Набор команд

Содержание **Стр.**

Контакты 2
Катушки 3
Команды битовой логики 5
Блоковые команды 6
Команды управления битами 11
Счетчики 13
Таймеры 14
Команды управления программой 17
Команды управления программой. Подпрограммы 21
Команды сдвига данных 22
Команды перемещения данных 27
Команды сравнения 36
Команды преобразования данных 42
Команды BCD арифметики 51
Команды двоичной арифметики 55
Специальные математические команды 58
Команды реального времени 61
Логические команды 63
Команды флагов и регистров 67
Команды стандартной коммуникации 69
Команды сетевых коммуникаций 71
Команды управления прерываниями 74
Команды управления входами/выходами 76
Команды диагностики и ошибок 78
Системные команды 81
Дополнительные команды 84

Контакты

Процесс			Интерпретация в программе PLC				
Тип датчика	Состояние датчика	Напряжение на входе	Состояние сигнала на входе	Опрос для состояния «1»		Опрос для состояния «0»	
				Символ	Результат	Символ	Результат
Норм. откр.	активирован	Есть	1	**Ladder** 	Цепочка замкнута	**Ladder** 	Цепочка разомкнута
	не активир.	Нет	0	**FP** 	Цепочка разомкнута	**FP** 	Цепочка замкнута
Норм. замкн.	активирован	Нет	0		Цепочка разомкнута		Цепочка замкнута
	не активир	Есть	1	**STL** LD x.y	Цепочка замкнута	**STL** LD NOT x.y	Цепочка разомкнута

Использование нормально – открытого или нормально- закрытого контактов для датчиков – сенсоров в автоматической системе зависит от требований безопасности.
Нормально разомкнутые контакты всегда используются для блокировок и выключателей безопасности, чтобы в случае обрыва проводов в цепи, соединяющей датчики, не возникли опасные условия.
Нормально замкнутые контакты по той же причине используются для выключения оборудования.

При составлении программы нет необходимости руководствоваться тем, поступает в действительности сигнал «1» от нормально открытого или нормально замкнутого контакта. Необходимо руководствоваться следующим правилом:

если выход вычислительной цепочки должен устанавливаться в «1» при единичном значении переменной, то эта переменная должна быть представлена нормально разомкнутым контактом, и наоборот, если выход должен устанавливаться при нулевом значении переменной, то эта переменная должна быть представлена нормально – замкнутым контактом.

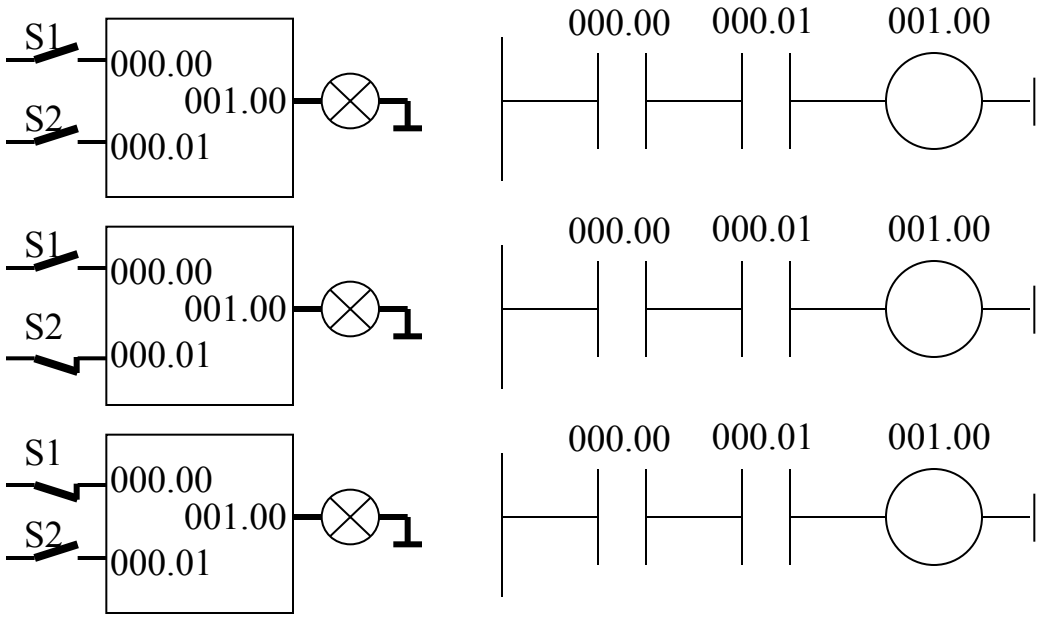
Катушки

Командой Open Output битовому операнду будет присваиваться значение «1» - при условии замкнутой цепочки, и значение «0» - при условии разомкнутой цепочки.

Командой Closed Output битовому операнду будет присваиваться значение «0» - при условии замкнутой цепочки, и значение «1» - при условии разомкнутой цепочки.

Самый простой способ выдать результат комбинации условия исполнения – прямая выдача командами Open Output и Closed Output. Данные команды используются для управления состоянием битового операнда в соответствии с условием замкнутой или разорванной вычислительной цепочкой.

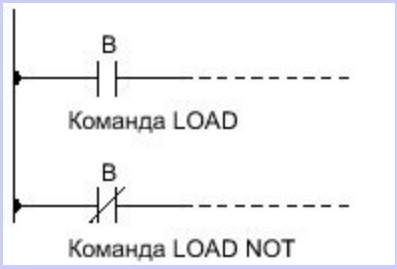
Упражнение



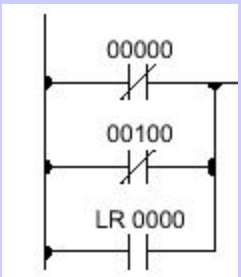
Завершите программы приведенные на рисунке, чтобы выполнить следующее задание: Когда ключ S1 активируется, а ключ S2 не активируется, свет должен гореть во всех трех случаях.

Команды битовой логики

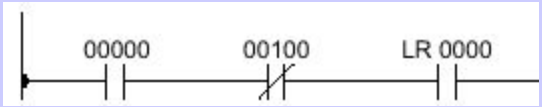
LOAD и LOAD NOT



OR и OR NOT



AND и AND NOT



Первыми условиями, которые начинают любую вычислительную цепочку, являются команды Load или Load Not.

Для реализации функции логического умножения необходимо контакты расположить последовательно друг за другом. Каждая команда **AND** выполняет ЛОГИЧЕСКОЕ И над своим условием исполнения (т.е результатом всех условий до данной точки) и состоянием битового операнда самой команды. Если оба этих условия =1, то условие исполнения для следующей команды будет =1. Если хотя бы одно из из этих условий =0, то условие исполнения следующей команды будет =0.

Каждая команда **AND NOT** выполняет ЛОГИЧЕСКОЕ И над своим условием исполнения (т.е результатом всех условий до данной точки) и инверсией битового операнда самой команды.

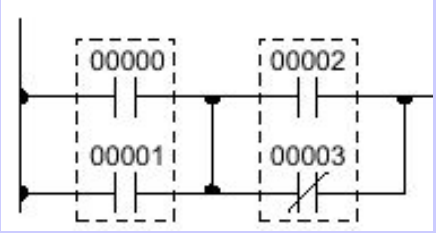
Для реализации функции логического сложения необходимо контакты расположить параллельно друг другу. Каждая команда **OR** выполняет ЛОГИЧЕСКОЕ ИЛИ над своим условием исполнения (т.е результатом всех условий до данной точки) и состоянием битового операнда самой команды. Если хотя бы одно из из этих условий =1, то условие исполнения следующей команды будет =1.

Каждая команда **OR NOT** выполняет ЛОГИЧЕСКОЕ ИЛИ над своим условием исполнения (т.е результатом всех условий до данной точки) и инверсией битового операнда самой команды.

При совместном использовании обеих команд приоритет исполнения имеет команда OR или OR NOT.

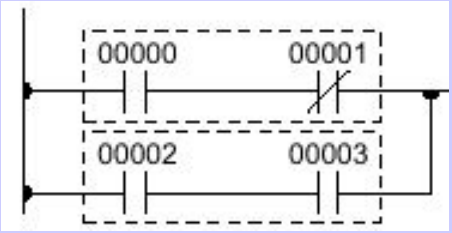
Блоковые команды

AND LOAD



LD	00000
OR	00001
LD	00002
OR NOT	00003
AND LD	-

OR LOAD



LD	00000
AND NOT	00001
LD	00002
AND	00003
OR LD	

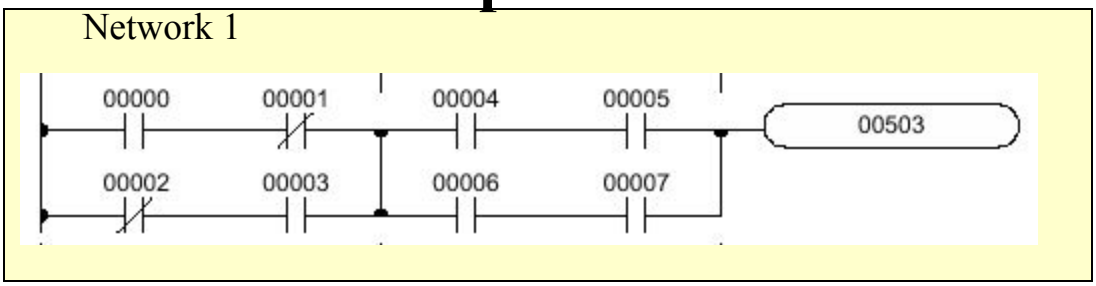
Блоковые команды не опрашивают конкретные условия РКС, а описывают отношение между модулями или блоками вычислительной цепочки. Блоком называются несколько, последовательно выполняемых друг за другом, однотипных команд. Тогда, при составлении программы, начальный операнд блока определяется командой LOAD или LOAD NOT, а после описания всех команд блока начинается программирование следующего блока. Сами блоки объединяются командой AND LOAD – если блоки расположены последовательно, и командой OR LOAD – при параллельном расположении блоков.

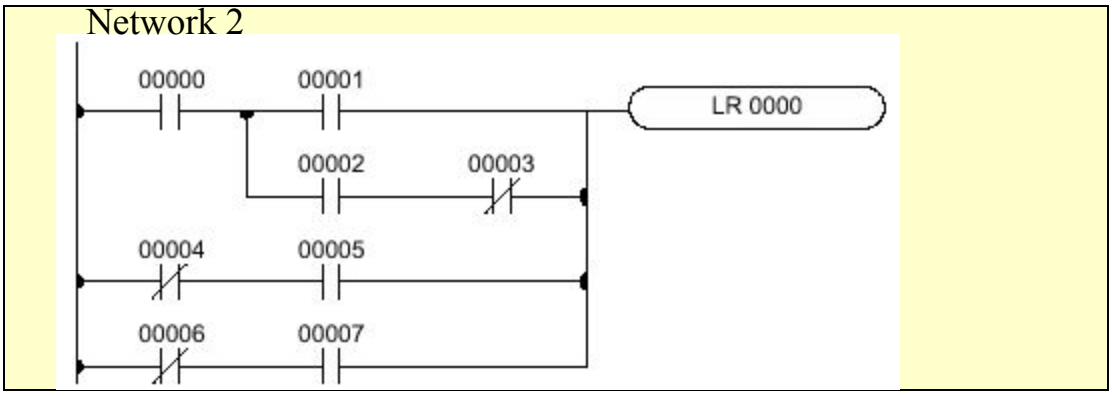
Существует два способа объединения блоков:

1. Описать все блоки вычислительной цепочки, а затем командами AND LOAD и OR LOAD последовательно объединить их. При этом число команд объединения будет на 1 меньше, чем число блоков, но при этом, общее количество блоков не должно превышать 8.

1. Описать два блока, провести объединение. Описать следующий – объединить, и т.д. Этим способом можно объединять неограниченное количество блоков.

Упражнение





На рисунке приведены две вычислительные цепочки. Необходимо составить программы на STL, а затем проверить правильность с помощью программного обеспечения.

Network 1

Network 2

Программа на STL для Network 1

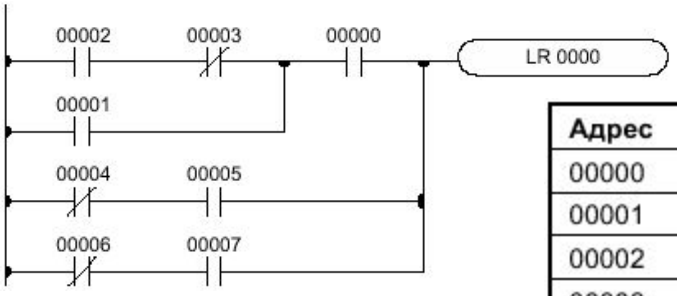
Адрес	Инструкция	Операнд
00000	LD	00000
00001	OR NOT	00001
00002	LD NOT	00002
00003	OR	00003
00004	AND LD	-
00005	LD	00004
00006	OR	00005
00007	AND LD	-
00008	OUT	00500

Адрес	Инструкция	Операнд
00000	LD	00000
00001	OR NOT	00001
00002	LD NOT	00002
00003	OR	00003
00004	LD	00004
00005	OR	00005
00006	AND LD	-
00007	AND LD	-
00008	OUT	00500

Программа на STL для Network 2

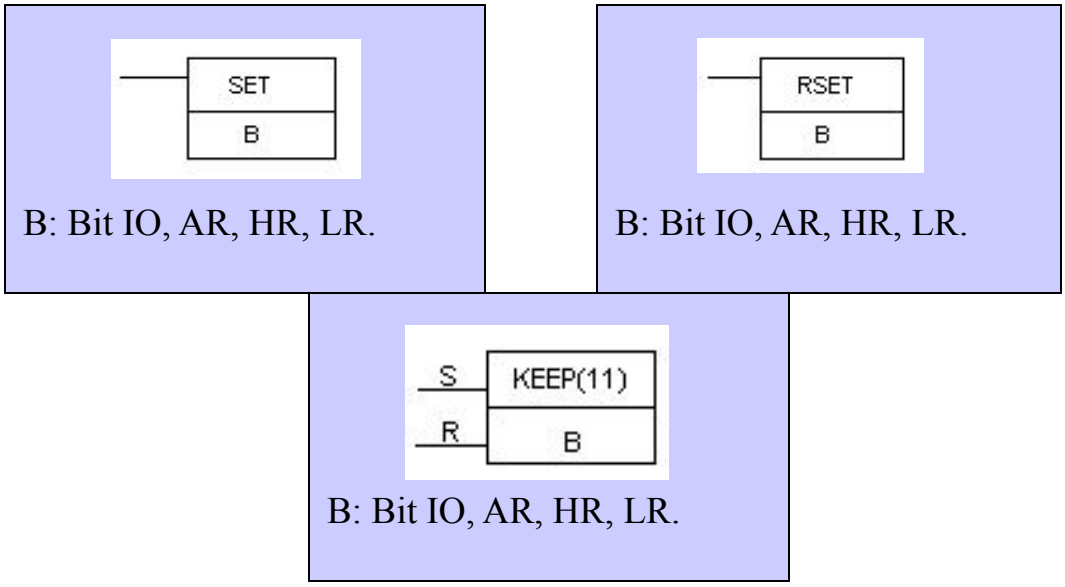
Адрес	Инструкция	Операнд
00000	LD	00000
00001	LD	00001
00002	LD	00002
00003	AND NOT	00003
00004	OR LD	-
00005	AND LD	-
00006	LD NOT	00004
00007	AND	00005
00008	OR LD	-
00009	LD NOT	00006
00010	AND	00007
00011	OR LD	-
0012	OUT	LR 0000

Оптимизация Network 2



Адрес	Инструкция	Операнд
00000	LD	00002
00001	AND NOT	00003
00002	OR	00001
00003	AND	00000
00004	LD NOT	00004
00005	AND	00005
00006	OR LD	-
00007	LD NOT	00006
00008	AND	00007
00009	OR LD	-
00010	OUT	LR 0000

Команды управления битами: установка, сохранение и сброс



SET включает битовый операнд в 1, когда условие исполнения =1 и не влияет на состояние операнда, когда условие исполнения = 0.

RSET включает битовый операнд в 0, когда условие исполнения =1 и не влияет на состояние операнда, когда условие исполнения = 0.

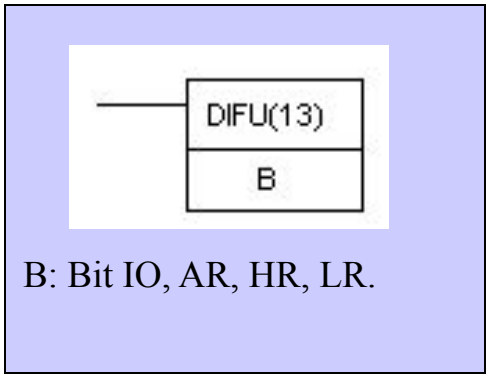
Операция SET отличается от OUT, поскольку OUT устанавливает битовый операнд в 0, когда условие исполнения =0. Точно так же RSET отличается от OUT NOT, тем, что OUT NOT устанавливает битовый операнд в 1, когда условие исполнения =0.

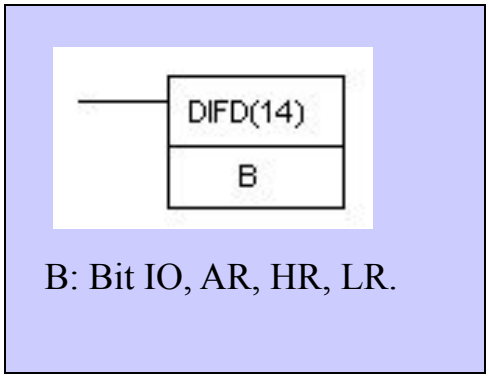
KEEP используется для поддержания состояния заданного бита, и работает как триггер исходя из двух условий – S и R. S- вход установки, R- вход сброса.

Когда S=1, указанный бит устанавливается в 1, и остается в этом состоянии до появление 1 на входе R, вне зависимости от состояния входа S.
Когда R=1, указанный бит устанавливается в 0, и остается в этом состоянии до появление 1 на входе S, но при этом на входе R должно соблюдаться условие 0.

KEEP имеет приоритет по входу R.

Команды управления битами: дифференцирование





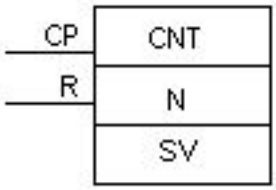
DIFU и DIFD используются для установки указанного бита только на 1 цикл.

При выполнении **DIFU** сравнивается текущее условия исполнения с условием исполнения прошлого цикла. Если условие исполнения в прошлом цикле было =0, а текущее = 1, то DIFU устанавливает в 1 указанный бит. Если условие исполнения в предыдущем цикле было =1, то независимо от текущего состояния DIFU устанавливает указанный бит в 0.

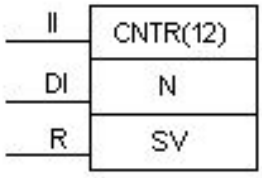
При выполнении **DIFD** сравнивается текущее условия исполнения с условием исполнения прошлого цикла. Если условие исполнения в прошлом цикле было =1, а текущее = 0, то DIFD устанавливает в 1 указанный бит. Если условие исполнения в предыдущем цикле было =0, то независимо от текущего состояния DIFU устанавливает указанный бит в 0.

Данные команды используются в случае, если у команды нет версии 0/1 (т. е однократное срабатывание по переднему фронту условия, графически обозначаются @) , а желательно исполнение отдельной команды в течение одного цикла, а так же в других случаях.

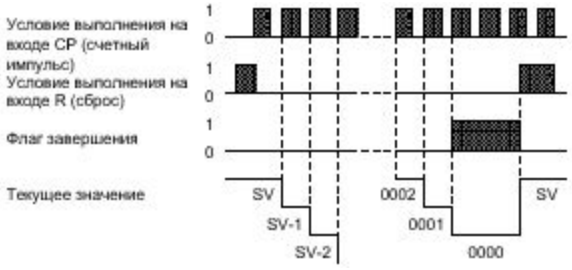
Счетчики

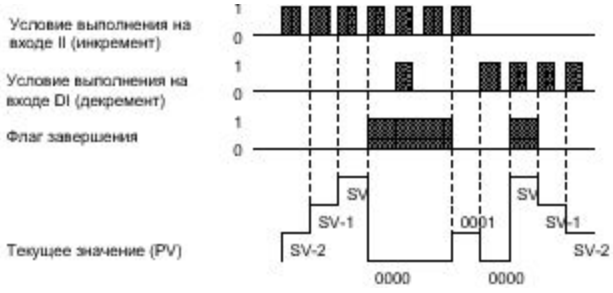


N: TC Number 000 – 511
SV:Set value (word, BCD) IO,
AR, DM, HR, LR, #



N: TC Number 000 – 511
SV:Set value (word, BCD) IO,
AR, DM, HR, LR, #



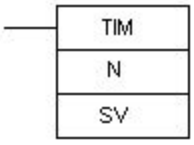


CNT служит для отсчета вниз от заданного значения, когда исполнение условия на счетном входе,CP, изменяется из 0 в 1, т.е текущее значение будет декрементировано (уменьшено на 1) при текущем состоянии счетного входа =1 и состоянием в прошлом цикле =0. При переходе из 1 в 0 состояние счетчика не изменяется. Флаг завершения счета устанавливается в1, когда текущее значение становится равным 0.
Счетчик сбрасывается при единичном значении на входе R. Пока R=1, текущее значение не изменяется. Текущее значение не сбрасывается при использовании счетчика в заблокированных секциях программы и при прерывании питания.

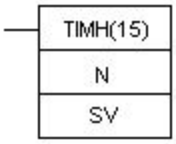
CNTR - реверсивный, двусторонний кольцевой счетчик, т. е он служит для счета от 0 до задания в зависимости от изменения двух условий исполнения: на входе инкрементирования (II) и входе декрементирования (DI).
Флаг счетчика будет находиться в состоянии 1 в случае:
-при инкрементировании был переход текущего значения из состояния ОТСЧИТАНО в 0 и текущее значение не изменяется;
- при декрементировании был переход текущего значения из 0 в состояние ОТСЧИТАНО и текущее значение не изменяется.

Счетчик сбрасывается при единичном значении на входе R. Пока R=1, текущее значение не изменяется. Текущее значение не сбрасывается при использовании счетчика в заблокированных секциях программы и при прерывании питания.

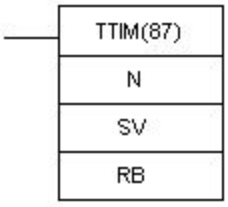
Таймеры



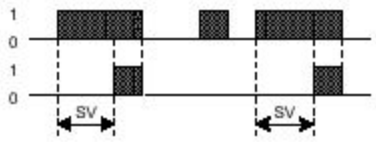
N:TC Number 000 - 511
SV:Set value (word, BCD)
IO, AR, DM, HR, #

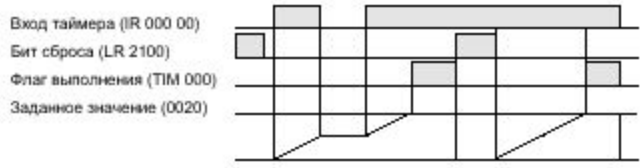


N:TC number # (000 - 015)
SV:Set value (word, BCD)
IO, AR, DM, HR, #



N:TC number TIM 000 through 511
SV:Set Value (word, BCD)
IO, AR, DM, HR, LR, #
RB:Reset Bit IO, AR, HR, LR





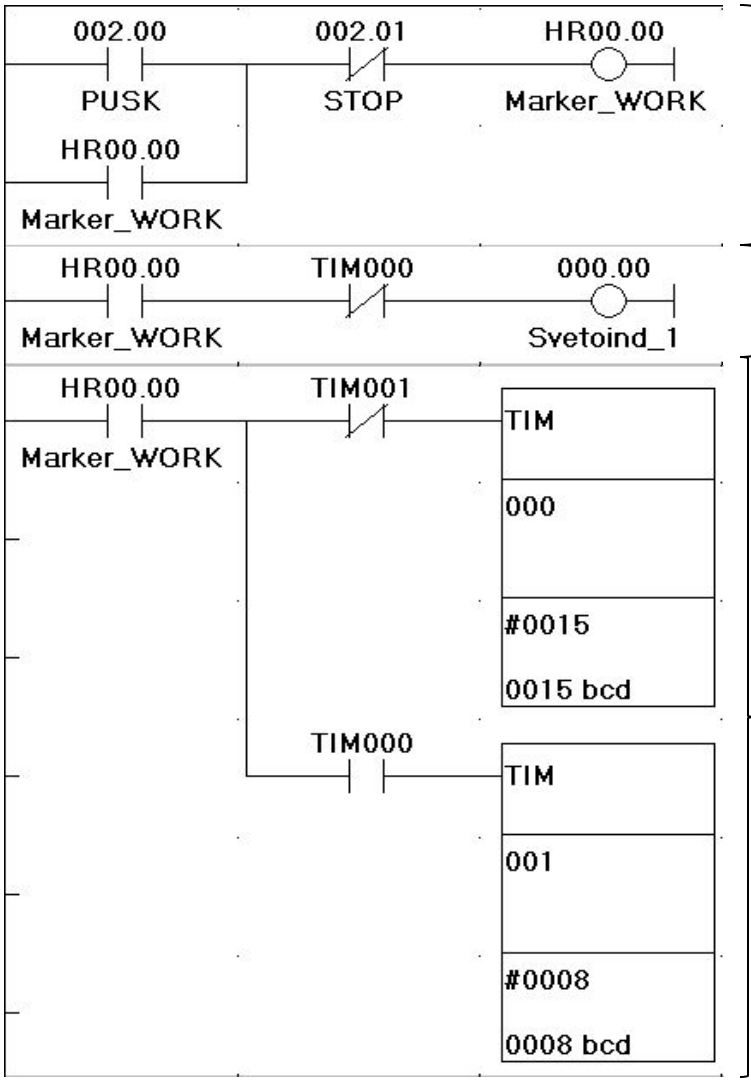
Команды **TIM** и **TIMH** команды декрементирующего таймера, включающегося в 1, и требующие номеров ТС и заданного значения (SV).

Таймер запускается, когда условие срабатывания устанавливается в 1 и сбрасывается (на заданное значение), когда условие срабатывания =0.
После запуска TIM отсчитывает время, вычитая по дискрете (0,1 с) от задания. После запуска TIMH отсчитывает время, вычитая по дискрете (0,01 с) от задания. Если условие срабатывания остается в 1 достаточно долго для отсчета текущего значения до нуля, флаг завершения устанавливается в 1 и остается в 1 до сброса таймера.

TTIM служит для создания таймера, который инкрементирует текущее значение каждые 0,1 с (диапазон счета 0,1… 999,9 с). Таймер будет производить отсчет, пока условие исполнения=1, пока не достигнет значения уставки или не будет сброшен.

Решение задачи

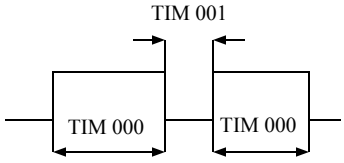
При нажатии на кнопку «ПУСК», на выходе ПЛК начинает работать световая сигнализация в режиме 1,5с горит – 0,8 с выключено. После 5 миганий включается второй выход. Если в момент работы нажать кнопку «СТОП» работа сигнализации прекращается.



Установка рабочего бита HR00.00 при условии нажатия кнопки PUSK. Сброс маркера осуществляется нажатием STOP.

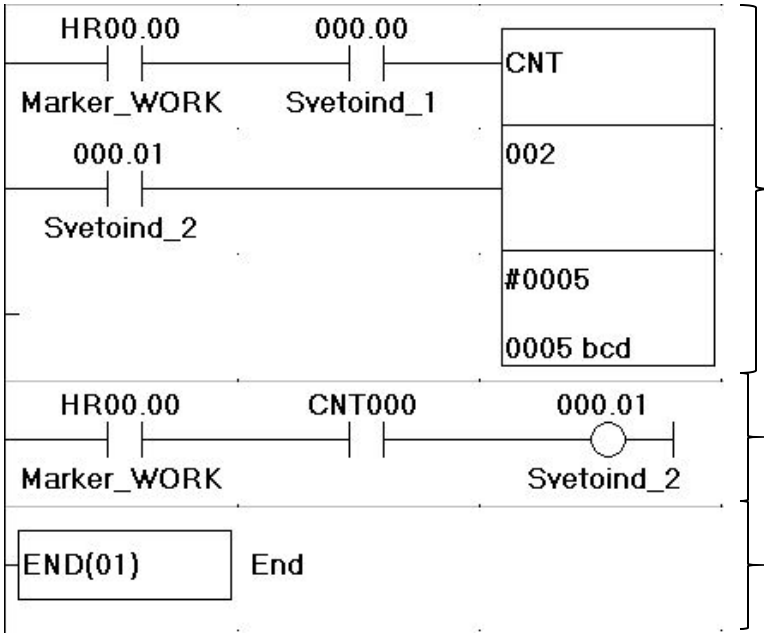
При условии того, что установлен рабочий бит, на момент времени 1,5 с на выходе 000.00 будет сформирован сигнал.

При условии того, что установлен рабочий бит, будет запущен генератор импульсов, в котором TIM 000 формирует длительность единичного уровня сигнала, а TIM 001 длительность отсутствия сигнала.



Решение задачи (продолжение)

При нажатии на кнопку «ПУСК», на выходе ПЛК начинает работать световая сигнализация в режиме 1,5с горит – 0,8 с выключено. После 5 миганий включается второй выход. Если в момент работы нажать кнопку «СТОП» работа сигнализации прекращается.

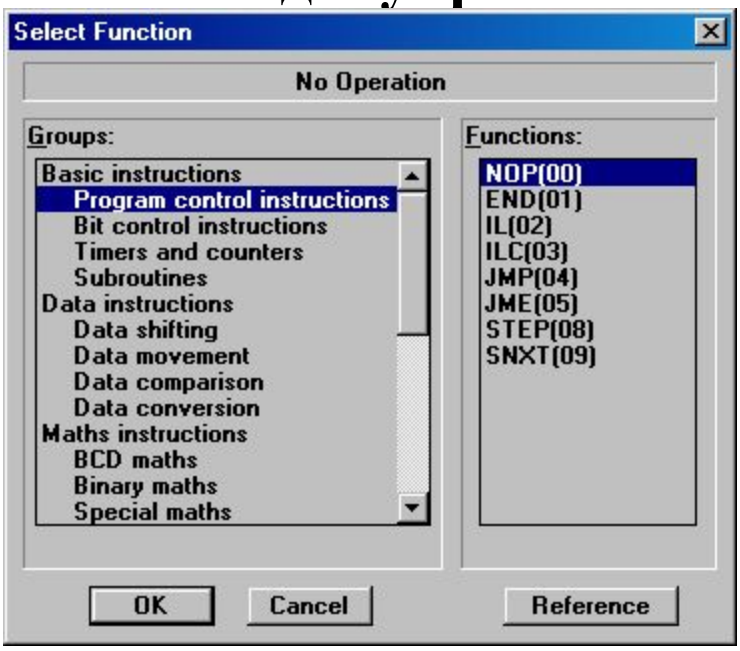


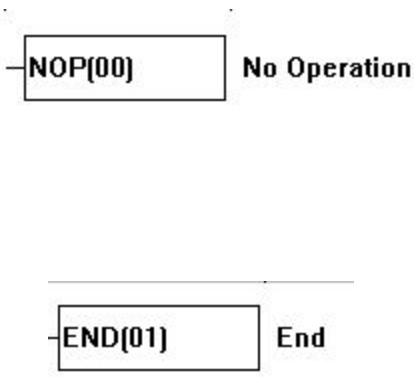
Количество единичных состояний на выходе 000.00 подсчитывается счетчиком, уставка которого = 5

По достижении заданного значения счета, на выходе 000.01 формируется сигнал единичного уровня

Окончание программы

Команды управления программой



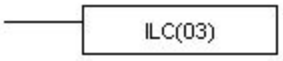


При составлении программы в Ladder Editor применение команды NOP практически не имеет смысла. Когда NOP обнаруживается в программе, действий не производится и программа переходит к следующей команде. Когда память очищена перед программированием, во всех адресах записана команда NOP.

END требуется в качестве последней команды, и располагается в последней, отдельной вычислительной цепочке. Если есть подпрограммы, END помещается после последней подпрограммы. Команды записанные после END не выполняются. END можно поместить в любом месте программы, чтобы выполнялись команды до данного места, что иногда делается для отладки программы.

Если в программе отсутствует END то такая программа не будет загружена в память контроллера и появится сообщение Missing END statement.

Команды управления программой
Секция INTERLOCK



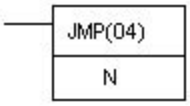
Команда	Обработка
OUT и OUT NOT	Заданные биты устанавливаются в 0
SET и RSET	Сохраняется состояние битов
TIM TIMH	Сброс
TTIM	Сохраняется текущее значение
CNT, CNTR	Сохраняется текущее значение
KEEP	Сохраняется состояние бита
DIFU, DIFD	Не выполняются (смотри ниже)
Все другие команды	Команды не выполняются.

IL всегда используется совместно с ILC для создания секции INTERLOCK.

Перед использованием IL задается условие. Если условие =1 то программа находящаяся внутри секции исполняется без ограничений.
Если условие = 0, то программа будет обрабатываться как показано в таблице.

IL и ILC не обязательно использовать в паре. IL можно использовать несколько раз, каждая IL создает секцию INTERLOCK до ближайшей ILC. ILC можно использовать только когда для нее имеется хотя бы одна IL между ней и любой предыдущей ILC, т.е вложения невозможны.

Команды управления программой
Переход и конец перехода



N:Jump Number 0 to 99

Заданную секцию программу можно пропустить в зависимости от заданных условий исполнения. Хотя это похоже на тот случай, когда условие исполнения для команды INTERLOCK = 0, при JUMP операнды всех условий сохраняют состояние. Переходами можно пользоваться для управления устройствами, которым требуется стабильный выход, т.е. пневматика и гидравлика, в то время как INTERLOCK можно использовать для управления устройств, которые не требуют установившегося выхода, напр. электронные устройства.

JMP всегда применяется совместно с JME для создания переходов, т.е пропуска от одной до другой точки в программе. JMP определяет точку, с которой будет делаться переход JME определяет адрес перехода. Когда условие исполнения для JMP=1, перехода не происходит, и программа выполняется без пропусков. Когда условие для JMP=0, происходит переход к JME с номером, таким же, как и у JMP, и далее выполняются команды после JME. Состояния таймеров, счетчиков и битов управляемых состояниями между JMP и JME не будут изменяться. Номера переходов 01-99 можно использовать только в паре для определения одного перехода. Номера переходов 00-99 используются для сокращения времени цикла.

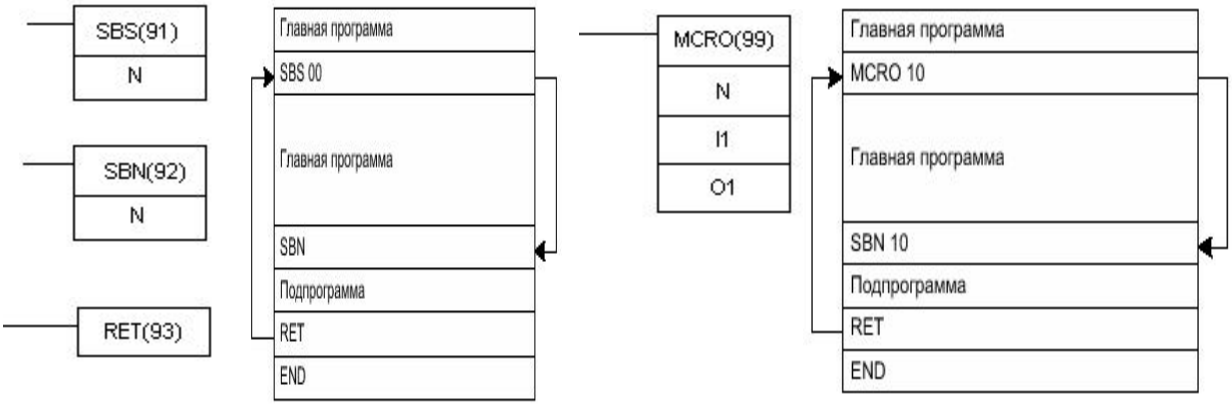
Номер перехода 00 можно использовать сколько угодно раз. Если номер перехода 00, ЦПУ будет искать ближайшую JME с номером 00. Для этого оно производит поиск по всей программе, увеличивая время цикла, по сравнению с другими переходами. Таким образом можно использовать несколько JMP 00 и завершать их одним JME 00.

Команды управления программой
Секция STEP

Команды секции STEP: STEP и SNXT используются совместно для задания точек разрыва между секциями в больших программах, чтобы секции можно было выполнить как блоки и сбрасывать после исполнения. Секция программы обычно определяется для соответствия физическому процессу. Команды секции STEP аналогичны остальным, за исключением того, что некоторые команды (например IL/ ILC, JMP/ JME) нельзя включать между ними.

STEP использует бит управления в области IR и HR для определения начала секции программы. Команда STEP не требует условий исполнения, т.е ее исполнение определяется состоянием бита управления. Для пуска исполнения секции STEP служит SNXT, с тем же управляющим битов, что и в STEP. Если условие исполнения для SNXT=1, то выполняется только секция STEP с тем же битом управления. При начале выполнения другой секции STEP, бит управления предыдущей секции сбрасывается.

Команды управления программой
Подпрограммы



Ranges:		PLC	IN	OUT
N:Subroutine number	0 to 99	C200HS,E,G,X	290	294

Подпрограммы разбивают большие задачи управления на небольшие и позволяют повторно использовать набор команд. Когда главная программа вызывает подпрограмму, управление передается к подпрограмме и выполняются ее команды. После исполнения подпрограммы управление возвращается к главной программе в точку, сразу за точкой, из которой была вызвана подпрограмма.

Ограничение: *номера подпрограмм 00-15 используются с подпрограммами прерываний, а подпрограмма 99 служит для прерываний по расписанию.*

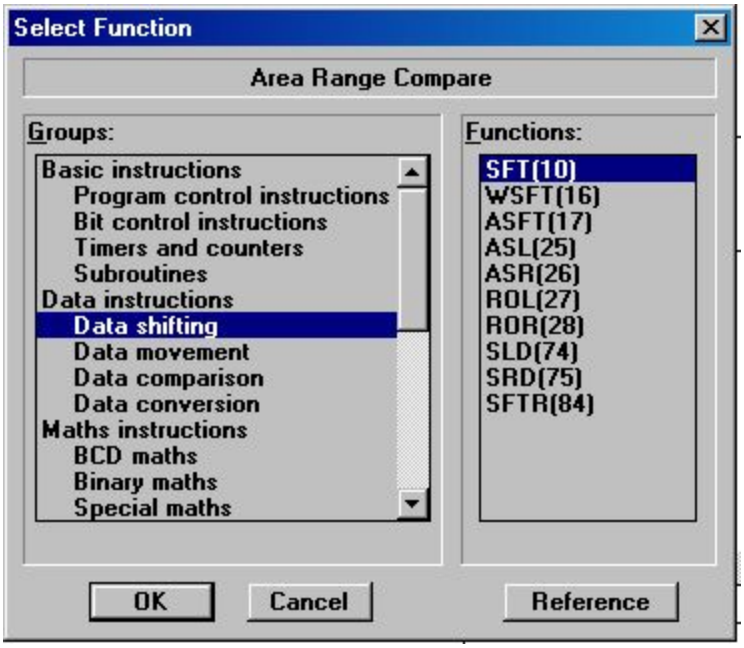
Подпрограмма выполняется путем помещения команды SBS в главной программе в точке, в которой необходим ее вызов . Номер N в SBS указывает номер вызываемой подпрограммы, и начинают выполняться команды между SBN с соответствующим номером и первой командой RET, следующей после нее.

MCRO

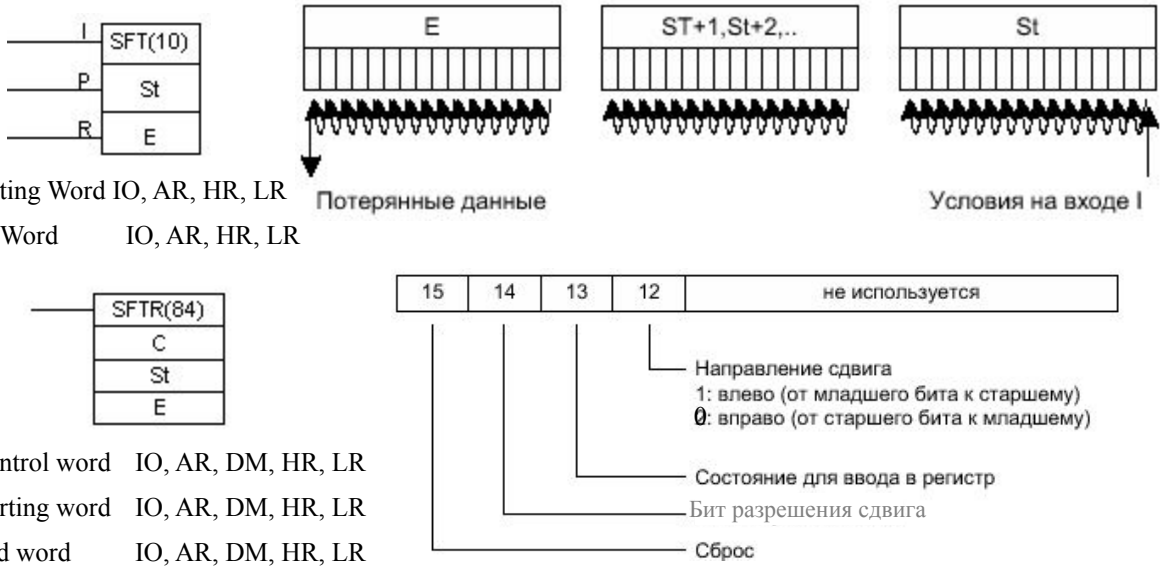
Команда MCRO позволяет одной подпрограмме (образцу) заменить несколько подпрограмм, имеющих идентичную структуру, но разные операнды. Поскольку несколько одинаковых программных секций могут управляться одной подпрограммой, количество шагов программы можно сократить.

Есть 4 слова входа SR 290…SR 293 и 4 слова выхода SR 294… SR 297, которые используются в подпрограмме и берут свое содержимое из I1…I3 и пересылают в O1… O3 при исполнении подпрограммы.

Команды сдвига данных



Команды сдвига данных



SFT – Сдвиговый регистр

Регистр управляется тремя условиями:

I – условие на входе, которое будет записано в момент сдвига;

P – условие сдвига. Сдвиг происходит при переходе из 0 в 1;

R – сброс регистра.

Для сдвига необходимо обеспечить подачу импульса на входе P, на R должен присутствовать сигнал 0, при этом все биты регистра будут сдвигаться влево, а состояние I заносится в младший разряд регистра.

St определяет младшее а E – старшее слово регистра, причем E и St должны находиться в одной области данных и E большее либо равное St.

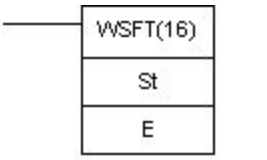
SFTR – Реверсивный регистр сдвига

SFTR служит для создания регистра сдвига одного или нескольких слов, который может сдвигать данные и вправо и влево. St определяет младшее а E – старшее слово регистра, причем E и St должны находиться в одной области данных и E большее либо равное St. Для создания регистра из одного слова задайте одинаковыми St и E.

Данные в регистре сдвига будут сдвигаться на один бит в направлении, указанном битом 12, выталкивая один бит в CY и принимая с другой стороны состояние бита 13, по переднему фронту условия на входе функции и при соблюдении условий слова состояния:

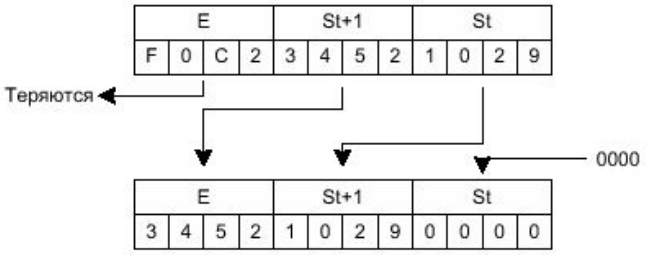
- бит разрешения сдвига = 1;
- бит сброса =0

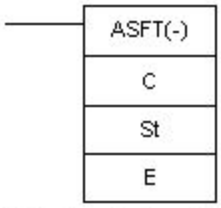
Команды сдвига данных



Differentiated variant available.

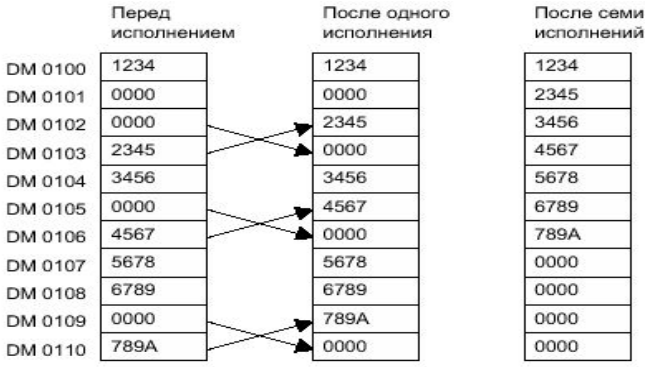
St:Starting word
IO, AR, DM, HR, LR
E:End word
IO, AR, DM, HR, LR





Differentiated variant available.

C:Control word IO, AR, DM, HR, LR, #
St:Starting word IO, AR, DM, HR, LR
E:End word IO, AR, DM, HR, LR



WSFT- Сдвиг слова

Когда условие исполнения =1, WSFT сдвигает данные между ST и E. Нули записываются и ST а содержание E теряется.

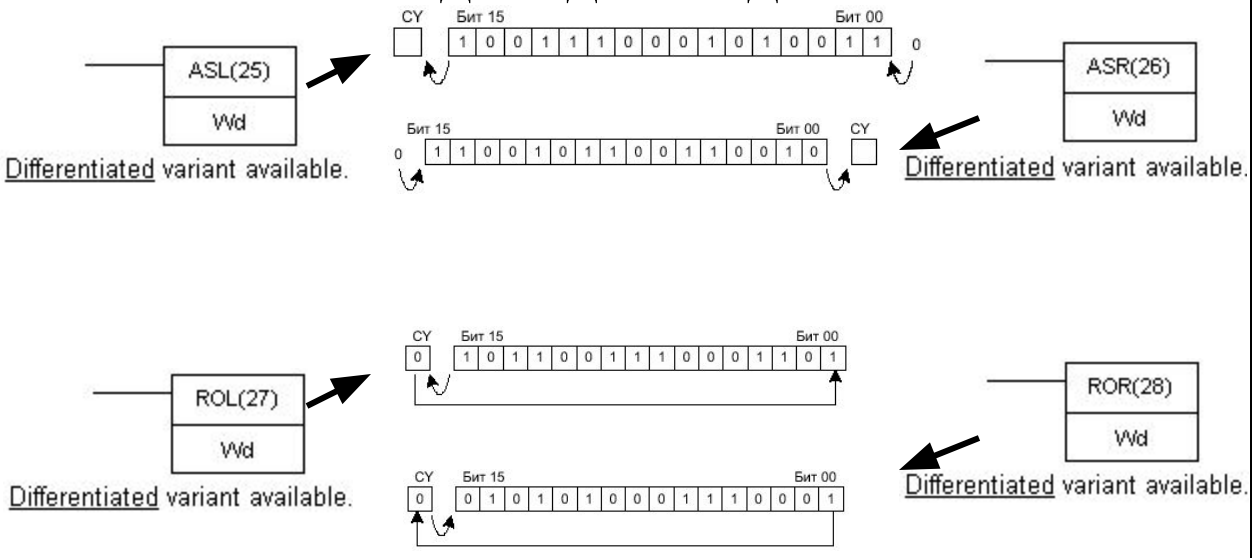
E и St должны находиться в одной области данных и E большее либо равное St.

ASFT – Асинхронный регистр сдвига

Данный регистр сдвигает слова, когда следующее слово в регистре = 0. Когда содержание слова переместиться в следующее (содержащее нули) , содержание исходного слова установиться в 0, т.е каждое нулевое слово в регистре меняется со следующим словом. Направление сдвига и сброс регистра определяется в слове управления:

- бит 13 – направление сдвига(1 – направление вниз, к младшим словам, 0 – к старшим словам);
- бит 14 – бит разрешения сдвига (1 – разрешение сдвига, 0 – запрет сдвига);
- бит 15 – бит сброса (при 1 регистр будет сброшен).

Команды сдвига данных



Wd: Shift word IO, AR, DM, HR, LR

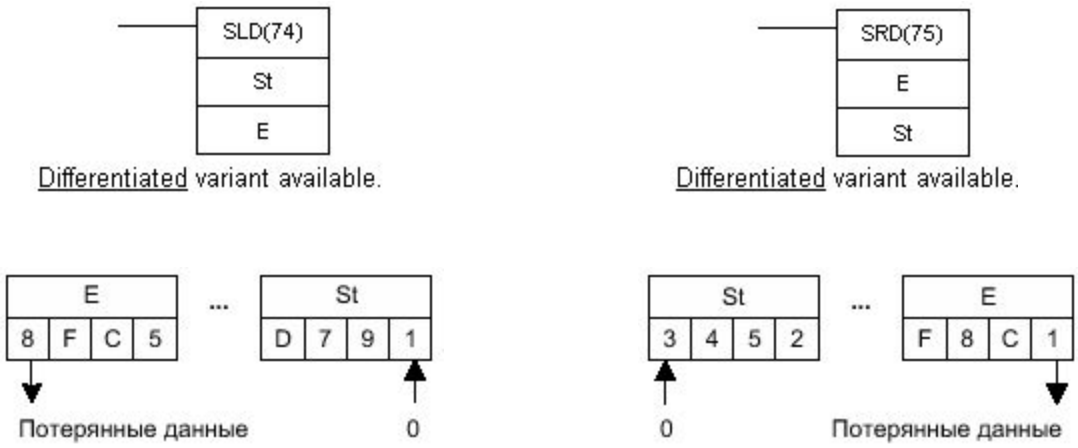
ASL- арифметический сдвиг слова влево
Когда условие исполнения =0, команда не выполняется. Когда условие исполнения =1, ASL помещает 0 в нулевой бит слова Wd, сдвигает на 1 биты слова WD влево, и переносит состояние бита 15 в CY. Данные перемещенные в CY при следующем сдвиге теряются.

ASR- арифметический сдвиг слова вправо
Когда условие исполнения =0, команда не выполняется. Когда условие исполнения =1, ASR помещает 0 в пятнадцатый бит слова Wd, сдвигает на 1 биты слова WD вправо, и переносит состояние бита 00 в CY. Данные перемещенные в CY при следующем сдвиге теряются.

ROL- циклический сдвиг слова влево
Когда условие исполнения =0, команда не выполняется. Когда условие исполнения =1, ROL сдвигает на 1 биты слова WD влево, помещает CY в бит 00 Wd, и переносит состояние бита 15 в CY.

ROR- циклический сдвиг слова вправо
Когда условие исполнения =0, команда не выполняется. Когда условие исполнения =1, ROR сдвигает на 1 биты слова WD вправо, помещает CY в бит15 Wd, и переносит состояние бита 00 в CY.

Команды сдвига данных



E:End word IO, AR, DM, HR, LR

St:Starting word IO, AR, DM, HR, LR

SLD – Сдвиг влево на одну цифру

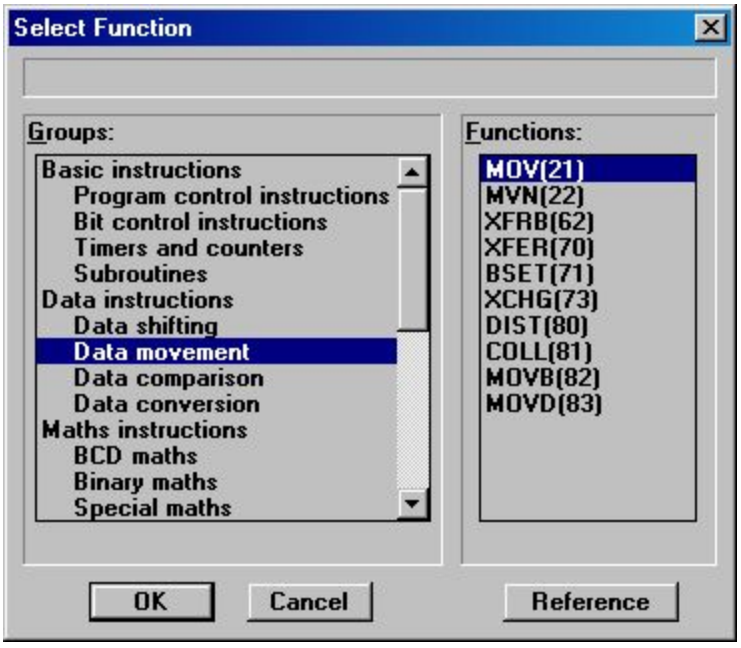
Когда условие исполнения =0, SLD не выполняется. Когда условие исполнения =1 SLD сдвигает данные между St и E (включительно) на одну цифру (4 бита) влево. В младшую цифру St записываются нули, а старшая цифра E теряется.

SRD – Сдвиг влево на одну цифру

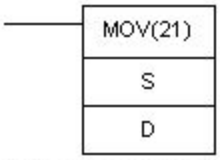
Когда условие исполнения =0, SRD не выполняется. Когда условие исполнения =1 SRD сдвигает данные между St и E (включительно) на одну цифру (4 бита) вправо. В старшую цифру St записываются нули, а младшая цифра E теряется.

***Предосторожности**: если во время операции сдвига более 50 слов происходит прерывание питания, операция сдвига может не завершиться. Задавайте диапазон между St и E не более 50 слов.*

Команды перемещения данных



Команды перемещения данных

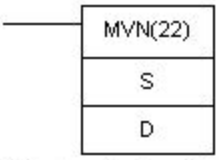


Differentiated variant available.

S:Source word IO, AR, DM, HR, TC, LR, #

D:Destination word IO, AR, DM, HR, LR

Состояние бит не изменяется



Differentiated variant available.

Состояние бит инвертируется

MOV- пересылка

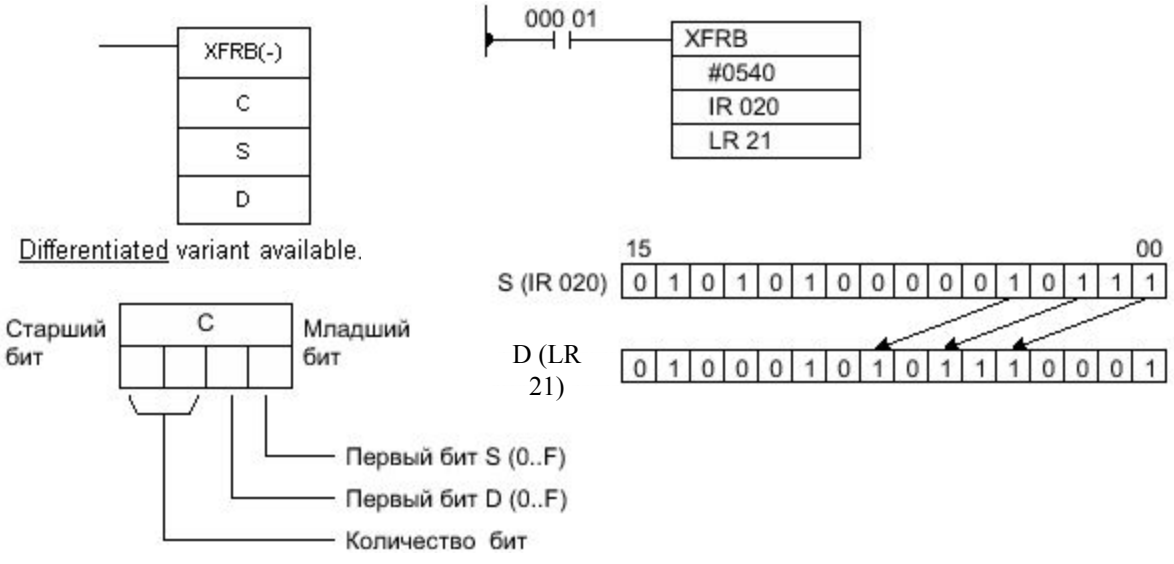
Когда условие исполнения =0, MOV не выполняется. Когда условие =1, MOV копирует содержимое S в D.

Предосторожности: номера ТС нельзя задавать в качестве D для изменения их текущего значения. Однако текущее состояние легко изменить командой BSET.

MVN- пересылка инверсного значения

Когда условие исполнения =0, MVN не выполняется. Когда условие =1, MVN копирует инвертированное содержимое S в D, т.е для каждого бита S=0 соответствующий бит в D будет=1.

Команды перемещения данных



XFRB – переслать биты

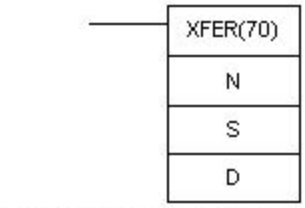
Когда условие исполнения =0, XFRB не выполняется. Когда условие =1, XFRB копирует указанные биты источника в указанные биты приемника. Две младшие цифры слова C задают начальные биты в S и D, а две старшие цифры C задают количество битов, подлежащих копированию.

Ограничения

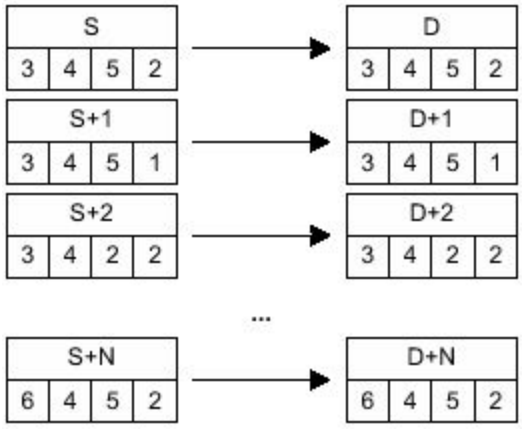
Заданные биты источника должны находиться в одной области данных.
Заданные биты приемника должны находиться в одной области данных.
За раз можно копировать до 255 битов.

В примере XFRB используется для передачи 5 битов из IR 020 в LR 21. Стартовый бит в IR 020 = 0, а стартовый бит в LR 21=4, так что IR 02000…IR 02004 копируются в LR2104… LR2108

Команды перемещения данных



Differentiated variant available.



N:Number of words IO, AR, DM, HR, TC, LR, #

St:1st source word IO, AR, DM, HR, TC, LR

D:1st destination word IO, AR, DM, HR, TC, LR

XFER – переслать блок

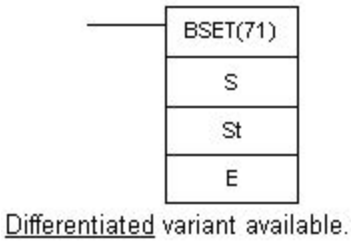
Когда условие исполнения =0, XFER не выполняется. Когда условие =1, XFER копирует содержимое S, S+1,…S+N в D, D+1,… D+N.

Ограничения

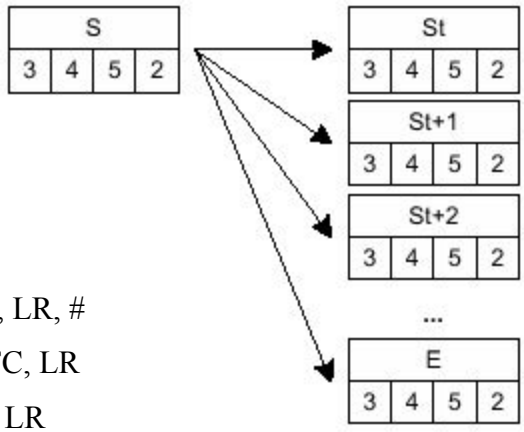
S… S+N и D… D+N должны лежать в одной области данных, но области их блоков могут перекрываться. S и D могут лежать в одной области данных и области ох блоков могут пересекаться.

N должно быть двоично – десятичным в диапазоне 0000 … 6144

Команды перемещения данных



Differentiated variant available.



S: Source Data IO, AR, DM, HR, TC, LR, #

St: Starting word IO, AR, DM, HR, TC, LR

E: End word IO, AR, DM, HR, TC, LR

BSET – заполнение блока

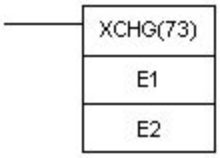
Когда условие исполнения =0, BSET не выполняется. Когда условие =1, BSET копирует содержимое S во все слова от St до E.

Ограничения

St должно быть меньше либо равно E, St и E должны лежать в одной области памяти данных.

BSET можно использовать для изменения текущего значения таймеров и счетчиков.(Это нельзя сделать командами MOV и MVN). BSET можно также использовать для очистки блока данных, т. е области DM, для подготовки исполнения других команд.

Команды перемещения данных



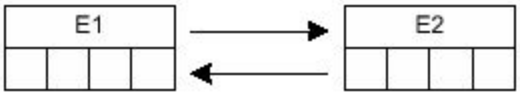
Differentiated variant available.

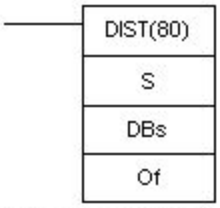
E1: Exchange word 1

IO, AR, DM, HR, TC, LR

E2: Exchange word 2

IO, AR, DM, HR, TC, LR



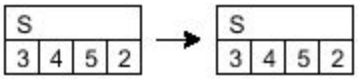


Differentiated variant available.

S: Source Data IO, AR, DM, HR, TC, LR, #

DBs: 1st destination word IO, AR, DM, HR, TC, LR

Of: Offset data (BCD) IO, AR, DM, HR, TC, LR, #



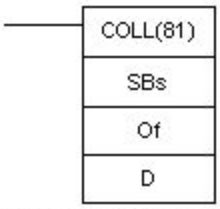
XCHG – обмен данными.
Когда условие исполнения =0, XCHG не выполняется. Когда условие =1, XCHG обменивает данными E1 и E2.

DIST – распределение одного слова
В зависимости от значения Of DIST служит либо как команда распределения данных, либо как команда работы со стеком. Если Of находится между 0000 … 6655, DIST будет работать как команда распределения данных и копировать содержимое S в DBS+Of . Если старшая цифра Of = 9, DIST будет работать со стеком и создавать стек с числом слов, заданным в трех младших цифрах Of.

При использовании команды для распределения данных, DIST копирует содержимое S в DBs+Of, т.е Of добавляется в DBs для определения слова приемника.

Ограничения
Of должно быть в двоично – десятичном виде. DBs должно быть в той же области что и DBs+Of.

Команды перемещения данных

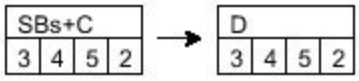


Differentiated variant available.

SBs:Source base word IO, AR, DM, HR, TC, LR

Of: Offset data (BCD) IO, AR, DM, HR, TC, LR, #

D: Destination word IO, AR, DM, HR, TC, LR



COLL – сбор данных

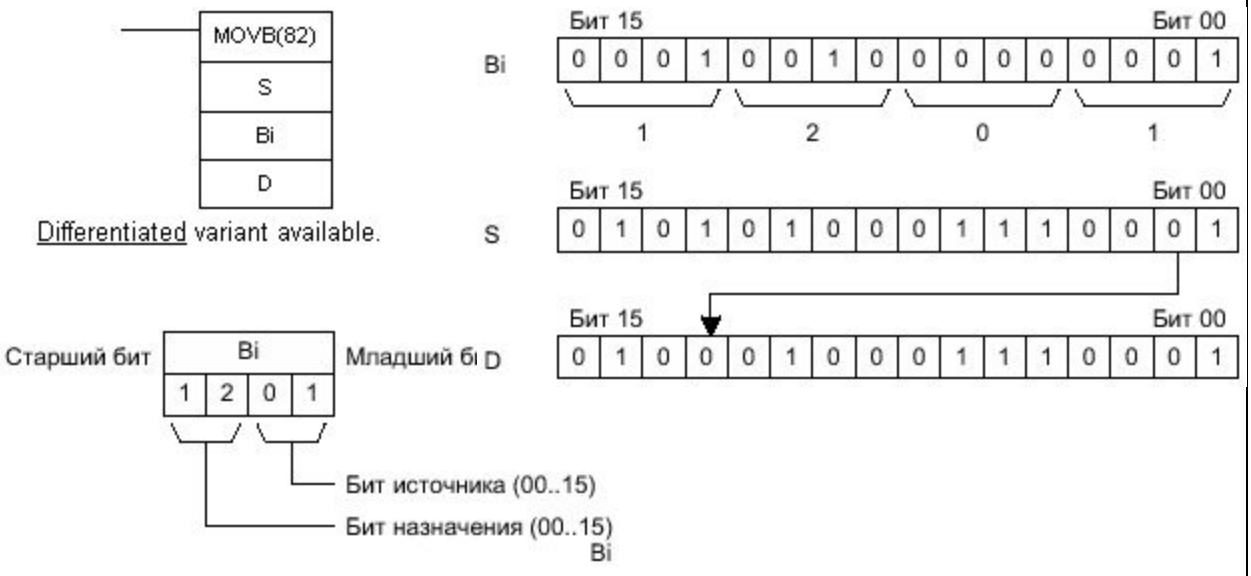
В зависимости от значения Of COLL может действовать как команда сбора данных, команда работы со стеком FIFO или как команда работы со стеком LIFO. Если Of находится между 0000 … 6655, COLL действует как команда сбора данных и копирует содержимое SBs+Of в D. Если старшая цифра Of = 9, COLL будет работать со стеком FIFO. Если старшая цифра Of = 8, COLL будет работать со стеком LIFO. Обе операции со стеком используют стек, начиная с SBs с длиной, заданной в трех младших цифрах Of.

При использовании операции сбора данных, COLL копирует содержимое SBs+ Of в D, т.е Of добавляется в SBs для определения слова приемника.

Ограничения

Of должно быть в двоично – десятичном виде. SBs должно быть в той же области что и SBs+Of.

Команды перемещения данных



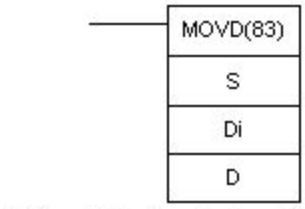
MOVB – переслать бит

Когда условие исполнения =0, MOVB не выполняется. Когда условие =1, MOVB копирует указанный бит S в указанный бит D. Биты S и D задаются в Bi. Две младшие цифры слова Bi задают бит источника, две старшие цифры слова Bi задают бит приемника.

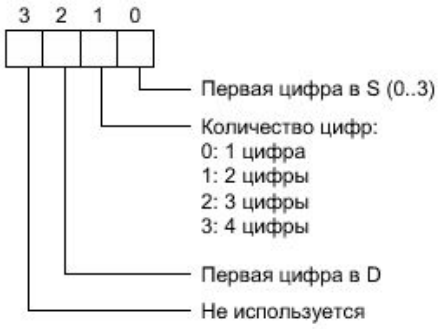
Ограничение

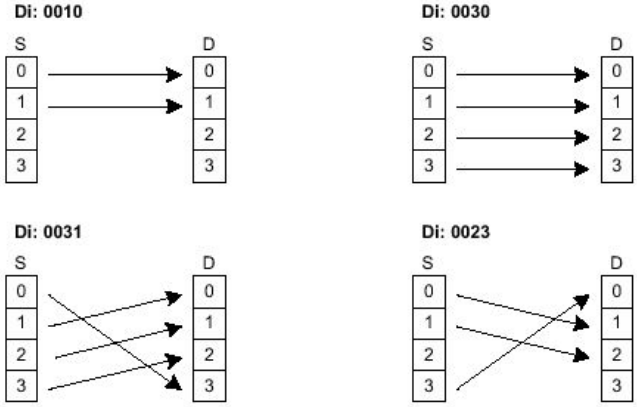
Две младших цифры и две старших цифры Bi должны быть в диапазоне 00… 15.

Команды перемещения данных



Differentiated variant available.





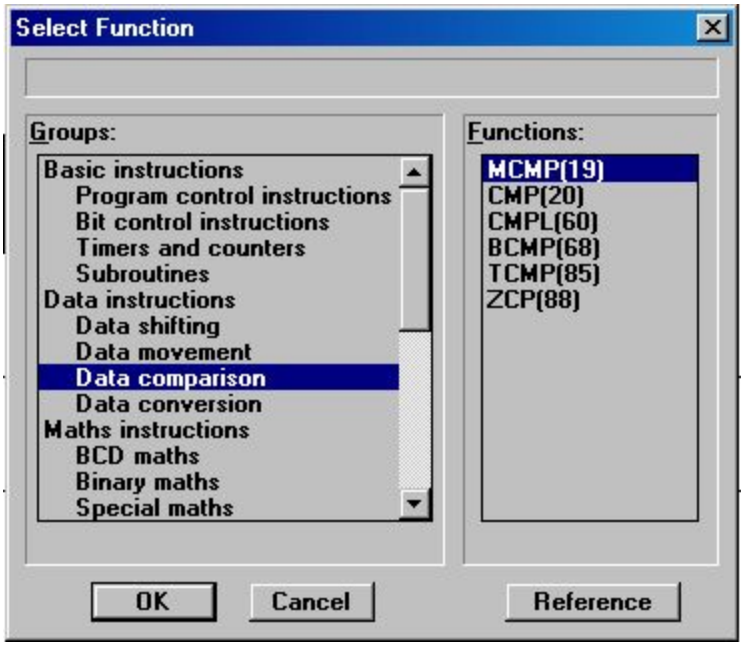
MOVB – переслать бит

Когда условие исполнения =0, MOVD не выполняется. Когда условие =1, MOVD копирует содержание указанных цифр из S в указанные цифры D. За один раз можно переслать до 4 цифр. Цифры из S будут копироваться последовательно в D, начиная с указанной цифры. После записи последней цифры D, оставшиеся цифры S будут записаны в D начиная с нулевой цифры.

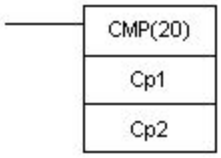
Ограничение

Три младших цифры Di должны быть в диапазоне 0…3 .

Команды сравнения

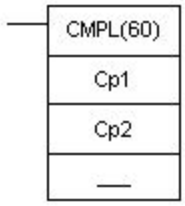


Команды сравнения



Differentiated variant available

Cp1:1st compare word IO, AR, DM, HR, TC, LR, #
Cp2: 2nd compare word IO, AR, DM, HR, TC, LR, #



Differentiated variant available.

Cp1:1st word of first compare word pair
IO, AR, DM, HR, TC, LR
Cp2:1st word of second compare word pair
IO, AR, DM, HR, TC, LR

Флаг	Адрес	C1 < C2	C1 = C2	C1> C2
GR	25505	0	0	1
EQ	25506	0	1	0
LE	25507	1	0	0

CMP – сравнение

Когда условие исполнения =0, CMP не выполняется. Когда условие =1, CMP сравнивает содержимое слов Cp1 и Cp2 и выдает результат во флаги GR, EQ и LE в области SR.

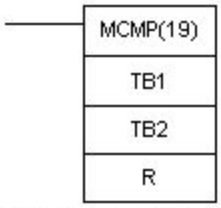
Ограничения

При сравнении текущего значения таймера или счетчика значение должно быть BCD. Размещение между командой CMP и командами, которые используют флаги GR, EQ и LE , других команд , могут изменить состояние этих флагов. Используйте эти флаги перед тем, как они изменятся. CMP нельзя использовать для сравнения чисел со знаком.

CMPL – сравнение чисел двойной длины

Когда условие исполнения =0, CMPL не выполняется. Когда условие =1, CMPL объединяет 4- значное 16- речное содержимое слов Cp1 с 4- значным содержимым Cp1+1 и содержимое Cp2 с Cp2+1 для создания 8 – значных чисел. Два 8- значных числа сравниваются и результатом являются флаги GR, EQ и LE в области SR.

Команды сравнения

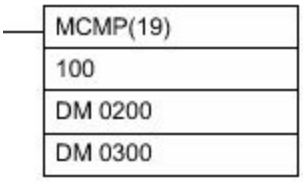


Differentiated variant available.

TB1: 1st word of table 1 IO, AR, DM, HR, TC, LR

TB2: 1st word of table 2 IO, AR, DM, HR, TC, LR

R: Result word IO, AR, DM, HR, TC, LR



TB1: IR 100		TB2: DM 0200		R: DM 0300	
IR 100	0100	DM 0200	0100	DM 030000	0
IR 101	0200	DM 0201	0200	DM 030001	0
IR 102	0210	DM 0202	0210	DM 030002	0
IR 103	ABCD	DM 0203	0400	DM 030003	1
IR 104	ABCD	DM 0204	0500	DM 030004	1
IR 105	ABCD	DM 0205	0600	DM 030005	1
IR 106	ABCD	DM 0206	0700	DM 030006	1
IR 107	0800	DM 0207	0800	DM 030007	0
IR 108	0900	DM 0208	0900	DM 030008	0
IR 109	1000	DM 0209	1000	DM 030009	0
IR 110	ABCD	DM 0210	0210	DM 030010	1
IR 111	ABCD	DM 0211	1200	DM 030011	1
IR 112	ABCD	DM 0212	1300	DM 030012	1
IR 113	1400	DM 0213	1400	DM 030013	0
IR 114	0210	DM 0214	0210	DM 030014	0
IR 115	1212	DM 0215	1600	DM 030015	1

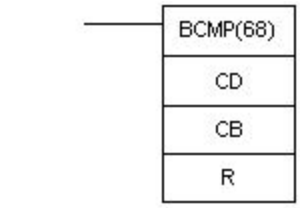
MCMP – сравнение нескольких слов

Когда условие исполнения =0, MCMP не выполняется. Когда условие =1, MCMP сравнивает содержимое TB1 с TB2, TB1+1 с TB2+1, … , TB1+15 с TB2+15. Если первая пара равна, первый бит слова (бит 00) устанавливается в 0, и т.д., т.е если содержание TB1+1 равно содержанию TB2+1, бит 01 устанавливается в 0 и т.д. Остальные биты R будут в состоянии 1.

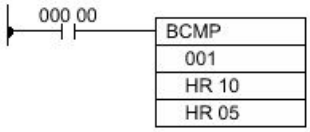
Ограничения

TB1 и TB1+15, TB2 и TB2+15 должны лежать в одной области данных.

Команды сравнения



Differentiated variant available.



CD: 001 IR 001 0210

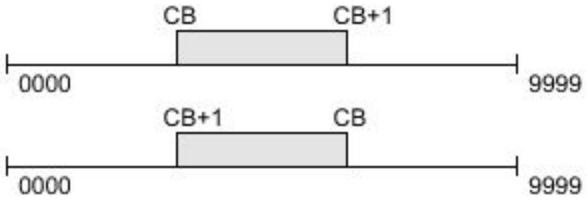
Сравнить данные в IR 001 (там содержится 210) с приведенными зонами

Верхние границы		Нижние границы		R: HR 05	
HR 10	0000	HR 11	0100	HR 0500	0
HR 12	0101	HR 13	0200	HR 0501	0
HR 14	0201	HR 15	0300	HR 0502	1
HR 16	0301	HR 17	0400	HR 0503	0
HR 18	0401	HR 19	0500	HR 0504	0
HR 20	0501	HR 21	0600	HR 0505	0
HR 22	0601	HR 23	0700	HR 0506	0
HR 24	0701	HR 25	0800	HR 0507	0
HR 26	0801	HR 27	0900	HR 0508	0
HR 28	0901	HR 29	1000	HR 0509	0
HR 30	1001	HR 31	1100	HR 0510	0
HR 32	1101	HR 33	1200	HR 0511	0
HR 34	1201	HR 35	1300	HR 0512	0
HR 36	1301	HR 37	1400	HR 0513	0
HR 38	1401	HR 39	1500	HR 0514	0
HR 40	1501	HR 41	1600	HR 0515	0

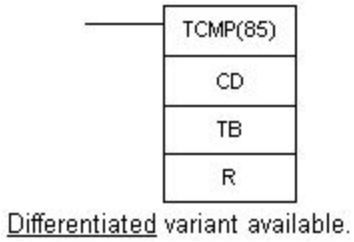
BCMP – блоковое сравнение

Когда условие исполнения =0, BCMP не выполняется. Когда условие =1, BCMP сравнивает значение CD с зонами, заданными блоком, состоящим из CB,CB+1,…, CB+31. Каждая зона задается двумя словами, первое задает нижнюю,второе – верхнюю. Если CD оказывается внутри одной из таких зон, включая границы, устанавливается соответствующий бит в R. Остальные значения R = 0.

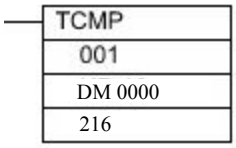
Как правило, первое слово в зоне меньше второго, но если первое слово в зоне больше второго, соответствующий бит в R установиться в 1, когда CD вне зоны, заданной двумя словами, как показано на диаграмме.

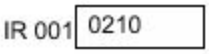


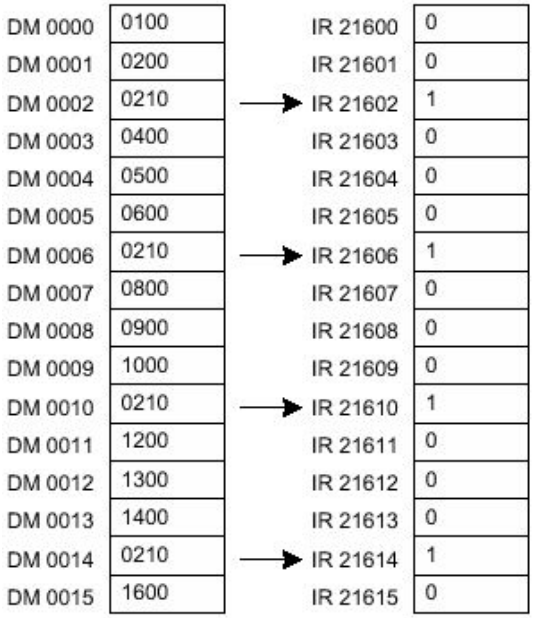
Команды сравнения



Differentiated variant available.







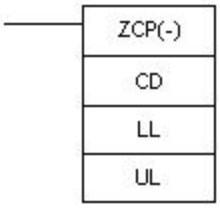
TCMP – табличное сравнение

Когда условие исполнения =0, TCMP не выполняется. Когда условие =1, TCMP сравнивает CD с содержанием TB1, TB1+1, … , TB1+15. Если CD равно содержимому одного из слов, устанавливается соответствующий бит в R. Остальные биты R будут установлены в 0.

Ограничения

TB и TB+15 должны быть в одной области данных.

Команды сравнения



CD:Compare data IO, AR, DM, HR, TC, LR, #

LL: Lower limit of range IO, AR, DM, HR, TC, LR, #

UL: Upper limit of range IO, AR, DM, HR, TC, LR, #

Differentiated variant available.

Результат сравнения	Состояние флага		
	GR (SR 25505)	EQ (SR 25506)	LE (SR 25507)
CD<LL	0	0	1
LL≤CD≤UL	0	1	0
UL<CD	0	0	0

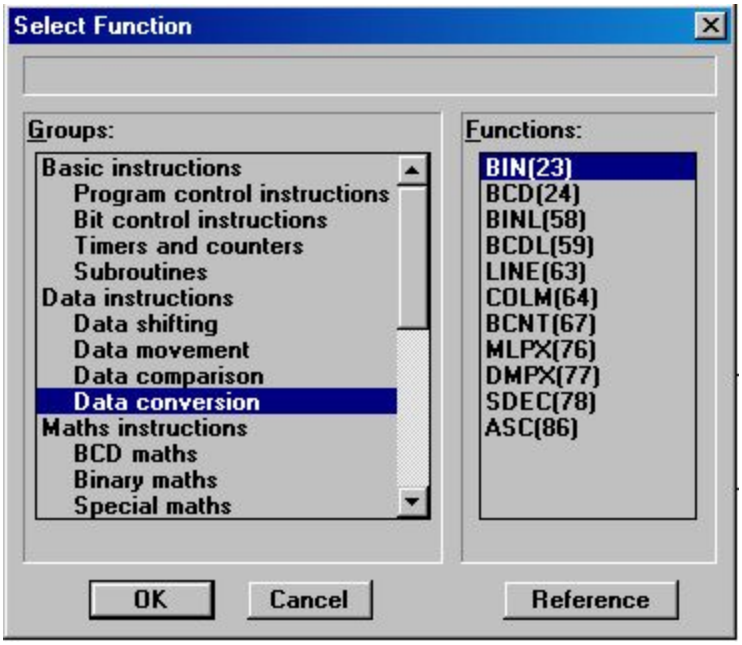
ZCP – сравнение с зоной

Когда условие исполнения =0, ZCP не выполняется. Когда условие =1, ZCP сравнивает CD в зоне, заданной нижней границей LL и верхней границей UL и выдает результат во флаги GR, EQ и LE в области SR.

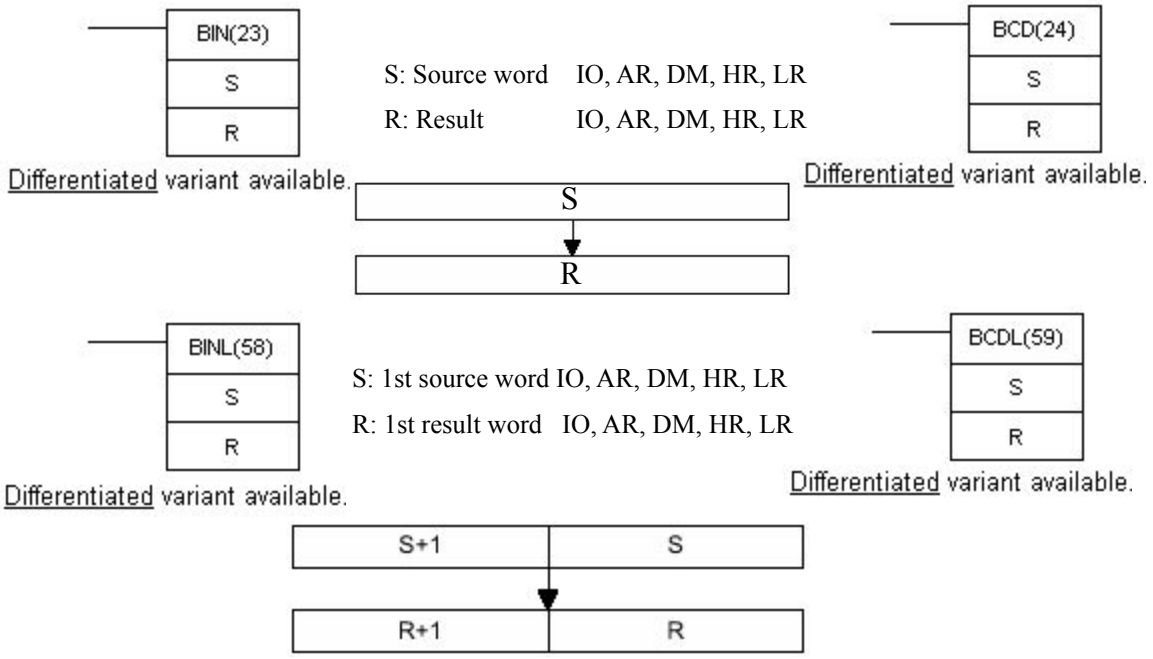
Ограничения

LL должна быть меньше либо равна UL. Размещение между командой ZCP и командами, которые используют флаги GR, EQ и LE , других команд , могут изменить состояние этих флагов. Используйте эти флаги перед тем, как они изменятся

Команды преобразования данных



Команды преобразования данных



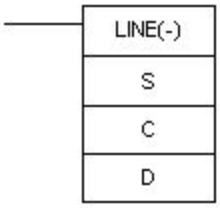
BIN – преобразование двоично- десятичного числа в двоичное
Когда условие исполнения =0, BIN не выполняется. Когда условие =1, BIN преобразует двоично- десятичное содержание S в двоичный эквивалент и помещает двоичное значение в R. Изменяется только содержание R, содержание S не изменяется.

BCD – преобразование двоичного числа в двоично- десятичное
Когда условие исполнения =0, BCD не выполняется. Когда условие =1, BCD преобразует двоичное содержание S в двоично- десятичный эквивалент и помещает значение в R. Изменяется только содержание R, содержание S не изменяется.

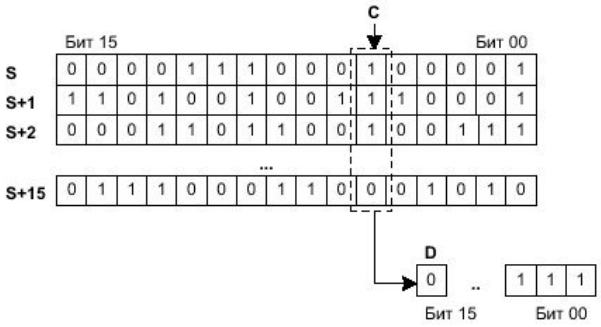
BINL – преобразование двоично- десятичного числа двойной длины в двоичное двойной длины
Когда условие исполнения =0, BINL не выполняется. Когда условие =1, BINL преобразует длинное число (8 цифр)из S и S+1 в 32- битовое двоичное число и помещает двоичное значение в R и R+1. Изменяется только содержание R, содержание S не изменяется.

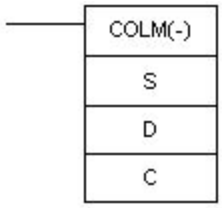
BCDL – преобразование двоичного числа двойной длины в двоично-десятичное число двойной длины
Когда условие исполнения =0, BCDL не выполняется. Когда условие =1, BCDL преобразует 32- битовое содержимое S и S+1 в 8 двоично- десятичных цифр и помещает результат преобразования в R и R+1.
Ограничение
Если содержимое S превышает 05F5 E0FF, результат будет больше 9999 9999 и BCDL не выполнится и содержание R и R+1 не изменится.

Команды преобразования данных

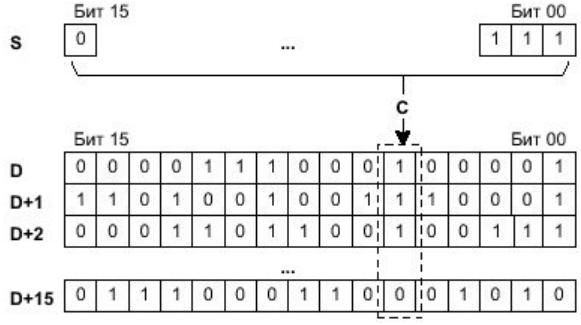


Differentiated variant available.





Differentiated variant available.



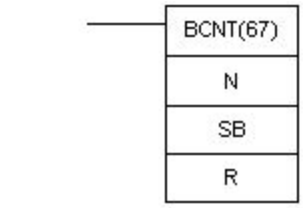
LINE – преобразование Столбец – в Строку
Когда условие исполнения =0, LINE не выполняется. Когда условие =1, LINE копирует столбец битов C из блока 16 слов (от S до S+15) в 16 бит слова D.

Ограничения
S и S+15 должны находиться в одной области данных. C должно быть BCD между # 0000 и # 0015.

COLM – преобразование Строка- в - Столбец
Когда условие исполнения =0, COLM не выполняется. Когда условие =1, COLM копирует 16 битов слова S в столбец битов, C, блока из 16 слов (D…D+15).

Ограничения
D и D+15 должны находиться в одной области данных. C должно быть BCD между # 0000 и # 0015.

Команды преобразования данных



Differentiated variant available.

N: Number of words IO, AR, DM, HR, TC, LR, #

SB: 1st source word IO, AR, DM, HR, TC, LR

R: Destination word IO, AR, DM, HR, TC, LR

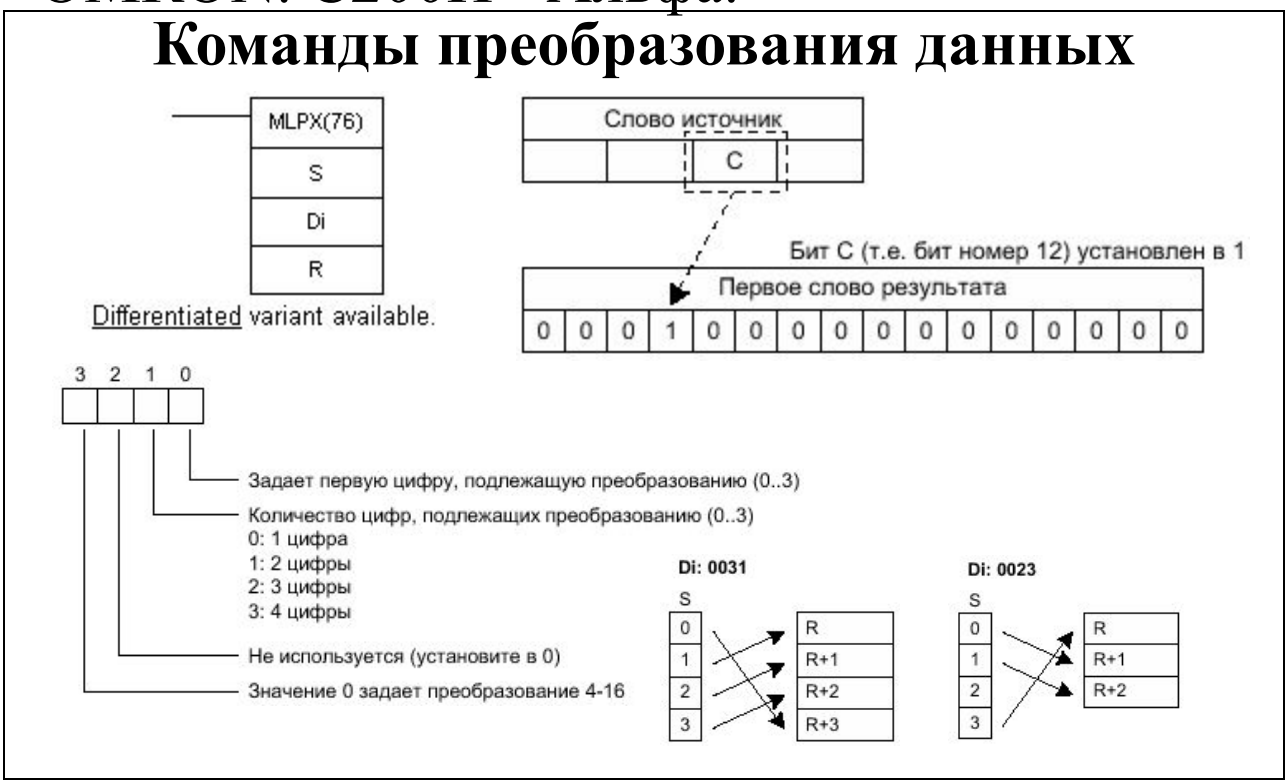
BCNT – счетчик битов

Когда условие исполнения =0, BCNT не выполняется. Когда условие =1, BCNT считает общее количество битов в состоянии 1 во всех словах между SB и SB+(N-1) и помещает результат в R.

Ограничения

N должен быть BCD в диапазоне 0000… 6656.

Команды преобразования данных



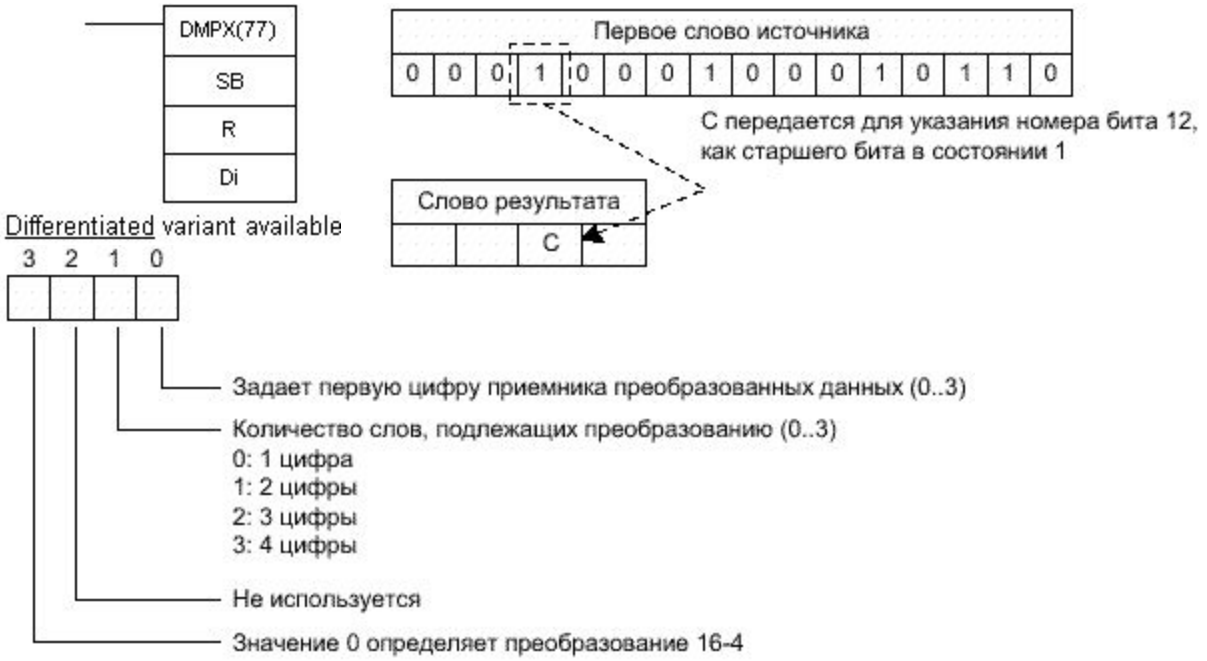
MPLX – преобразователь 4-в-16 / 8-в-256
В зависимости от значений C MPLX работает ка преобразователь 4 бит в 16 бит, или как 8 бит в 256 бит.

Преобразователь 4-в-16
MLPX работает как преобразователь 4-в-16**,** когда старшая цифра C=0.
16- ричное значение цифр в источнике S служит для указания битов в словах результата (до четырех). Указанный бит в каждом слове результата будет =1, а остальные 15 бит в каждом слове = 0. Если задано более одной цифры, тогда один бит будет установлен в 1 в каждом из последовательных слов, начинающихся с R.

Преобразователь 8-в-256
MLPX работает как преобразователь 8 бит в 256 бит **,** когда старшая цифра C=1.
16- ричное значение байта в источнике S служит для указания битов в словах результата (до двух), или двух групп из 16 последовательных слов результата (256 бит). Указанный бит в каждой группе результата будет =1, а остальные 255 бит этой группы = 0.

Ограничения
Когда старшая цифра C=0, две младшие цифры C должны быть 0… 3.
Когда старшая цифра C=1, две младшие цифры C должны быть 0… 1.
Слова результата должны находиться в одной области памяти.

Команды преобразования данных



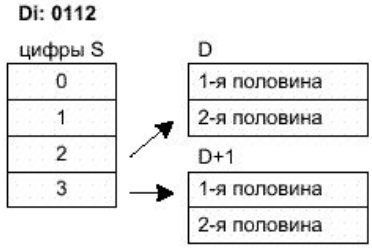
DMPX – преобразователь 16 -в- 4/ 256 -в- 8
В зависимости от значения C DMPX работает как преобразователь 16- бит в 4- бит, или как 256 – бит в 8 – бит.

Ограничение
Когда старшая цифра C = 0, две правые цифры C должны лежать в диапазоне 0..3
Когда старшая цифра C = 1, две правые цифры C должны лежать в диапазоне 0..1
Все слова результата должны находиться в одной области данных.

Преобразователь 16 – бит в 4 – бит
DMPX работает как преобразователь 16- бит в 4- бит, когда C=0. Когда условие исполнения = 1, DMPX определяет позицию старшего бита слова в SB, установленного в 1 и кодирует его в слово R.

Преобразователь 256 – бит в 8 – бит
DMPX работает как преобразователь 256- бит в 8- бит, когда C=1. Когда условие исполнения = 1, DMPX определяет позицию старшего бита установленного в 1, в группе из 16 слов источника от SB до SB+15 или от SB+16 до SB+31, и кодирует его в двухразрядную 16- ричную цифру, соответствующую положению бита между 256 битов группы, затем передает это значение в указанный байт в R.

Команды преобразования данных



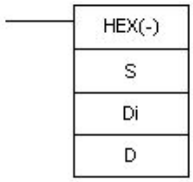
Цифра	Код ASCII
0	30
1	31
2	32
3	33
4	34
5	35
6	36
7	37
8	38
9	39
A	41
B	42
C	43
D	44
E	45
F	46

ASC- преобразование в коды ASCII
Когда условие исполнения =0, ASC преобразует указанные цифры слова S в 8 – битовый эквивалент кода ASCII и помещает его в слово приемника, начинающегося с D.

Любая или все цифры в S преобразуются последовательно, начиная от указанной первой цифры. Первая цифра, количество цифр и половина D, принимающая первый код ASCII задаются в Di. Если задано больше цифр, чем остается в S, остальные цифры будут отсчитываться с начала S.

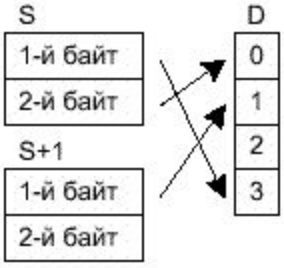
Контроль четности
Старший бит каждого знака ASCII может быть автоматически настроен на контроль четности или нечетности.
Когда контроль установлен на «четно», старший бит будет принимать такое значение, чтобы общее количество битов, находящихся в состоянии 1 было четным. Например, код ASCII «31»(0011 0001) будет равным «B1» (1011 0001) т.е., бит четности установится в 1, для того , чтобы общее количество единичных бит было четным.

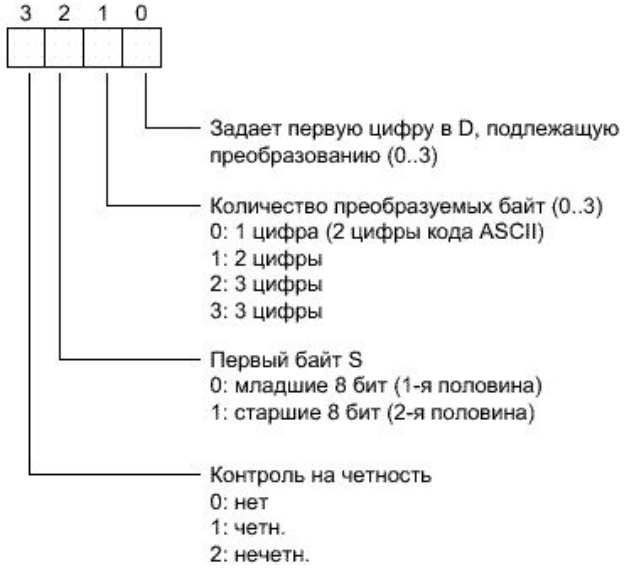
Команды преобразования данных



Differentiated variant available.

Di: 0023



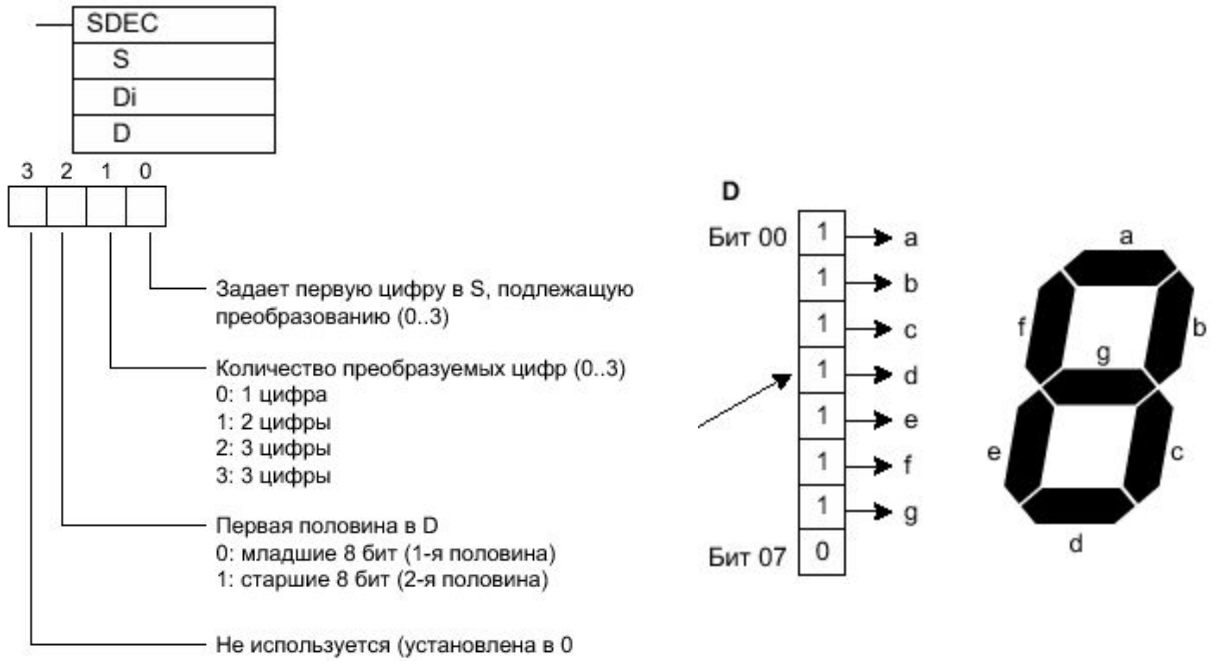


HEX – преобразование из кодов ASCII в 16-ричное число
Когда условие исполнения = 1, HEX преобразует указанный бит кода ASCII из слова источника в 16- ричный эквивалент и помещает их в D.

Можно преобразовать до 4 кодов ASCII, начиная с первого байта S. Преобразованное 16- ричное значение далее пересылается в D, начиная от заданной цифры.

Если задано больше цифр чем осталось в D, остальные цифры будут браться, начиная с начала D. Цифры в D, которые не принимают преобразованных данных, не изменяются.

Команды преобразования данных

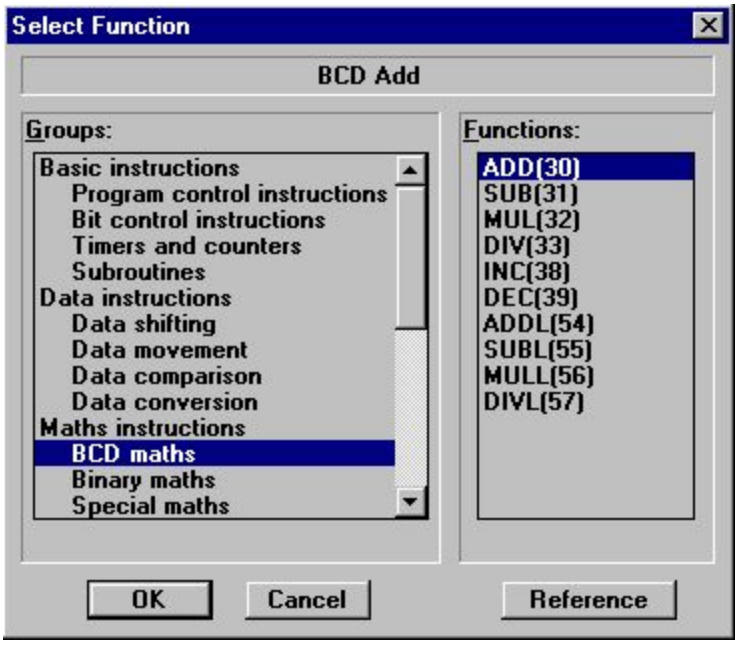


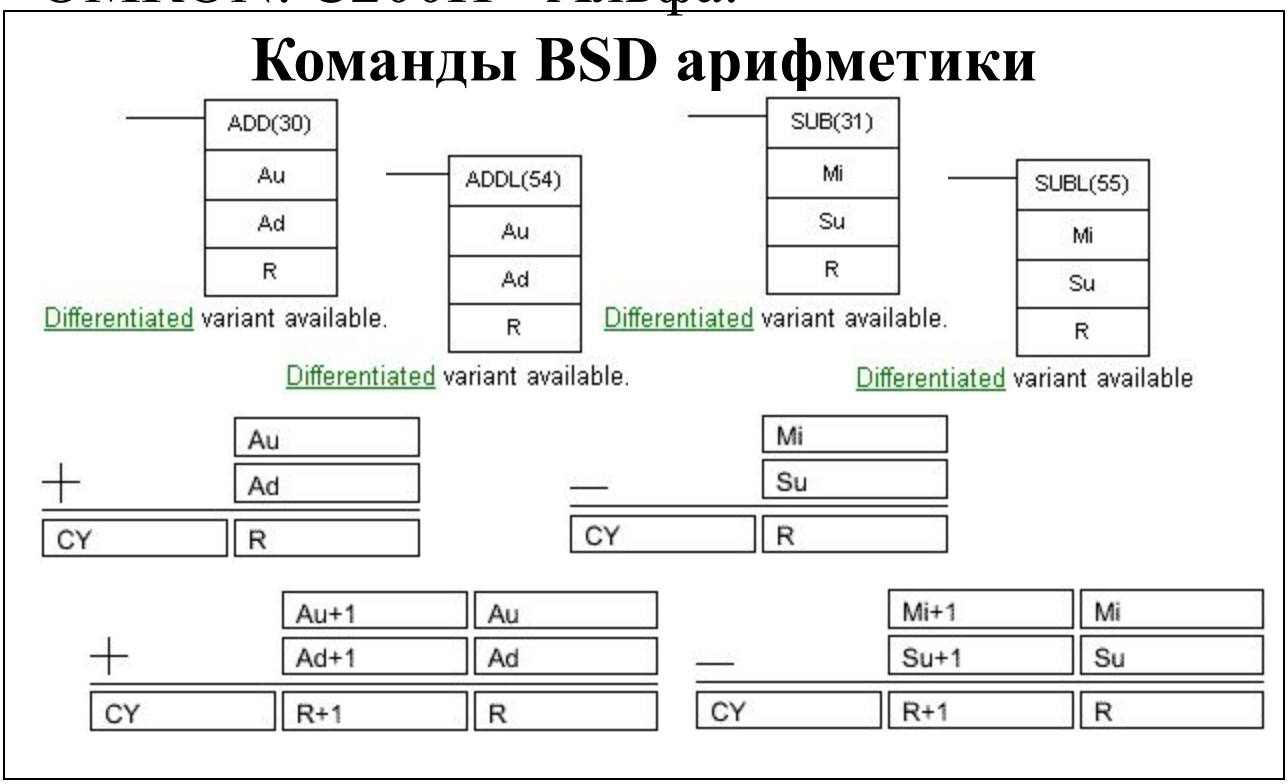
SDEC – преобразование в коды 7- сегментного индикатора

Когда условие исполнения = 1, SDEC преобразует указанные цифры слова S в 8-битовый эквивалент – код 7- сегментного индикатора и помещает его в слово приемника, начинающегося с D.

Исходные данные					Преобразованные коды (сегменты)								Индикация
Цифра	Биты				-	g	f	e	d	c	b	a	
0	0	0	0	0	0	0	1	1	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	0	0	0	1
2	0	0	1	0	0	0	0	1	0	0	1	1	2
3	0	0	1	1	0	0	0	1	0	0	1	1	3
4	0	1	0	0	0	0	1	1	0	1	0	0	4
5	0	1	0	1	0	0	1	1	0	1	0	1	5
6	0	1	1	0	0	0	1	1	0	1	0	1	6
7	0	1	1	1	0	0	1	1	0	1	1	1	7
8	1	0	0	0	0	0	1	1	1	0	0	0	8
9	1	0	0	1	0	0	1	1	1	0	0	1	9
A	1	0	1	0	0	1	0	0	0	0	0	1	A
B	1	0	1	1	0	1	0	0	0	1	1	0	b
C	1	1	0	0	0	1	0	0	0	1	1	1	C
D	1	1	0	1	0	1	1	0	0	0	0	0	d
E	1	1	1	0	0	1	1	0	0	0	0	1	E
F	1	1	1	1	0	1	1	0	0	1	1	0	F

Команды BSD арифметики





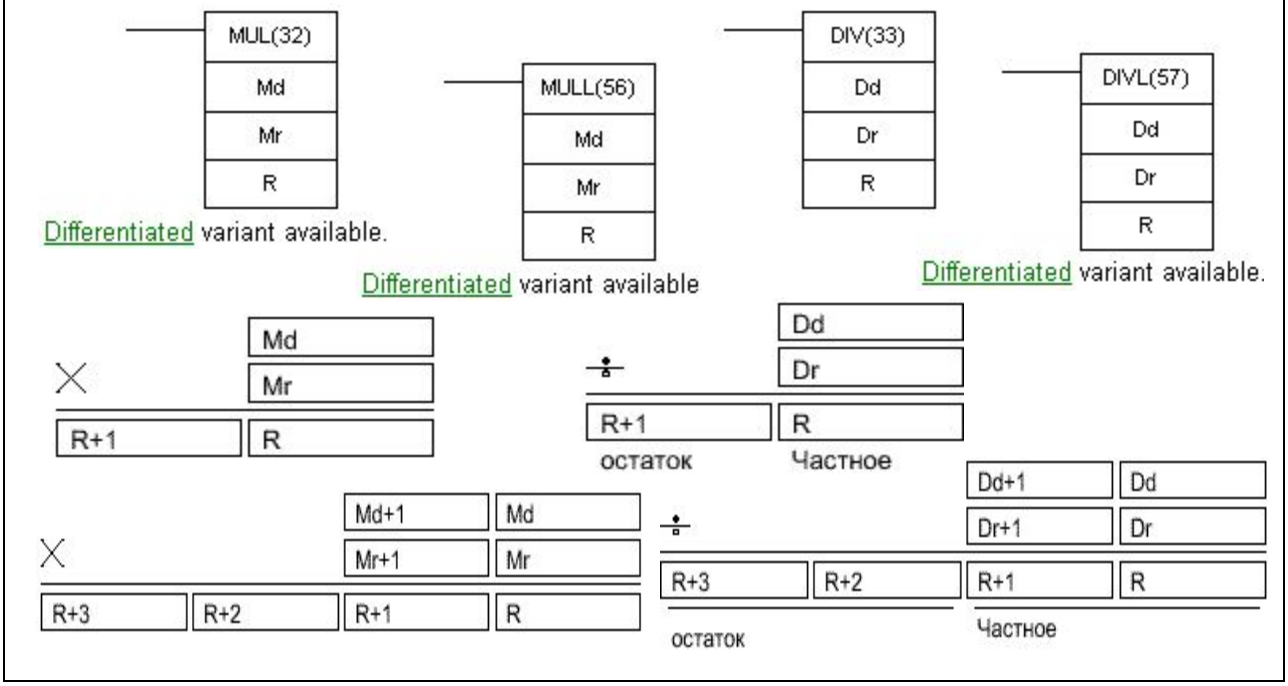
ADD – сложение двоично- десятичных чисел
Когда условие исполнения =1, ADD складывает содержимое слов Au и Ad и помещает результат в R. CY установится в 1, когда результат больше 9999.

ADDL – сложение двоично- десятичных чисел двойной длины
Когда условие исполнения =1, ADDL складывает восьмиразрядное число из Au и Au+1 и восьмиразрядное число из Ad и Ad+1 и помещает результат в R и R+1. CY установится в 1, когда результат больше 9999 9999.

SUB – вычитание двоично- десятичных чисел
Когда условие исполнения =1, SUB вычитает содержимое слова Su из Mi и помещает результат в R. Если результат отрицательный CY установится в 1 и в R будет помещено дополнение до 10 фактического результата.

SUBL – вычитание двоично- десятичных чисел двойной длины
Когда условие исполнения =1, SUBL вычитает восьмиразрядное BCD число Su и Su+1 из восьмиразрядного BCD числа Mi и Mi+1 и помещает результат в R и R+1. Если результат отрицательный CY установится в 1 и в R будет помещено дополнение до 10 фактического результата.

Команды BSD арифметики



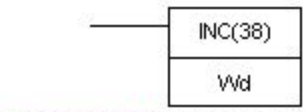
MUL – умножение двоично – десятичных чисел
Когда условие исполнения =1, MUL умножает двоично – десятичное число из Md на двоично – десятичное число из Mr и помещает результат в R и R+1.

MUL – умножение двоично – десятичных чисел двойной длины
Когда условие исполнения =1, MULL умножает 8-разрядное двоично – десятичное число из Md и Md+1 на двоично – десятичное число в Mr и Mr+1и помещает результат в R … R+3.

DIV – деление двоично – десятичных чисел
Когда условие исполнения =1, двоично – десятичное число в Dd делится на двоично – десятичное число в Dr и результат помещается в R и R+1: частное в R, остаток в R+1.

DIVL – деление двоично – десятичных чисел двойной длины
Когда условие исполнения =1, MULL умножает 8-разрядное двоично – десятичное число в Dd и Dd+1 делится на двоично – десятичное число Dr и Dr+1 и результат помещается в R … R+3: частное R, R+1 и остаток в R+2, R+3.

Команды BSD арифметики



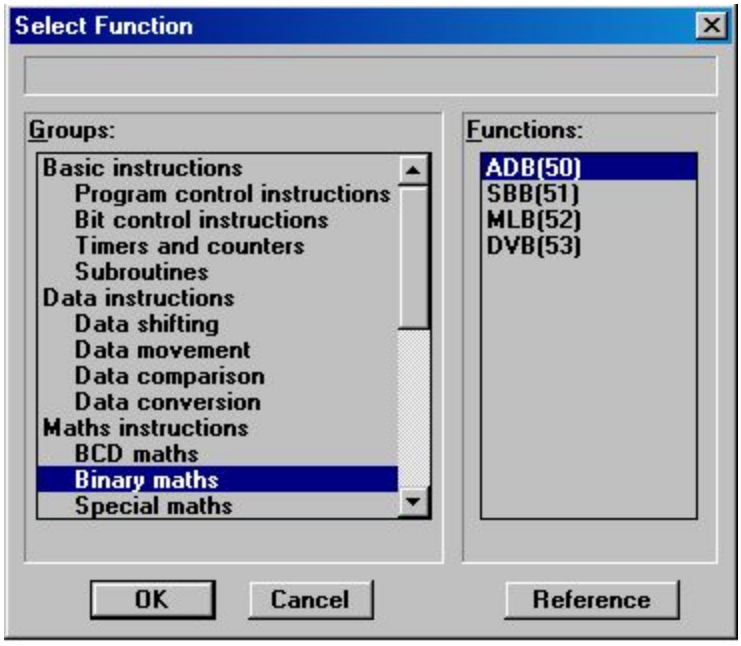
Differentiated variant available.

Differentiated variant available.

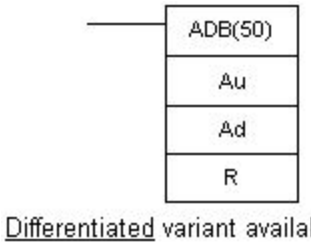
INC – инкремент двоично- десятичного числа
Когда условие исполнения =1, INC инкрементирует (увеличивает на 1) содержимое слова Wd, без воздействия на флаг переноса CY.

DEC – декремент двоично- десятичного числа
Когда условие исполнения =1, DEC декрементирует (уменьшает на 1) содержимое слова Wd, без воздействия на флаг переноса CY.

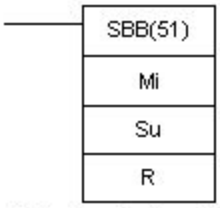
Команды двоичной арифметики



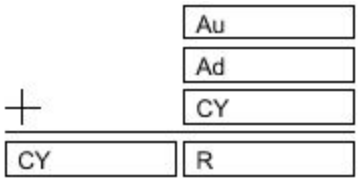
Команды двоичной арифметики

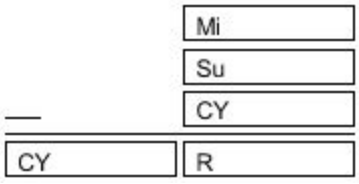


Differentiated variant available.



Differentiated variant available.





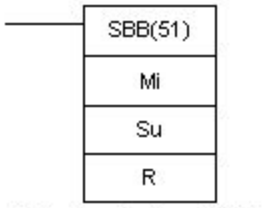
ADB – сложение двоичных чисел
Когда условие исполнения =1, ADB складывает содержимое слов Au, Ad и CY и помещает результат в R. CY установится в 1, когда результат больше FFFF.

ADB может также складывать двоичные числа со знаком. Флаги переполнения SR254.04 и SR254.05 указывают, что результат перешел за верхнюю или нижнюю границы 16 – разрядного двоичного числа со знаком.

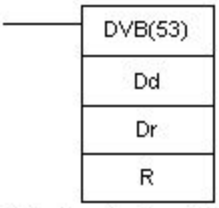
SBB – вычитание двоичных чисел
Когда условие исполнения =1, SBB вычитает содержимое слова Su и перенос CY из Mi и помещает результат в R. Если результат отрицательный CY установится в 1 и в R будет помещено дополнение до 2 фактического результата.

SBB можно также использовать для вычитания двоичных чисел со знаком. Флаги переполнения SR254.04 и SR254.05 указывают, что результат перешел за верхнюю или нижнюю границы 16 – разрядного двоичного числа со знаком.

Команды двоичной арифметики



Differentiated variant available.



Differentiated variant available

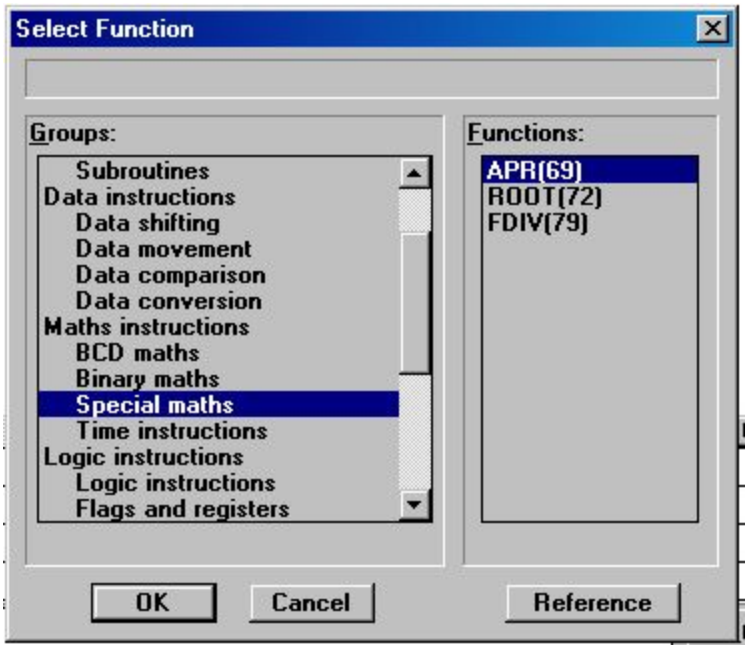
MLB – умножение двоичных чисел
Когда условие исполнения =1, MUL умножает содержимое Mr на содержимое из Md и помещает 4 младшие цифры результата в R и 4 старшие цифры в R+1.

MLB нельзя использовать для умножения двоичных данных со знаком. Вместо этого нужно использовать команду MBS.

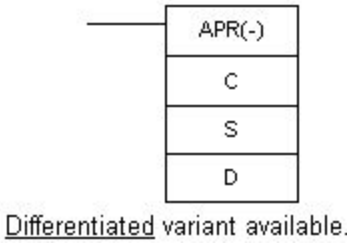
DVD – деление двоичных чисел
Когда условие исполнения =1, DVD делит содержимое Dd на число в Dr и результат помещается в R и R+1: частное в R, остаток в R+1.

DVD нельзя использовать для деления двоичных данных со знаком. Для этого нужно использовать команду DBS.

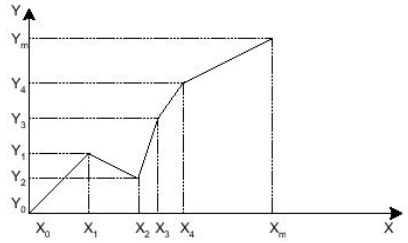
Специальные математические команды



Специальные математические команды



Слово	Координата
C+1	X_m
C+2	Y_0
C+3	X_1
C+4	Y_1
C+5	X_2
C+6	Y_2
..	
C+(2m+1)	X_m
C+(2m+2)	Y_m



APR – математические вычисления

Когда условие исполнения = 1, операция выполняемая APR зависит от значения слова управления С:

-если С= # 0000 или # 0001, APR вычисляет SIN или COS двоично- десятичного значения в S, в котором задан угол, с точностью до одной десятой градуса;

- если С задано как адрес, то APR вычисляет F(x) функцию, введенную ранее, и имеющую начало в слове С.

Функция – это серия отрезков прямой (которыми аппроксимируется кривая), заданная оператором. Двоично – десятичное или 10- ричное значение S задает x.

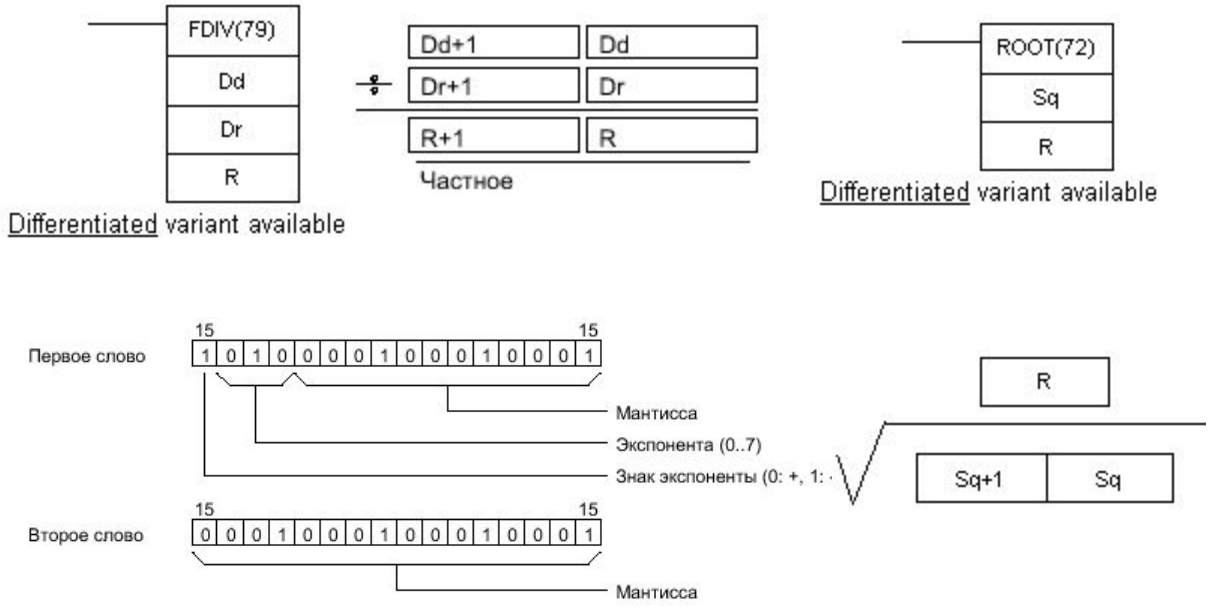
Линейная аппроксимация

Линейная аппроксимация APR задана, когда в С задан адрес памяти. Слово С является первым словом блока памяти, содержащего параметры линейной аппроксимации.

Содержимое слова С задает количество отрезков линий в аппроксимации, и вид здания ввода/ вывода – двоично – десятичный или двоичный. Биты 00 – 07 содержат количество сегментов минус 1, как двоичные данные. Биты 14 и 15 определяют, соответственно, формы ввода/ вывода: 0 – двоично – десятичная,1- двоичная.

Координаты m+1 конечных точек, которые определяют m отрезков, задаются как показано в таблице. Все координаты вводятся в двоичной форме с наименьшего X (X1) до наибольшего X(Xm). X0 = 0000, и его вводить не нужно.

Специальные математические команды



FDIV – деление чисел с плавающей точкой

Для представления значений с плавающей точкой 7 младших цифр используются в качестве мантиссы, а старшая цифра служит показателем степени, как показано на рисунке. Мантисса выражена величиной, меньшей чем 1, т.е до 7 десятичных позиций.

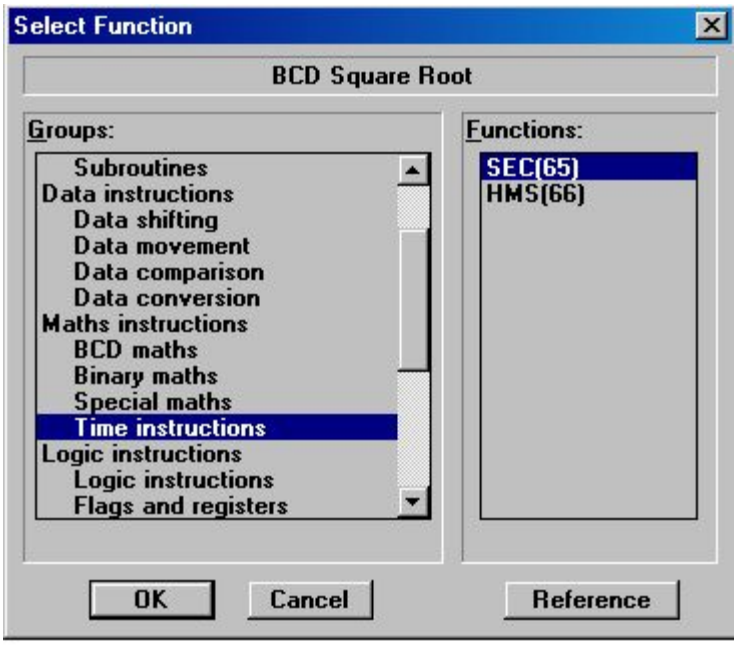
Когда условие исполнения =1, FDIV делит число в Dd и Dd+1 на содержимое Dr и Dr+1 и посылает результат в R и R+1.

ROOT – квадратный корень

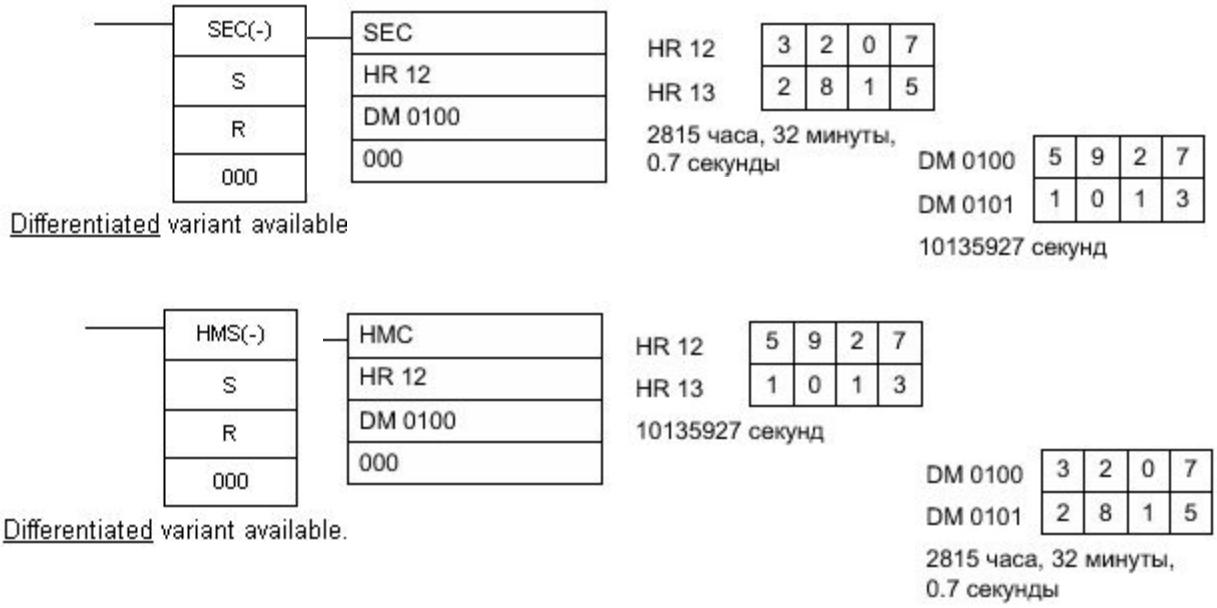
Когда условие исполнения =1, ROOT вычисляет квадратный корень 8- разрядного содержимого Sq+1 и Sq и выдает результат в R. Дробная часть опускается.

Sq и Sq+1 должны находиться в одной области данных.

Команды «реального времени»



Команды «реального времени»



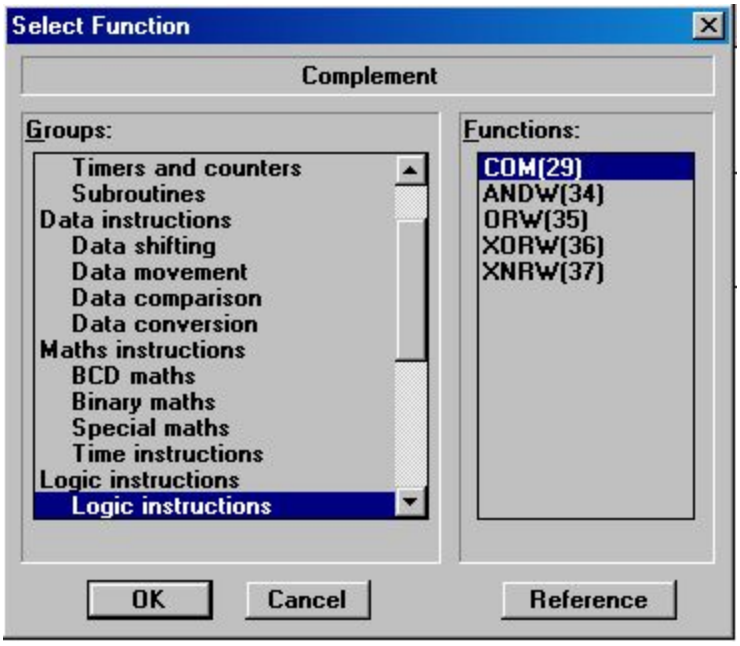
SEC – преобразование часы - в – секунды
SEC служит для преобразования из формата часы/ минуты/ секунды в секунды. В словах источниках секунды расположены в битах 00 – 07, минуты 08 – 15 слова S. Часы расположены в S+1. Максимальное значение 9 999 часов, 59 минут, 59 секунд Результат выдается в R и R+1. Максимальное значение 35 999 999 секунд.

S и S+1 должны находиться в одной области данных.
R и R+1 должны находиться в одной области данных.
S и S+1 должны быть в двоично- десятичном виде и в требуемом формате часы/ минуты/ секунды в секунды.

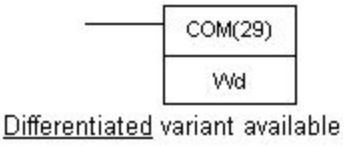
HMS – преобразование секунды - в –часы
HMS служит для преобразования из формата секунд в формат часы/ минуты/ секунды . В словах результата секунды расположены в битах 00 – 07, минуты 08 – 15 слова R. Часы расположены в R+1. Максимальное значение 9 999 часов, 59 минут, 59 секунд
Количество секунд, находящиеся в S и S1, преобразуется в формат часы/ минуты/ секунды и результат выдается в R и R+1.

S и S+1 должны находиться в одной области данных.
R и R+1 должны находиться в одной области данных.
S и S+1 должны быть в двоично- десятичном виде и находиться между 0 и 35 999 999

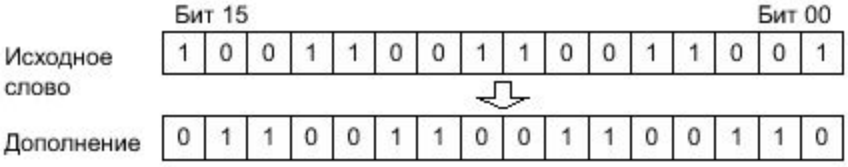
Логические команды



Логические команды



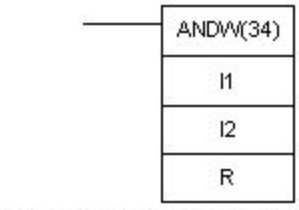
Differentiated variant available



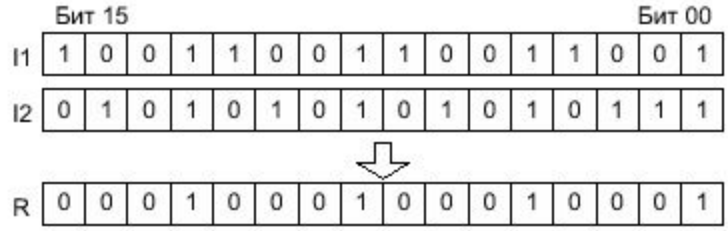
COM - дополнение

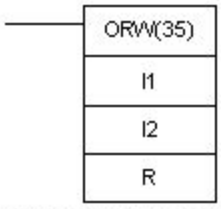
Когда условие исполнения =1, COM очищает в слове Wd биты, находящиеся в состоянии 1 и устанавливает в 1 все биты, находящиеся в состоянии 0.
COM будет постоянно менять значение слова Wd каждый цикл, при условии исполнения =1. При необходимости нужно использовать @COM.

Логические команды

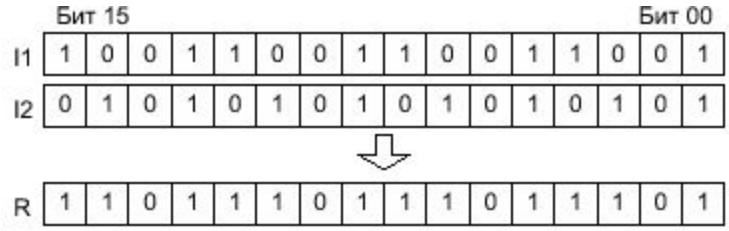


Differentiated variant available.





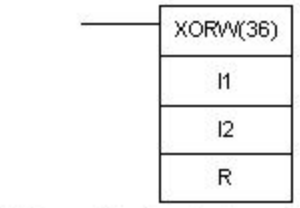
Differentiated variant available



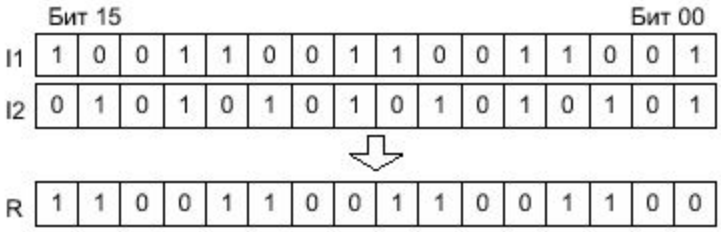
ANDW – логическое И
Когда условие исполнения =1, ANDW производит операцию «ЛОГИЧЕСКОЕ И» с содержимым I1 и I2 побитно и выдает результат в R.

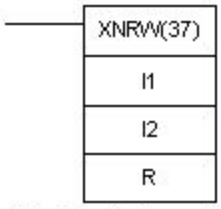
ORW – логическое ИЛИ
Когда условие исполнения =1, ORW производит операцию «ЛОГИЧЕСКОЕ ИЛИ» с содержимым I1 и I2 побитно и выдает результат в R.

Логические команды

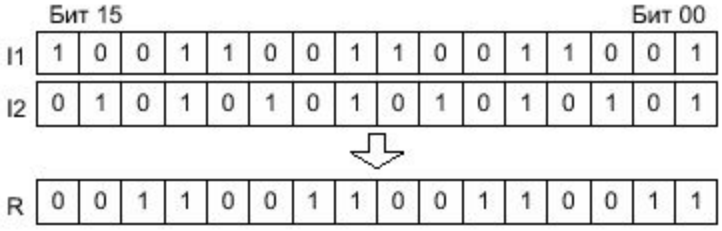


Differentiated variant available.





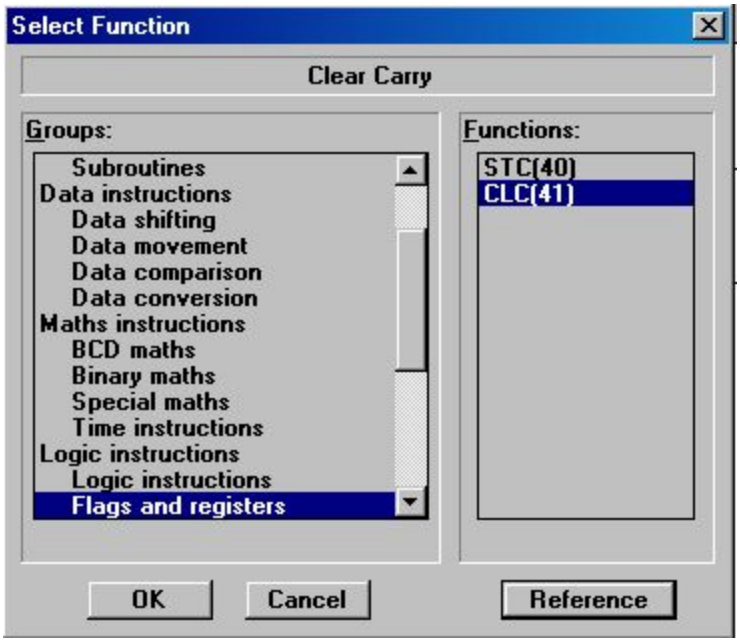
Differentiated variant available.



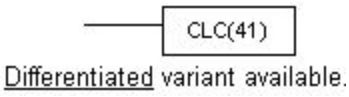
XORW – исключающее ИЛИ
Когда условие исполнения =1, XORW производит операцию «ИСКЛЮЧАЮЩЕЕ ИЛИ» с содержимым I1 и I2 побитно и выдает результат в R.

XNRW – исключающее ИЛИ - НЕ
Когда условие исполнения =1, XNRW производит операцию «ИСКЛЮЧАЮЩЕЕ ИЛИ - НЕ» с содержимым I1 и I2 побитно и выдает результат в R.

Команды флагов и регистров



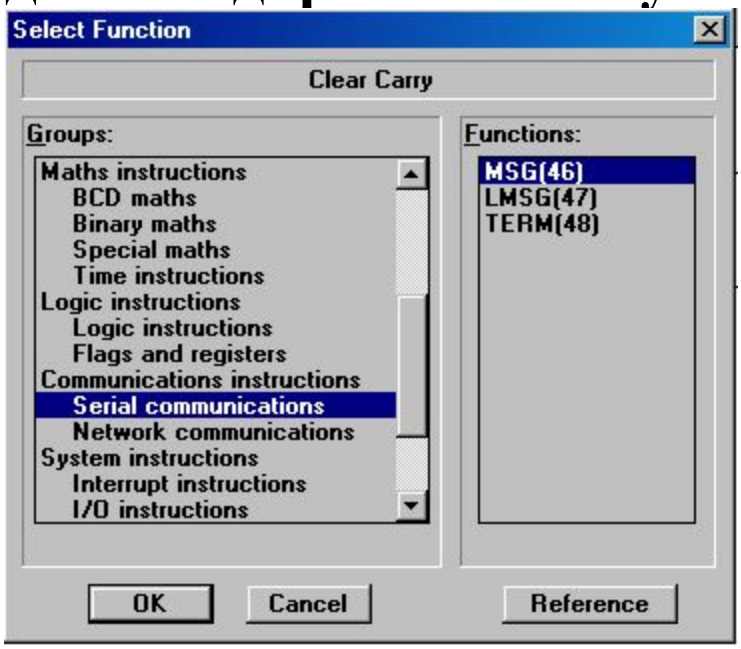
Команды флагов и регистров



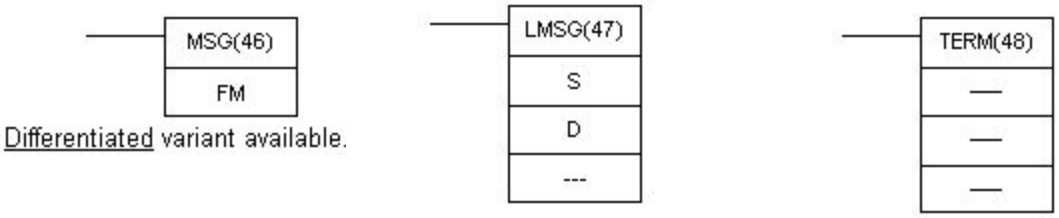
STC – установка флага переноса
Когда условие исполнения =1,STC устанавливает флаг переноса CY (SR25504) в1.

CLC – сброс флага переноса
Когда условие исполнения =1,CLC сбрасывает флаг переноса CY (SR25504) в 0. Кроме того, CY автоматически сбрасывается в 0 при выполнении команды END в конце каждого цикла.

Команды стандартной коммуникации



Команды стандартной коммуникации



MSG – сообщение

Когда условие исполнения =1,MSG читает 8 слов расширенного кода ASCII и индицирует сообщение на программаторе. Индицируемое сообщение может быть длиной до 16 знаков, т.е каждый знак требует 8 бит (2 цифры).

Если длина сообщения меньше 8 слов, его можно прервать в любом месте, вводя «OD». Когда в сообщении встречается «OD», слова больше не читаются, и ячейки памяти, которые нормально были использованы для сообщений, можно использовать для других целей.

LMSG – длинное сообщение

LMSG служит для вывода сообщений на программаторе длиной 32 знака. сообщение должно быть в кодах ASCII , начинаться в S и оканчиваться максимально в S+15. Более короткое сообщение можно получить прервать его в любом месте, вводя «OD».

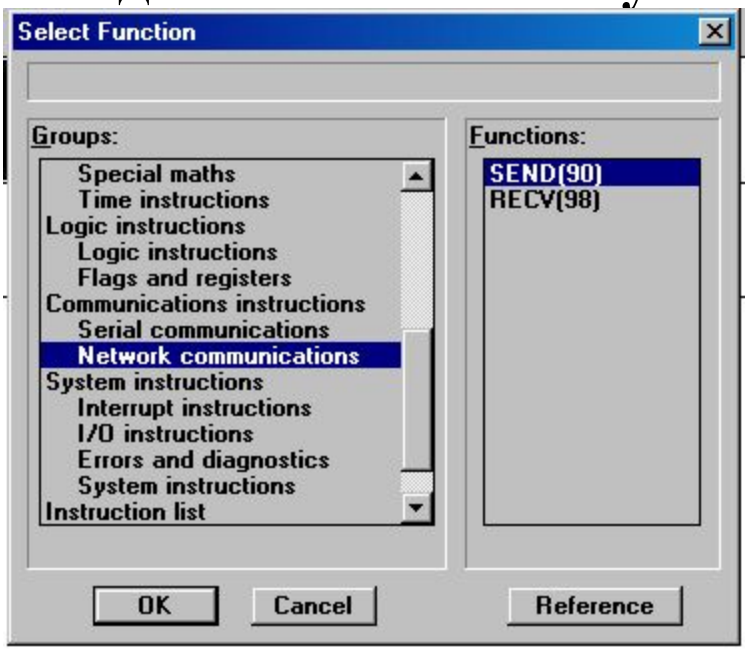
Для вывод сообщения на программаторе, он должен быть установлен в режим «TERMINAL».

TERM – переключение в режим TERMINAL

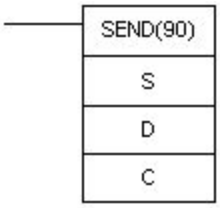
Когда условие исполнения =1, TERM переключает программатор в режим TERMINAL.

Когда секция 6 DIP переключателя на ЦПУ= «ON», программатор можно переключить в режим TERMINAL, установив в 1 бит AR 07.09.

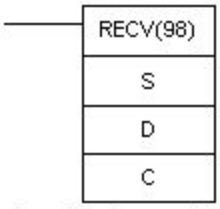
Команды сетевых коммуникаций



Команды сетевых коммуникаций



Differentiated variant available.



Differentiated variant available.

	Номер инструкции	Операнды		
Контроллер	Nr	S	D	C
C200H□-CPU□□-E	90	IR, SR, HR, AR, LR, TC, DM	IR, SR, HR, AR, LR, TC, DM	IR, SR, HR, AR, LR, TC, DM
C200H□-CPU□□-ZE	090	IR, SR, HR, AR, LR, TC, DM, EM	IR, SR, HR, AR, LR, TC, DM, EM	IR, SR, HR, AR, LR, TC, DM, EM

Команды сетевых коммуникаций используются для связи с другими ПЛК, блоками BASIC или управляющими компьютерами, связанными по системе SYSMAC NET, SYSMAC LINK или ETHERNET.

SEND – передача по сети
Когда условие исполнения =1, SEND передает данные, начинающиеся в слове S по адресу указанному в D, на заданный узел системы SYSMAC NET, SYSMAC LINK или ETHERNET.В словах управления, начинающихся с C задаются узел адресата, и другие параметры, приведенные далее.

RECV – прием из сети
Когда условие исполнения =1, RECV принимает данные, начинающиеся в слове S из узела системы SYSMAC NET, SYSMAC LINK или ETHERNET в слова, .В словах управления, начинающихся с C задаются количество принимаемых слов, узел адресата, и другие параметры, приведенные далее.

Параметры управления сетевыми коммуникациями

Для системы ETHERNET

Слово	Биты 00..07	Биты 08..15
C	Количество слов (0..1000, 4-разрядные шестнадцатиричные, например, 0000 $_{16\text{-рич}}$..03E8 $_{16\text{-рич}}$)	
C+1	Предельное значение времени ответа (0.1..25.5 секунд с дискретой задания 0.1 с, 2-разрядные 16-ричные без десятичной точки, например, $01_{16\text{-рич}}$..$FF_{16\text{-рич}}$). По умолчанию $00_{16\text{-рич}}$ (2.2 секунд).	Биты 08..11: Число повторов (0..15 16-ричных, например, $0_{16\text{-рич}}$..$F_{16\text{-рич}}$ Бит 12: 1: Косвенная адресация, 0: Прямая адресация Бит 13: 1: Ответ не возвращается, 0: Ответ возвращается Бит 14: 1: Уровень 0, 0: Уровень 1 Бит 15: Установить в 1
C+2	Узел адресата (0..127, 2-разрядные 16-ричные, например, $00_{16\text{-рич}}$..$7E_{16\text{-рич}}$)*	Биты 08..12: Адрес блока узла источника. Задайте в $00_{16\text{-рич}}$ Биты 13..15: Задайте = 0.

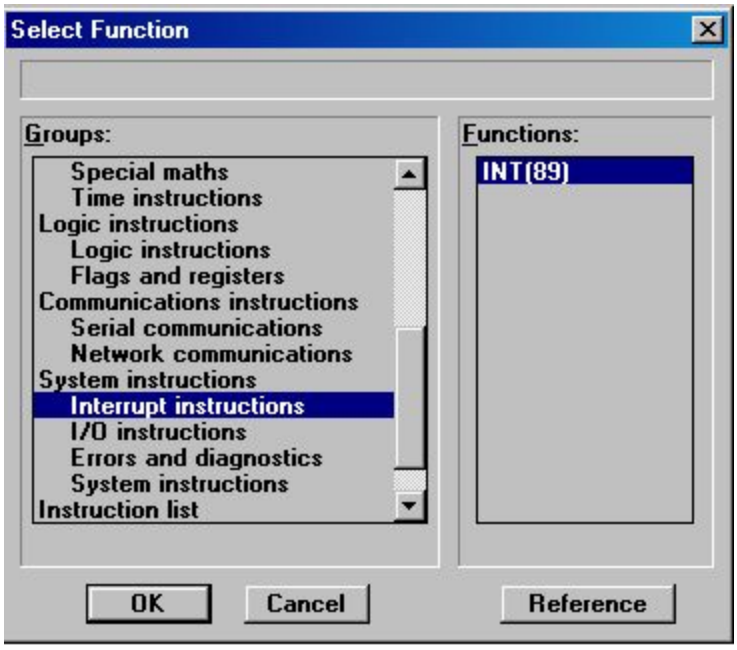
Для системы SYSMAC NET

Слово	Биты 00..07	Биты 08..15
C	Количество слов (0..1000, 4-разрядные шестнадцатиричные, например, 0000 $_{16\text{-рич}}$..03E8 $_{16\text{-рич}}$)	
C+1	Номер сети (0..127, 2-разрядные 16-ричные, например, $00_{16\text{-рич}}$..$7F_{16\text{-рич}}$)*	Бит 14: 1: Уровень 0, 0: Уровень 1 Биты 08..13 и 15: Установить в 0.
C+2	Узел адресата (0..126, 2-разрядные 16-ричные, например, $00_{16\text{-рич}}$..$7E_{16\text{-рич}}$)*	Порт источника NSB: 00 NSU: 01/02

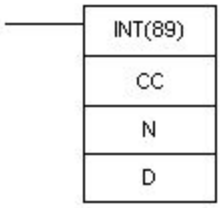
Для системы SYSMAC NET

Слово	Биты 00..07	Биты 08..15
C	Количество слов (0..1000, 4-разрядные шестнадцатиричные, например, 0000 $_{16\text{-рич}}$..03E8 $_{16\text{-рич}}$)	
C+1	Предельное значение времени ответа (0.1..25.4 секунд, 2-разрядные 16-ричные без десятичной точки, например, $00_{16\text{-рич}}$..$FF_{16\text{-рич}}$) Прим.: Предельное значение времени ответа будет равно 2с, если предел задан на $0_{16\text{-рич}}$. Если предел зададать $FF_{16\text{-рич}}$, ограничения времени не будет.	Биты 08..11: Число повторов (0..15 16-ричных,например, $0_{16\text{-рич}}$..$F_{16\text{-рич}}$ Бит 12: Установить в 0. Бит 13: Установить в 0. Бит 14: 1: Уровень 0, 0: Уровень 1 Бит 15: Установить в 1
C+2	Узел адресата (0..62, 2-разрядные 16-ричные, например, $00_{16\text{-рич}}$..$3E_{16\text{-рич}}$)*	Установить в 0

Команды управления прерываниями



Команды управления прерываниями



Differentiated variant available.

Прерывание	C	Функция INT	Комментарии
Входные прерывания от блока входных прерываний 0 (N=000)	000	Маскировать/размаскировать входные прерывания	Биты 00.. 07 слова D указывают входы 00.. 07
	001	Очистить входные прерывания	
	002	Читать текущее состоянии маски	Состояние записано в D
Входные прерывания от блока входных прерываний 1 (N=001)	000	Маскировать/размаскировать входные прерывания	Биты 00.. 07 слова D указывают входы 00..07
	001	Очистить входные прерывания	
	002	Читать текущее состоянии маски	Состояние записано в D
По расписанию (N=004)	000	Задать интервал прерываний	-
	001	Задать время первого прерывания	-
	002	Читать интервал прерываний	-

Значение C	Функция INT
100	Замаскировать все прерывания
200	Размаскировать все прерывания

	Номер инструкции	Операнды		
Контроллер	Nr	C	N	D
		слово состояния	тип прерывания	контрольное слово
C200H□-CPU□□-E	89	# (000, 001, 002, 100, 200)	# (000, 001, 004)	IR, SR, HR, AR, LR, TC, DM, #
C200H□-CPU□□-ZE	089	# (000, 001, 002, 100, 200)	# (000, 001, 004)	IR, SR, HR, AR, LR, TC, DM, EM, #

INT – управление прерываниями

INT осуществляет одну из 11 функций, зависящих от значения C и N. Имеется 6 функций входных прерываний, 3 прерывания по расписанию 2 для маскирования или размаскирования прерываний.

Маскировать/ размаскироовать входные прерывания (N=000 или 001, C=000)

Данная функция служит для маскирования и размаскирования входных прерываний 00…07. Замаскированные прерывания запоминаются. Но игнорируются, и программа прерывания замаскированного входа будет исполняться только после размаскирования. Порядковый номер установленного в 1 бита в D указывает замаскированный вход прерывания.

Очистить входные прерывания(N=000 или 001, C=001)

Поскольку замаскированные прерывания запоминаются, то если не очистить память, то после размаскирования они будут обработаны.

Читать текущее состояние маски (N=000 или 001, C=002)

Данная функция служит для записи текущего значения состояния маски входов прерываний 00…07 в слово D. Если вход замаскирован, соответствующий бит =1.

Задать интервал прерываний (N= 004, C= 000)

Содержимое D (BCD: 0001..9999) умножается на дискрету (1 или 10 мс) для получения интервала.Дискрета времени задается в DM 6622.

Задать время первого прерывания (N=004, C=001)

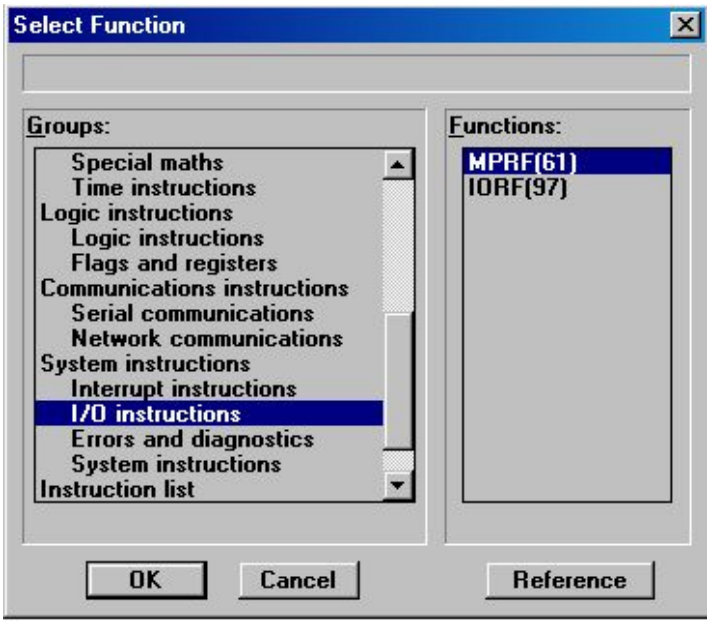
Содержимое D (BCD: 0001..9999) умножается на дискрету (1 или 10 мс) для получения времени первого прерывания по расписанию. Дискрета - в DM 6622.

Читать интервал прерываний (N= 004, C= 002)

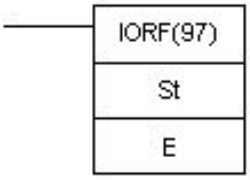
Маскировать/ размаскировать все прерывания (C=100/200)

Параметры D и N не используются.

Команды управления входами/ выходами



Команды управления входами/ выходами

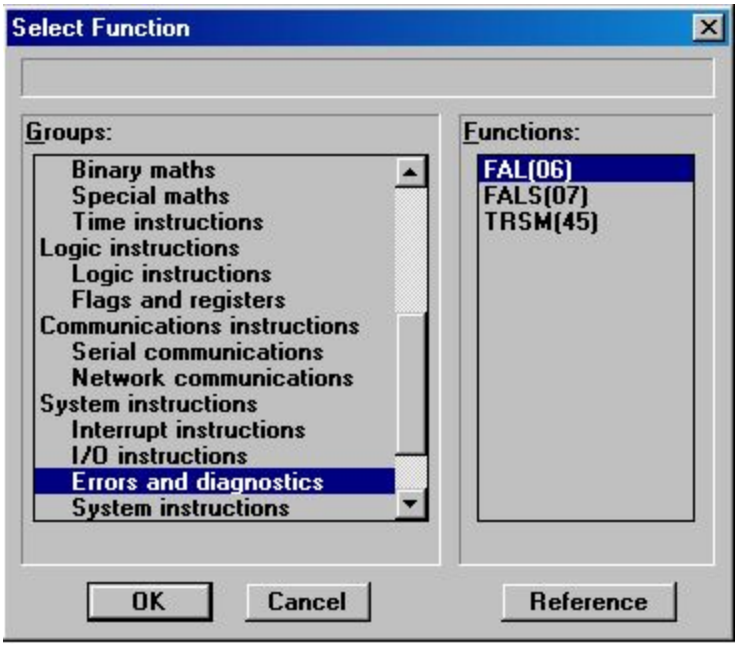


MPRF – обновление модулей «группы 2»
Когда условие исполнения =1, слова, выделенные блокам входов / выходов «группы 2» с номерами St…E будут обновлены. Это производиться в дополнение к нормальному обновлению входов / выходов , которое осуществляется в каждом цикле CPU.
Операнды St… E нельзя задавать адресом, а только номером блока входов / выходов, установленном на модуле.

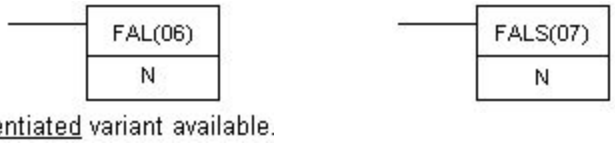
IORF - обновление модулей
Для обновления слов входов / выходов, выделенных панелям ЦПУ и расширения (IR000…IR029 и IR300… IR309), задается первое (St) и последнее (E) слово входов / выходов, подлежащих обновлению, выполняемого в дополнения к циклическому обновлению входов / выходов.
Для обновления слов, выделенных модулям специальных входов / выходов 0…9 (IR100…IR199), задается IR040…IR049, которые используются для идентификации соответствующего блока специальных входов / выходов.
Исполнение IORF не оказывает влияние на содержимое IR040...IR049.

IORF нельзя использовать для обновления слов, выделенных модулям входов / выходов, установленных на ведомых панелях, и модулям «группы 2».

Команды диагностики и ошибок

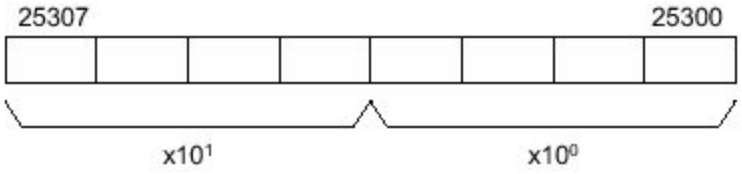


Команды диагностики и ошибок



Differentiated variant available.

Область ошибок FAL



FAL – предупреждение о нефатальной ошибке
FALS – предупреждение о фатальной ошибке
FAL и FALS используются для того, чтобы выдавать номера ошибок при работе, обслуживании и наладке. Когда условие исполнения =1, они выдают номер FAL в биты 00 … 07 SR253. Номер может иметь значения 00…99 и является определителем команд. FAL с определителем 00 используется для сброса SR253.

Когда выполняется FAL с условием 1, мигает индикатор ALARM/ ERROR на передней панели ЦПУ, но операции ПЛК продолжаются.
Когда выполняется FALS с условием 1, индикатор ALARM/ ERROR на передней панели ЦПУ горит, и операции ПЛК прекращаются.

Команды диагностики и ошибок

Флаг	Функция
AR 2515	Бит начала выборки
AR 2514	Бит начала трассировки
AR 2513	Флаг ТРАССИРОВКА
AR 2512	Флаг завершения трассировки

TRSM – выборка памяти трассировки

TRSM служит в программе для пометки мест, куда должны быть записаны специальные параметры в памяти трассировки. Для трассировки можно назначить до 12 битов и до 3 слов.

TRSM управляется не условиями исполнения, а двумя битами в AR.

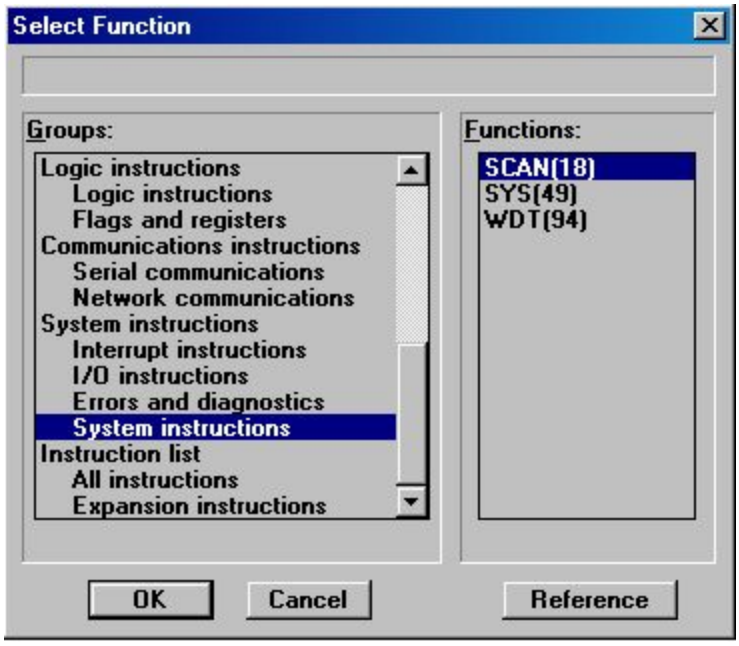
AR25.15 – бит начала выборки. Этот бит устанавливается в 1 для начала процесса задания выборки для трассировки. Бит начала выборки не может устанавливаться из программы.

AR25.14 – бит начала трассировки. Когда он установлен, назначенные данные загружаются в память трассировки. Бит начала трассировки можно устанавливать в 1 как из программы, так и с периферийного устройства. Можно вводить положительное или отрицательное смещение для изменения фактической точки, откуда будет начинаться трассировка.

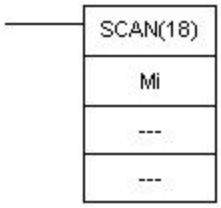
Данные можно записывать несколькими путями. TRSM можно поместить в одном или нескольких местах программы для индикации, где нужно трассировать назначенные данные. Если TRSM не используется, назначенные данные будут трассироваться после исполнения END. Третий метод включает задание интервала таймера с периферийного устройства, так что назначенные данные будут трассироваться через регулярные интервалы времени независимо от времени цикла.

TRSM можно встроить в любое место программы, любое число раз. Данные в памяти трассировки можно просматривать любыми периферийными устройствами.

Системные команды



Системные команды



Differentiated variant available

	Номер инструкции	Операнды
Контроллер	Nr	Mi
		задание
C200H□-CPU□□-E	18	IR, SR, HR, AR, LR, TC, DM, #
C200H□-CPU□□-ZE	018	IR, SR, HR, AR, LR, TC, DM, EM, #

SCAN – время цикла

Команда SCAN служит для задания минимального времени цикла. Mi является минимальным временем цикла, задаваемое с дискретностью 0,1 мс. Например, если Mi=1200, минимальное время цикл будет 120 мс. Возможный диапазон значений 000,0…999,9 мс.

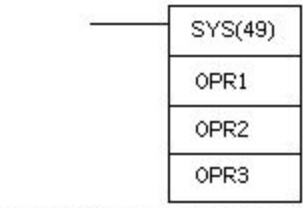
Если текущее время цикла меньше, чем заданное командой SCAN, ЦПУ будет ожидать пока истечет заданное время, прежде чем начать следующий цикл. Если текущее время цикла больше, чем заданное, то оно будет игнорироваться, и программа будет исполняться до завершения.

WDT – обновление сторожевого таймера

Когда условие исполнения =1, WDT заносит задание контрольного времени цикла из параметра DM6618. Значение цикла по умолчанию=120 мс. Контрольное время формируется = 100мс * T.

Задание для сторожевого таймера можно задать от 10 до 640 мс. Команду WDT можно исполнять более одного раза за цикл, но время цикла нельзя увеличить больше чем до 640 мс.

Системные команды



Differentiated variant available.

SYS – системные установки

Дополнительные команды

Код	Мнемоника	Наименование	Функция
17	(@)ASFT	ASYNCHRONOUS SHIFT REGISTER	Создает регистр сдвига, который обменивает содержимое соседних слов, когда одно из слов = 0, а другое нет.
18	(@)SCAN	CYCLE TIME	Задает минимальное время цикла (0..999.0 с)
19	(@)MCMP	MULTI-WORD COMPARE	Сравнивает блок из 16 последовательных слов с другим блоком из 16 последовательных слов.
47	(@)LMSG	32-CHARACTER MESSAGE	Выдает сообщение 32 символа на программатор
48	(@)TERM	TERMINAL MODE	Переключает программатор в режим TERMINAL для операции “отображение состояния клавиатуры”
60	CMPL	DOUBLE COMPARE	Сравнивает два 8-разрядных 16-ричных числа
61	(@)MPRF	GROUP-2 HIGH-DENSITY I/O REFRESH	Обновляет слова входов/выходов, выделенные блокам входов/выходов высокой плотности группы 2.
62	(@)XFRB	TRANSFER BITS	Копирует состояние до 255 заданных бит источника в указанные биты приемника.
63	(@)LINE	COLUMN TO LINE	Копирует столбец битов из 16 последовательных слов в указанное слово.
64	(@)COLM	LINE TO COLUMN	Копирует 16 битов из указанного слова в столбец битов 16 последовательных слов.

Дополнительные команды

Код	Мнемоника	Наименование	Функция
65	(@)SEC	HOURS TO SECONDS	Преобразует данные в форме часы и минуты в форму секунд.
66	(@)HMS	SECOND TO HOURS	Преобразует данные в секундах в данные в виде часов и минут.
67	(@)BCNT	BIT COUNTER	Подсчитывает общее число битов в состоянии 1 в указанном блоке слов.
68	(@)BCMP	BLOCK COMPARE	Определяет, находится ли значение слова в одной из 16 зон (задаются верхней и нижней границами).
69	(@)APR	ARITHMETIC PROCESS	Вычисляет синус, косинус, или линейную аппроксимацию.
87	TTIM	TOTALIZING TIMER	Создает суммирующий таймер
88	ZCP	AREA RANGE COMPARE	Сравнивает слово с зоной, заданной нижней и верхней границами и выдает результат в флаги GR, EQ и LE.
89	(@)INT	INTERRUPT CONTROL	Осуществляет управление прерываниями, маскирование и размаскирование входных прерываний.

Дополнительные команды

Код	Мнемоника	Наименование	Функция
-	7SEG	7-SEGMENT DISPLAY OUTPUT	Преобразует 4- или 8-разрядные данные в формат 7-сегментного индикатора и выводит преобразованные данные.
-	(@)ADBL	DOUBLE BINARY ADD	Складывает два 8-разрядных двоичных чисел двойной длины (со знаком или без знака) и выдает результат в R и R+1.
-	AVG	AVERAGE VALUE	Складывает указанное количество 16-ричных слов и вычисляет среднее значение. Округляет до 4 цифр после запятой.
-	(@)BXF2	EM BANK TRANSFER	Переносит содержимое нескольких последовательных слов источника в несколько последовательных слов приемника. Слова в текущем банке EM можно задать для источнике или приемника.
-	CPS	SIGNED BINARY COMPARE	Сравнивает два 16-битовых (4 цифры) двоичных чисел со знаком и выдает результат во флаги GR, EQ и LE.
-	CPSL	DOUBLE SIGNED BINARY COMPARE	Сравнивает два 32-битовых (8 цифр) двоичных чисел двойной длины со знаком и выдает результат во флаги GR, EQ и LE.

Дополнительные команды

Код	Мнемоника	Наименование	Функция
-	(@)DBS	SIGNED BINARY DIVIDE	Делит одно 16-битовое двоичное число со знаком на другое и выдает 32-битовый двоичный результат со знаком в R+1 и R.
-	(@)DBSL	DOUBLE SIGNED BINARY DIVIDE	Делит одно 32-битовое двоичное число со знаком на другое и выдает 64-битовый двоичный результат со знаком в R+3..R
-	DSW	DIGITAL SWITCH INPUT	Вводит 4- или 8-разрядные двоично-десятичные данные с кодового колеса.
-	(@)EMBC	SELECT EM BANK	Изменяет текущий банк EM на номер заданного банка.
-	(@)FCS	FCS CALCULATE	Проверяет ошибки в данных, переданных в запросе при связи с верхним уровнем.
-	FPD	FAILURE POINT DETECT	Ищет сбои в блоке команд.
-	(@)HEX	ASCII-TO-HEXADECIMAL	Преобразует данные ASCII в 16-ричные.
-	HKY	HEXADECIMAL KEY INPUT	Вводит до 8 цифр 16-ричных данных с клавиатуры 16 клавиш.

Дополнительные команды

Код	Мнемоника	Наименование	Функция
-	(@)IEMS	INDIRECT EM ADDRESSING	Изменяет адерата косвенной адресации DM (*DM) на DM или указанный банк EM. Данная команда может использоваться для изменения текущего банка EM.
-	(@)IORD	SPECIAL I/O UNIT READ	Передает данные из памяти указанных блоков специальных входов/выходов в слова ПК.
-	(@)IOWR	SPECIAL I/O UNIT WRITE	Передает данные из слов ПК в памяти указанных блоков специальных входов/выходов.
-	(@)MAX	FIND MAXIMUM	Находит максимальное значение в указанной области данных и выдает это значение в другое слово.
-	(@)MBS	SIGNED BINARY MULTIPLY	Умножает два 16-битовых двоичных числа со знаком и выдает 8-разрядный двоичный результат со знаком в R+1 и R.
-	(@)MBSL	DOUBLE SIGNED BINARY MULTIPLY	Умножает два 32-битовых двоичных числа двойной длины со знаком и выдает 16-разрядный двоичный результат со знаком в R+3..R.
-	(@)MIN	FIND MINIMUM	Находит минимальное значение в указанной области данных и выдает это значение в другое слово.
-	MTR	MATRIX INPUT	Вводит данные с матрицы 8 точек входа x 8 точек выхода.

Дополнительные команды

Код	Мнемоника	Наименование	Функция
-	(@)NEG	2'S COMPLEMENT	Преобразует 4-разрядное 16-ричное содержимое источника в дополнение до 2 и выдает в результат в R.
-	(@)NEGL	DOUBLE 2'S COMPLEMENT	Преобразует8-разрядное16-ричное содержимое слов источника в дополнение до 2 и выдает в результат в R и R+1.
-	(@)PID	PID CONTROL	Осуществляет ПИД-регулирование, основанное на предварительно заданных параметрах.
-	(@)PMCR	PROTOCOL MACRO	Вызывает и выполняет заданный протокол связи, который зарегистрирован в панели связи.
-	(@)RXD	RECEIVE	Получает данные через порт связи
-	(@)SBBL	DOUBLE BINARY SUBTRACT	Вычитает одно 8-разрядное двоичное число (со знаком или без) из другого и выдает результат в R+1 и R.
-	(@)SCL	SCALING	Осуществляет масштабирование вычисляемой величины.

Дополнительные команды

Код	Мнемоника	Наименование	Функция
-	(@)SRCH	DATA SEARCH	Ищет в заданной зоне памяти указанные данные. Выдает адрес (адреса) слов, которые содержат данные.
-	(@)STUP	CHAMGE RS-232C SETUP	Изменение установочных параметров указанного порта.
-	(@)SUM	SUM	Подсчитывает сумму содержимого слов в указанной зоне памяти.
-	(@)TKY	TEN KEY INPUT	Вводит 8 двоично-десятичных цифр из клавиатуры на 10 клавиш.
-	(@)TXD	TRANSMIT	Посылает данные через порт связи
-	(@)XDMR	EXPANSION DM READ	Содержимое указанного числа слов области расширенных DM читаются и выдаются в слова приемника ПК.
-	(@)XFR2	EM BLOCK TRANSFER	Переносит содержимое нескольких последовательных слов источника в несколько последовательных слов приемника. Для источника или приемника можно задавать слова в любом действующем банке EM. Команду можно использовать для изменения текущего банка.
-	ZCPL	DOUBLE AREA RANGE COMPARE	Сравнивает 8-разрядное значение с зоной, заданной нижней и верхней границами и выдает результат в флаги GR, EQ и LE.