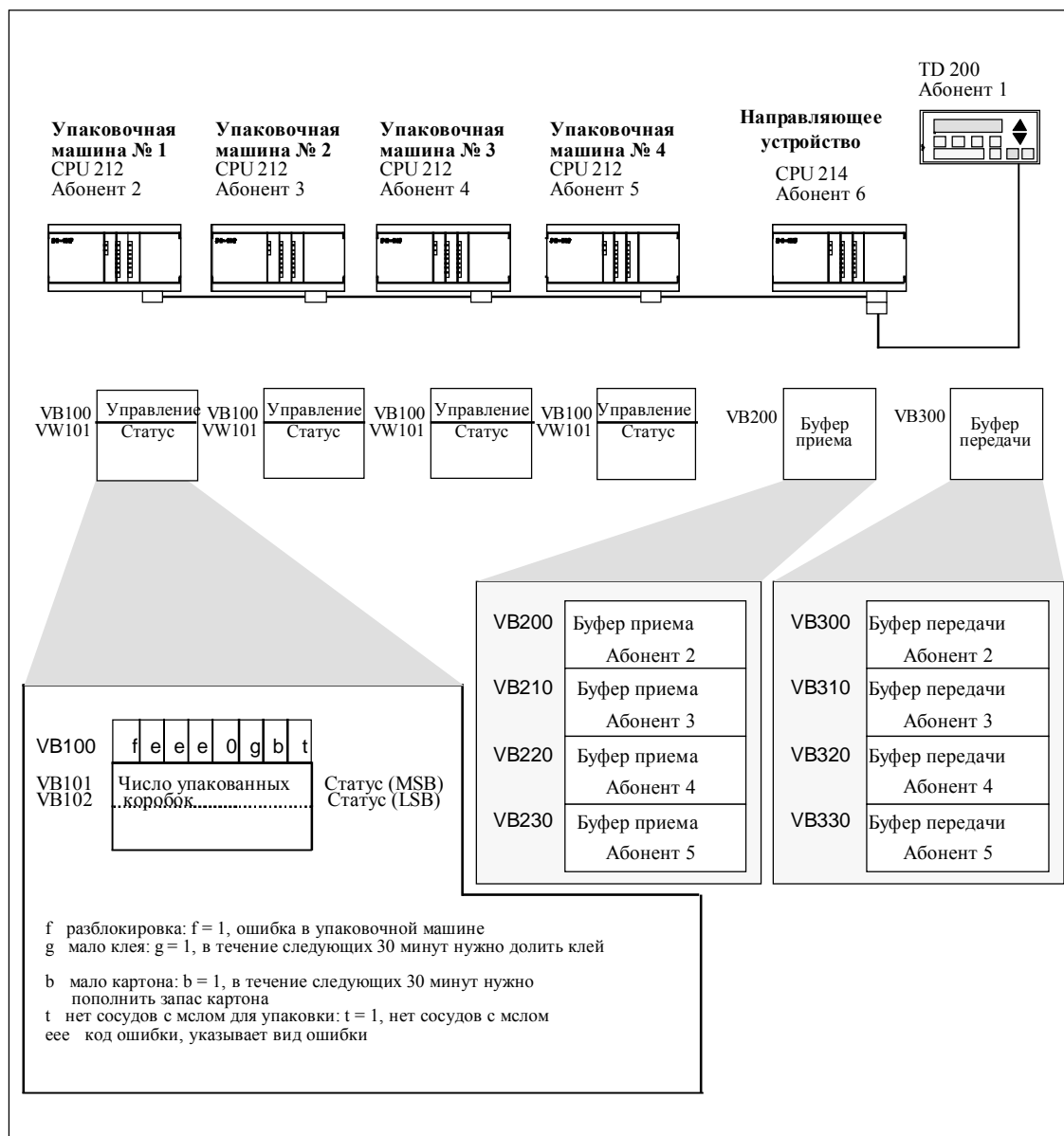


Рис. 9-61. Пример операций NETR и NETW

На рис. 9–62 подробно представлены буфера приема и передачи, которые служат для доступа к данным абонента 2 (эти буфера находятся в VB200 и VB300 соответственно).

CPU 214 с помощью операции NETR регулярно считывает управляющую и статусную информацию из отдельных упаковочных машин. Каждый раз, когда упаковочная машина упаковала 100 коробок, направляющее устройство с помощью операции NETW передает сообщение для того, чтобы сбросить слово статуса.

На рис. 9–63 показана программа, с помощью которой считывается управляющий байт, подсчитываются упакованные коробки и сбрасывается количество упакованных коробок отдельно для каждой упаковочной машины (в данном случае упаковочная машина № 1).

Буфер приема направляющего устр-ва для считывания с упаковочной машины №1					Буфер передачи направляющего устр-ва для сброса счетчика упаков. машины №1.						
	7			0		7			0		
VB200	D	A	E	0	код ошибки	VB300	D	A	E	0	код ошибки
VB201	адрес удаленной станции					VB301	адрес удаленной станции				
VB202	указатель					VB302	указатель				
VB203	области данных					VB303	области данных				
VB204	в удаленной					VB304	в удаленной				
VB205	станции = (&VB100)					VB305	станции = (&VB101)				
VB206	длина данных = 3 байта					VB306	длина данных = 2 байта				
VB207	управление					VB307	0				
VB208	статус (MSB)					VB308	0				
VB209	статус (LSB)										

Рис. 9-62. Определение TABLE для операций NETR и NETW

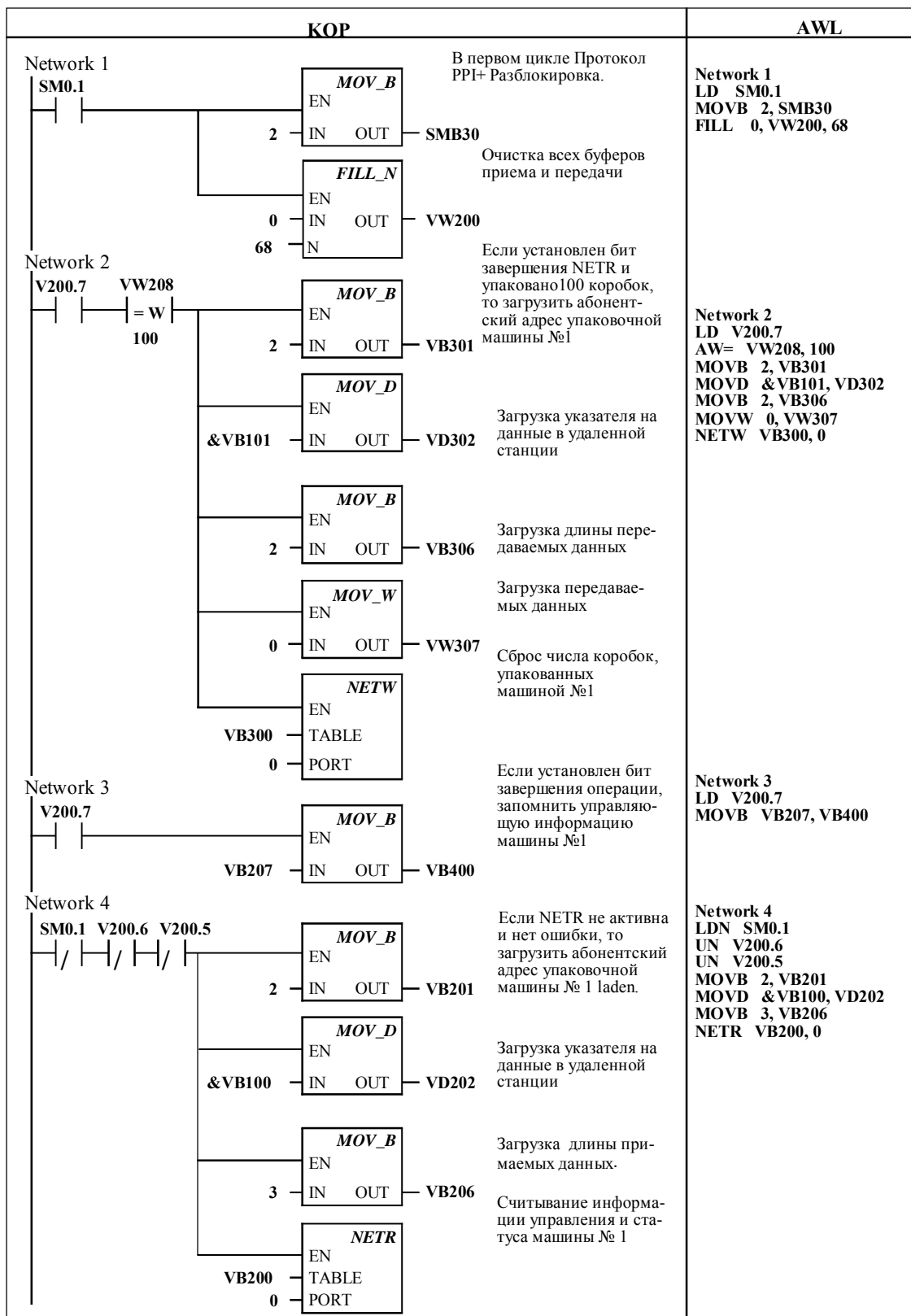
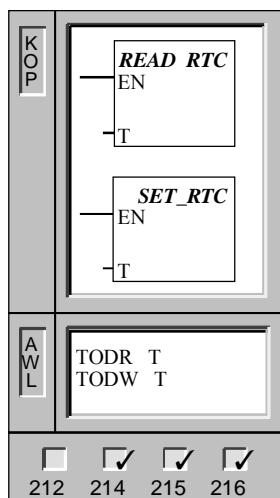


Рис. 9-63. Пример операций NETR и NETW

9.17 Операции с часами реального времени

Чтение часов реального времени и запись в часы реального времени



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

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

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

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

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

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

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

Вы должны кодировать дату и время в BCD-формате (например,)16#97 для года 1997). Используйте для этого следующие форматы данных:

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

Указание

S7-200 CPU не проверяет согласованность дня недели с датой. Вследствие этого могут приниматься недопустимые даты, такие как 30 февраля. Поэтому Вам следует всегда убедиться в том, что Вы правильно ввели дату.

Никогда не используйте операции TODR и TODW в главной программе и в программе обработки прерываний одновременно. В противном случае операция TOD больше не сможет выполняться в программе обработки прерываний при возникновении прерывания, если она уже была выполнена в главной программе. SM4.5 установится, указывая, что две операции попытались одновременно получить доступ к часам.

