Обзор программных блоков ПЛК

В ПЛК есть множество виртуальных реле, таймеров, счётчиков. У всех у них есть контакты “a” - нормально разомкнутые и “b” - нормально замкнутые.

Входные (X) и выходные (Y) реле.

Главный блок ПЛК содержит входные и выходные реле пронумерованные в восьмеричной системе X000-X007, X010-X017... и Y000-Y007, X010-X017... .
Первая цифра обозначения указывает на блок, в котором расположен вход или выход. 0 - главный блок, 1-7 возможный номер расширения. Например, в первом расширении входы/выходы будут начинаться с X100/Y100.

Вспомогательное реле M

Вспомогательное реле не овеществлено в ПЛК, получает управление только на информационном уровне, тоесть из программы пользователя, программы управления ПЛК или по командам, принятым через интерфейс Modbus. Понятие вспомогательного реле близко к понятию флагов или семафоров в языках программирования высокого уровня. Часть вспомогательных реле имеют функцию энергонезависимого хранения своего сотояния, тоесть длительные перерывы в питании ПЛК не нарушат сотояния установленного программой во время нормального функционирования. Вспомогательные реле с номерами начиная с 8000 имеют специальные служебные назначения: они позволяют управлять работой ПЛК и являются источниками информации о исполнении программы или функционирования ПЛК в общем для программы пользователя.

Статус S

Этот программный блок имеет такие же свойства, как и вспомогательное реле, но используется для управления ветвлением в программе.

Счётчик C

Счётчик считает количество появлений управляющего сигнала на своём входе.
Делятся на:
- 16 разрядные с положительным направлением счёта, диапазон значений 0..32767
- 32 разрядные с положительным/отрицательным направлением счёта, диапазон значений -2,147,483,648..+2,147,483,647
- высокоскоростные 32-разрядные с положительным/отрицательным направлением счёта, диапазон значений -2,147,483,648..+2,147,483,647

Первые 2 типа счётчиков используются только для счёта внутри программы, соответственно, учитывая цикличность исполнения программы, период цикла 10-50 мс и тот факт, что для повторной подачи управляющего сигнала его перед этим надо снять, минимальный период подачи сигнала на программные счётчики составляет 20-100 мс или 10-50 Гц для периодических сигналов.
Последний тип счётчиков не зависит от программного цикла и имеет электрические входы, в качестве которых используются часть дискретных входов. Максимальная частота счёта — 200 кГц. Возможны режимы счёта: простой на увеличение, с указанием направления и двухфазный.

Таймер T

Таймер - это счётчик, значение которого, если он запущен, увеличивается каждые 1, 10 или 100 мс по выбору пользователя. По достижении заданного значения на его выходе устанавливается высокий уровень.
Таймеры T100-T199 - накапливающие. После исчезновения управляющего сигнала на входе их значение не обнуляется и при повторной подаче управляющего сигнала они продолжают отсчёт с того же значения, что и было до отключения с периодом 100 мс. Остальные таймеры обнуляют своё значение при снятии управляющего сигнала.

Регистры данных D, FD

Регистры предназначены для хранения данных. Разрядность каждого регистра составляет 16 бит. Два регистра можно объединять в регистровую пару и использовать в вычислениях как 32-разрядное число. Числа в представлении с плавающей точкой хранятся в конроллере в регистровых парах. Регистры с префиксом D предназначены для оперативного хранения информации, необходимой для программы. Часть из них хранятся в ОЗУ контроллера, часть в ОЗУ, запитанном от литиевой батарейки. Последние сохраняют свои значения после обесточивания контроллера.
Регистры FD имеют такие же свойства при чтении из них.  Они расположены во флэш памяти контроллера, не зависят от какого-либо электропитания, но это накладывает следующие ограничения:

- запись нельзя осуществить командами, предполагающими помещение результата выполнения в регистр общего назначения. В тоже время по интерфейсу Modbus эти регистры модифицируются так же, как и обычные.
- запись происходит медленнее, чем в регистр общего назначения
- существует лимит на количество циклов записи в каждую ячейку памяти

Константы K, H

Константы используются как аргументы комманд. Префикс K соотвествует десятичным константам, H — шестнадцатиричным.

Указатели P, I

Указатели используются для указания места для перехода при ветвлении программы. Указатели с префиксом I используются для обозначения начала процедур обработки прерываний.

Введение в языки программирования LD и IL

Применяемый для программирования данного ПЛК язык ЛД изначально был предназначен для описания устройств основанных на релейной логике. С появлением ПЛК и заменой ими релейной логики аргументом для его принятия как их основного языка программирования стало то, что многим специалистам не придётся переучиваться. Используя аналогии с электрическими устройствами, им легко будет перейти на новое оборудование. Хотя этот аргумент уже не является таким актуальным, поскольку в ПЛК зачастую за этим графическим представлением существует программа на каком-нибудь более «привычном» (для людей связанным с программированием) языке и исполнение проходит последовательно, в отличие от релейных схем, где все устройства работают параллельно, что нарушает аналогию,  язык ЛД по сей день является традиционным для ПЛК. Его достоинством является то, что алгоритм упраления, который можно было бы реализовать в релейной логике, может быть описан на нём без использования каких-либо ключевых слов языка программирования контроллера, только графическими средствами языка. В то же время, поскольку системы управления сегодня не ограничиваются дискретными сигналами, в рассматриваемом ПЛК реализация языка ЛД позволяет интегрировать в программу комманды по обработке данных, будь то математические или связанные с пересылкой или обработкой двоичных данных. На самом же деле вся программа для ПЛК состоит из комманд на языке IL (Instruction List). Это преобразование происходит при компиляции программы на языке ЛД. Специалисты, которым больше по душе традиционные для микропроцессорных систем языки программирования, могут сразу писать управляющую программу для ПЛК с использованием мнемокода языка ИЛ.

Язык ЛД — это язык правил. Его название связано с визуальной аналогией, возникающей при просмотре программы на нём, с лестницей, у которой есть 2 шины и перекладины между ними. Левая из шин ассоциируется с шиной питания для реле и электромагнитных актуаторов. Каждая перекладина — это правило. Правая шина — исполнительные механизмы, реле, подключенные к общей шине источника питания.

Правило считается работающим, если прослеживается хоть один путь от мнимого источника питания слева до исполнительного механизма справа, и тогда на исполнительный механизм подаётся сигнал разрешения работы.

В дальнейшем при описании методов программирования будут приводиться комманды языка котроллера, а также их графические и электрические аналогии, где это применимо.

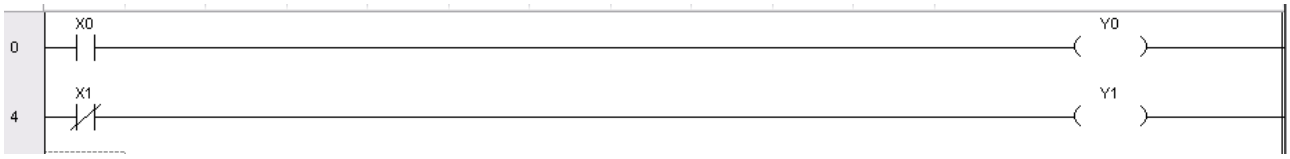
Основные инструкции.

LD, LDI, OUT

| мнемоническое обозначение | графическое обозначение | функция | применимые устройства |
|---|---|---|---|
| LD (LoaD) | ---\| \|--- | загрузка состояния входа или другого дискретного устройства, нормально разомкнутые контакты | X, Y, M, S, T, C, Dn.m, FDn.m |
| LDI (LoaD Inverse) | ---\|/\|--- | загрузка состояния входа или другого дискретного устройства с инверсией, нормально замкнутые контакты | X, Y, M, S, T, C, Dn.m, FDn.m |
| OUT (OUT) | ---( )--- | подача управляющего сигнала на исполнительное устройство | X, Y, M, S, T, C, Dn.m, FDn.m |

Комманда LD загружает состояние битовых устройств, таких как входы, выходы ПЛК, вспомогательные катушки, выходы таймеров и счётчиков. При считывании состояния входов высокий логический уровень, как результат исполнения команды, будет в случае, если контакты, подключённые ко входу замкнуты на общий проводник входов. Команда LDI возвратит логическую 1 если контакты на входе разомкнуты или считываемый бит равен 0. Команда OUT приводит в действие исполнильный механизм, указанный в её аргументе, если результат выполнения предыдущей операции есть логическая 1.

Пример:



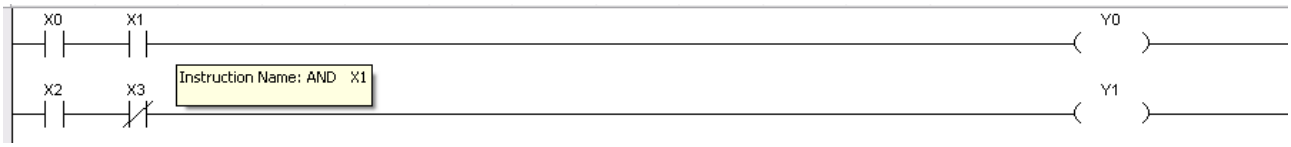

Приведённая выше программа считывает состояние входа X0 и, если контакты на этом входе были замкнуты, приводит в действие выход Y0. Таким образом состояние входа X0 без изменнеий передаётся на выход Y0. На выход Y1 передаётся проинвертированное состояние входа X1. Будучи преобразованной в инструкции контроллера, эта программа выглядит следующим образом.

```
LD    X0
OUT   Y0
LDI   X1
OUT   Y1
```

Для понимания следует представить, что результат комманды LD загружается в аккумулятор (регистр хранения данных в ранних микропроцессорах, в котором хранились аргументы комманд и в который сохранялся результат их выполнения),а  затем команда OUT выводит его в указанный порт.

AND, ANI

| мнемоническое обозначение | графическое обозначение | функция | применимые устройства |
|---|---|---|---|
| AND (AND) | ---\| \|------\| \|--- | загрузка состояния входа или другого дискретного устройства,  применение к полученному значению и результату выполнения предыдущей комманды логического И; последовательное соединение нормально разомкнутых контактов | X, Y, M, S, T, C, Dn.m, FDn.m |
| ANI (ANd Inverse) | ---\| \|------\|/\|--- | загрузка состояния входа или другого дискретного устройства,  применение к инвертированному полученному значению и результату выполнения предыдущей комманды логического И; последовательное соединение нормально разомкнутых контактов | X, Y, M, S, T, C, Dn.m, FDn.m |




Пример:

```
LD    X0
AND   X1
OUT   Y0
LD    X2
ANI   X3
OUT   Y1
```

При замкнутых контактах на входах X0 и X1 правило, включающее Y0, будет работать.
При замкнутых контактах на входе X2 и разомкнутых на X3 правило, включающее Y1, будет работать.

OR, ORI

| мнемоническое обозначение | графическое обозначение | функция | применимые устройства |
|---|---|---|---|
| OR (OR) | ---\| \|---\|<br>---\| \|---\| | загрузка состояния входа или другого дискретного устройства, применение к полученному значению и результату выполнения предыдущей комманды логического ИЛИ; параллельное соединение нормально разомкнутых контактов | X, Y, M, S, T, C, Dn.m, FDn.m |
| ORI (OR Inverse) | ---\| \|---\|<br>---\|/\|---\| | загрузка состояния входа или другого дискретного устройства, применение к инвертированному полученному значению и результату выполнения предыдущей комманды логического ИЛИ; параллельное соединение нормально разомкнутых контактов | X, Y, M, S, T, C, Dn.m, FDn.m |


Пример:

```
LD   X0
OR   X1
OUT  Y0
```

Правило для Y0 будет работать при замкнутых контактах на входах X0 и X1.

LDP, LDF, ANP, ANF, ORP, ORF

Комманда LDP возвращает результат обнаружения положительного перепада на входе, указанном как аргумент. Тоесть, если в предыдущем цикле исполнения программы (скане) указанный вход выдавал низкий уровень, а в текущем высокий — результат исполнения — высокий уровень. Из этого следует, что высокий уровень как результат исполнения этой комманды будет сохраняться в течение одного скана, так как в следующем скане не будет условий достаточных для обнаружения положительного перепада, поскольку с точки зрения следующего скана в предыдущем уже был высокий уровень и положительный перепад не может произойти.
LDF возвращает результат обнаружения отрицательного перепада на входе аналогичным образом.
Из предыдущих описаний комманд, таких как AND, понятно, что при их исполнении происходит по сути сначала загрузка состояния указанного входа, как если бы это сделала комманда LD, а потом применение соответствующей логической операции. Комманды логических функций, мнемокоды которых заканчиваются на P и F, загружают состояние указанного в аргументах входа, как это произошло бы при использовании комманд LDP и LDF соответственно.

| мнемоническое обозначение | графическое обозначение | функция | применимые устройства |
|---|---|---|---|
| LDP (LoaD Pulse) | стр вверх | обнаружение положительного перепада на входе | X, Y, M, S, T, C, Dn.m, FDn.m |
| LDP (LoaD Fall) | стр вниз | обнаружение отрицательного перепада на входе | X, Y, M, S, T, C, Dn.m, FDn.m |
| ANP (ANd Pulse) | | обнаружение положительного перепада на входе, примение к полученному результату и результату выполнения предыдущей комманды логической функции AND | X, Y, M, S, T, C, Dn.m, FDn.m |
| ANF (ANd Fall) | | обнаружение отрицательного перепада на входе, примение к полученному результату и результату выполнения предыдущей комманды логической функции AND | X, Y, M, S, T, C, Dn.m, FDn.m |
| ORP (OR Pulse) | | обнаружение положительного перепада на входе, примение к полученному результату и результату выполнения предыдущей комманды логической функции OR | X, Y, M, S, T, C, Dn.m, FDn.m |
| ORF (OR Fall) | | обнаружение | X, Y, M, S, T, C, |

| | |
|---|---|
| отрицательного перепада на входе, примение к полученному результату и результату выполнения предыдущей комманды логической функции OR | Dn.m, FDn.m |

PLS, PLF, SET, RST, ALT

Эти 4 комманды предназначены для вывода в битовые устройства.
PLS выводит высокий логический уровень в течение 1 скана, если для указанного выхода в предыдущем скане правило для этой инструкции не работало, тоесть результат выполнения предшествующей ей инструкции — 0.
PLS выводит низкий логический уровень в течение 1 скана, если для указанного выхода в предыдущем скане правило для этой инструкции не работало, тоесть результат выполнения предшествующей ей инструкции — 0.
SET устанавливает состояние указанного выхода постоянно в высокое состояние, до тех пор пока оно не будет сменено другой коммандой, даже после прекращения действия правила, запускающего комманду SET.
RST устанавливает состояние указанного выхода постоянно в высокое состояние, до тех пор пока оно не будет сменено другой коммандой, даже после прекращения действия правила, запускающего комманду RST.
ALT сменяет сотояние указаного выхода на противоположное и оставляет его постоянным, до смены другой инструкцией. Следует проявлять осторожность при применинии этой инструкции. Если правило для этой инструкции действует от каких-то длительно присутствующих сигналов, смена состояния на указанном выходе будет производится в каждом скане, пока правило не перестанет действовать. Поэтому обычно целесообразно предварять эту инструкцию, такими, которые обнаруживают перепад.

| мнемоническое обозначение | графическое обозначение | применимые устройства |
|---|---|---|
| PLS (PuLSe) | нет | Y, M, S, T, C, Dn.m, FDn.m |
| PLF (PuLse Fall) | нет | Y, M, S, T, C, Dn.m, FDn.m |
| SET (SET) | ---(S)--- | Y, M, S, T, C, Dn.m, FDn.m |
| RST (ReSeT) | ---(R)--- | Y, M, S, T, C, Dn.m, FDn.m |
| ALT (ALTernate) | нет | Y, M, S, T, C, Dn.m, FDn.m |

Прикладные инструкции

| Тип | Обозначение | Функция |
|---|---|---|
| Подпрограмма | CJ | Условный переход |
| | CALL | Вызов подпрограммы |
| | SRET | Возврат из подпрограммы |
| | STL | Начало отключаемого участка кода |
| | STLE | Конец отключаемого участка кода |
| | SET | Включить участок кода |
| | ST | Выключить участок кода, не отключая текущий |
| | FOR | Начало цикла FOR-NEXT |
| | NEXT | Конец цикла FOR-NEXT |
| | FEND | Первый конец программы |
| Сравнение данных | LD＝ | LD истина, если (S1) = (S2) |
| | LD＞ | LD истина, если (S1) > (S2) |
| | LD＜ | LD истина, если (S1) =< (S2) |
| | LD＜＞ | LD истина, если（S1）≠（S2） |
| | LD＜＝ | LD истина, если（S1）≤（S2） |
| | LD＞＝ | LD истина, если（S1）≥（S2） |
| | AND＝ | AND истина, если（S1）＝（S2） |
| | AND＞ | AND истина, если（S1）＞（S2） |
| | AND＜ | AND истина, если（S1）＜（S2） |
| | AND＜＞ | AND истина, если（S1）≠（S2） |
| | AND＜＝ | AND истина, если（S1）≤（S2） |
| | AND＞＝ | AND истина, если（S1）≥（S2） |
| | OR＝ | OR истина, если（S1）＝（S2） |
| | OR＞ | OR истина, если（S1）＞（S2） |
| | OR＜ | OR истина, если（S1）＜（S2） |
| | OR＜＞ | OR истина, если（S1）≠（S2） |
| | OR＜＝ | OR истина, если（S1）≤（S2） |
| | OR＞＝ | OR истина, если（S1）≥（S2） |
| Копирование содержимого | MOV | Копировать значение в один регистр |
| | BMOV | Копировать значение в несколько регистров |
| | FMOV | Fill move |
| | FWRT | Запись во Flash память |
| | MSET | Включение зоны |
| | ZRST | Отключение зоны |
| | SWAP | Старшие 8 разрядов слова меняются с младшими 8 разрядами |
| | XCH | перестановка |
| Операции с данными | ADD | Сложение |
| | SUB | Вычитание |
| | MUL | Умножение |
| | DIV | Деление |
| | INC | Инкремент |
| | DEC | Декремент |
| | MEAN | Среднее |
| | WAND | Слово «И» |
| | WOR | Слово «ИЛИ» |

| | | |
|---|---|---|
| | WXOR | Слово «Исключающее или» |
| | CML | Дополнение |
| | NEG | Негатив |
| | | |
| | | |

(Далее в начале каждого описания должна следовать табличка с особенностями, я приведу для примера как я их вижу, но ставить не буду, иначе если понадобиться переделать в каких-то эстетических или типографских целях — придётся переделывать все.)

Вобщем, вот концепт:

| Операция сложения ADD | | применима в устройствах |
|---|---|---|
| 16 bit: ADD | 32 bit: DADD | XC1, XC3, XC5 |

Применима к блокам:

| | | | | источник 1 | | | | |
|---|---|---|---|---|---|---|---|---|
| K/H | DX | DY | DM | DS | TD | CD | D | FD |
| | | | | источник 2 | | | | |
| K/H | DX | DY | DM | DS | TD | CD | D | FD |
| | | | | назначение (или получатель) | | | | |
| | | DY | DM | DS | TD | CD | D | |

CJ — Condition Jump — Условный переход

Допустимый аргумент: указатель — P
Допустимые границы аргумента: P0-P9999

Исполнение инструкции вызывает переход в программе на инструкцию следующую за указателем, указанным в аргументе при условии, что предыдущая инструкция возвратила высокий уровень.
На практике примение инструкции позволяет сократить длительность скана, избежать повторных воздействий на одни и те же устройства, а также менять поведение устройства в зависимости от его режима работы.

(рис с 75/85)

В приведённой выше программе видно, что существует 2 правила воздействия на Y0. В то же время, первое встречающееся правило для Y0 будет рассматриваться только при низком уровне на входе X0.
Инструкция не осуществляет переход от одной инструкции STLE к другой.

CALL — call subroutine, SRET — subroutine return — вызов и возврат из подпрограммы

Допустимый аргумент для CALL: указатель — P
Допустимые границы аргумента: P0-P9999

(рис с 76/86)

Инструкция CALL вызывает переход на подпрограмму, на которую указывает указатель в аргументе, при условии, что предыдущая инструкция возвратила высокий уровень. Инструкция SRET возвращает управление исполнением программы на инструкцию, следующую за последней исполненной инструкцией CALL.
В приведённом примере подпрограмма между указателем P10 и инструкцией SRET исполняется при высоком уровне на входе X0. Инструкция FEND показывает конец тела основной программы, а END - всей программы.

SET, ST, STL, STLE — управление потоками

Допустимый аргумент для SET, ST, STL: статус — S

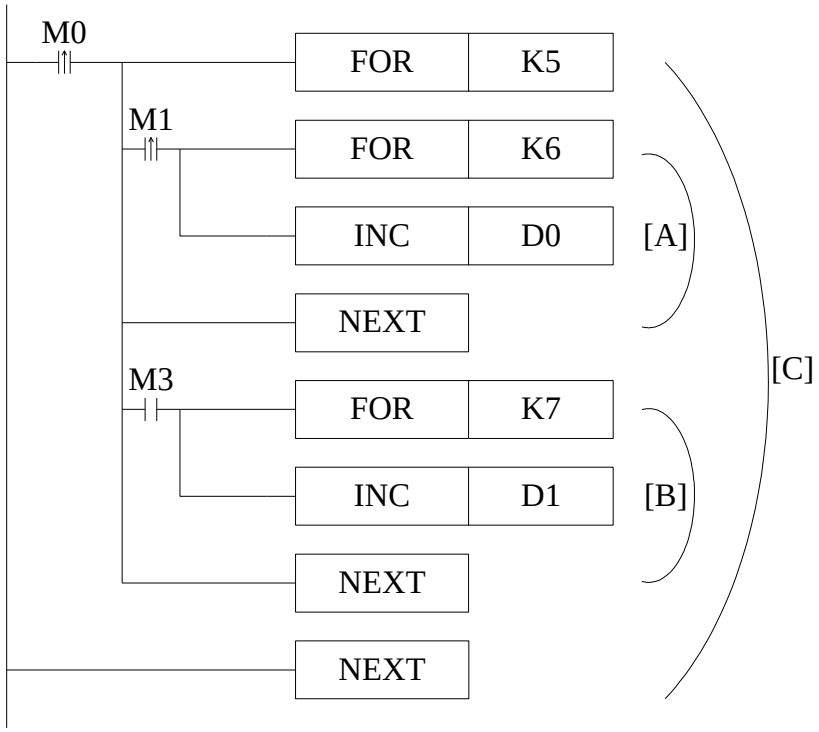
Эти 4 инструкции позволяют создавать и управлять программными потоками. В тексте программы начало потока обозначают инструкцией STL, а заканчивают инструкцией STLE. Затем, чтоб разрешить исполнение потока необходимо исполнить инструкции SET или ST с аргументом, указывающим на номер потока. Разница между ними в том, что SET прекращает работу потока, из которого она исполняется. Запретить работу потока можно инструкцией RST с аргументом, указывающим на поток.

(рис с 77/87)

Следует учитывать, что после завершения потока таймеры им используемые, а также регистры, используемые инструкциями PLS, PLF обнуляются.

Цикл FOR-NEXT
Аргументы ：DX、DY、DM、DS、T、C、D、FD、K

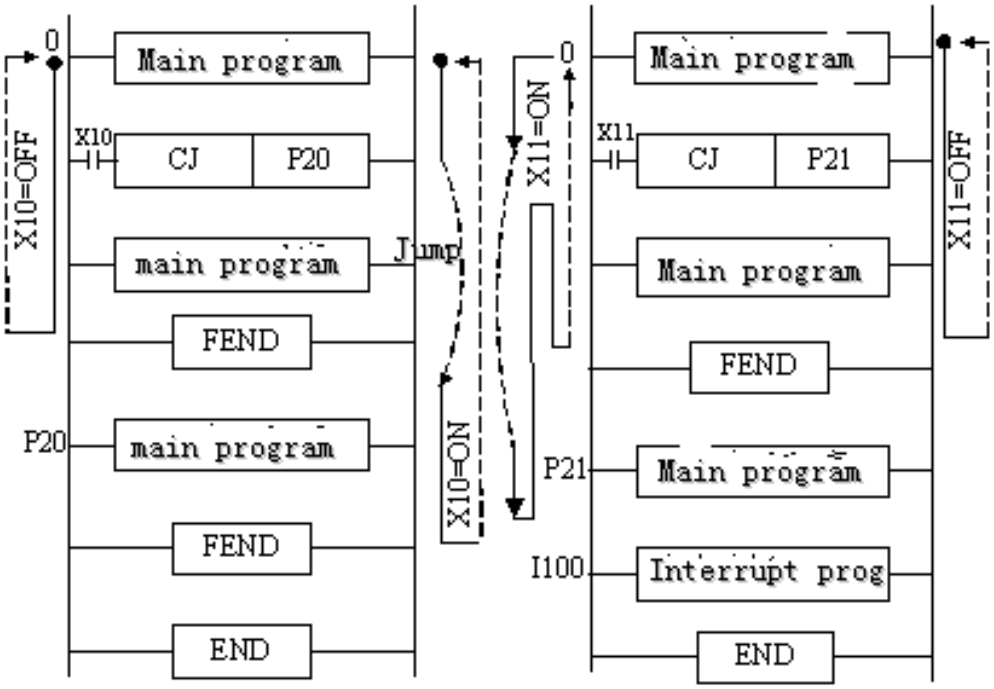
Первой выполняется инструкция, находящаяся между инструкцией FOR-NEXT на некоторое время (зависит от размера данных внутри цикла), затем выполняется инструкция за NEXT.




1) каждая инструкция FOR должна заканчиваться инструкцией NEXT. Максимальная вложенность таких пар не должна превышать 8-ми.
2) между FOR-NEXT инструкции LDP и LDF действуют один внешний цикл. Когда состояние катушки M0 изменяет свое состояние с 0 на 1 и M1 изменяет свое состояние с 0 на 1, цикл A выполнится 6 раз
3) когда катушки M0 и M3 меняют свое состояние с 0 на 1, цикл B выполнится 5*7=35 раз
4) если много циклов, то скан цикл будет продолжительным
5) если инструкция NEXT стоит перед FOR, или нет NEXT, или NEXT находится после блока FENG, END, или количество NEXT не равно FOR – это приведет к ошибке
6) между FOR – NEXT недопустимо размещение блоков CJ, также это касается STL , FOR – NEXT должны использоваться парно

**FEND и END**
**Аргументы – нет**

Блок **FEND** используется для указания конца главной программы и начало блоков подпрограмм (потоков). В нормальном состоянии (при нормальных обстоятельствах) инструкция FEND выполняет действия, подобные инструкции END, то есть обработку выходящих данных, входящих данных и таймер «сторожевая собака» обновляющий все выходы при выполнении.

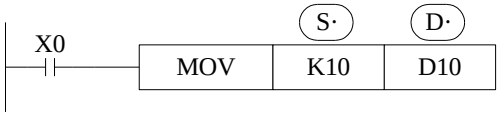



- если в программе инструкция CALL находится после инструкции FEND, должна быть инструкция SRET. Если используется точечный указатель программы, должна использоваться инструкция SRET (то есть, как я понимаю, после блока P1 должен стоять блок SRET)
- После выполнения инструкции CALL  и перед выполнением SRET инструкции, если выполняется FEND инструкция или выполняется FEND после выполнения FOR инструкции и перед выполнение NEXT – данная ситуация приведет к ошибке
- если используется инструкция FEND, то конец программы должен заканчиваться END-ом, если FEND не используется – использование END не обязательно.

# Запись значений в регистры и установка/ сброс катушек

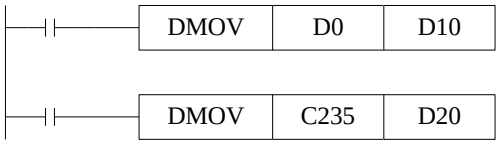
Блоки данного раздела схожи по функциям, они записывают в регистры какое-то значение, если это катушки, то записывается (катушка устанавливается) в 0 или 1. Занесение данных может производиться как в одиночные регистры, так и в блоки регистров.

***<u>Блок MOV</u>***
Аргументы - DX、DY、DM、DS、T、C、D、FD、K




В данном коде, при срабатывании катушки X0 в регистр D[10] будет занесено значение 10 (значение автоматически преобразуется в бинарный код).

Есть возможность также записывать по 2 слова сразу:

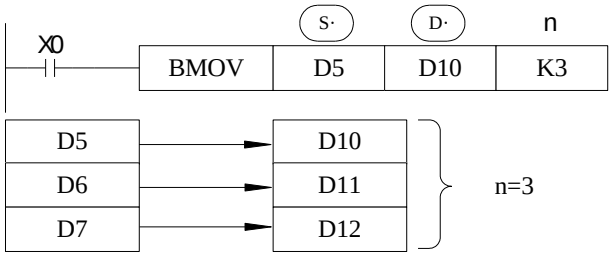



(D1, D0) → (D11, D10)
(C235 текущее значение) → (D21, D20)

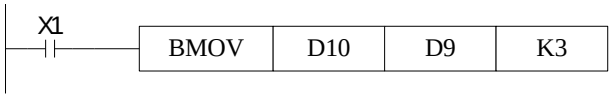
***<u>Блок BMOV</u>***

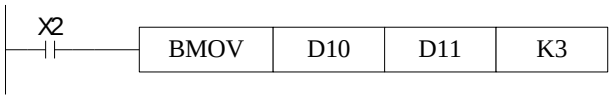
Аргументы: DX、DY、DM、DS、T、C、D、FD、K

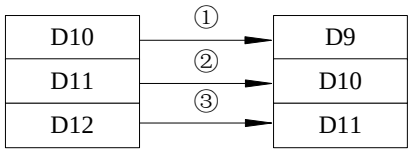
Данный блок служит для последовательного копирования значений регистров начиная с адреса S в регистры, которые начинаются с адреса D. Количество регистров для копирования указывается в n.

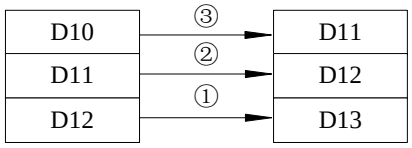



Значения адресов в S и D автоматически увеличивается на 1 до заданного значения в n, то есть



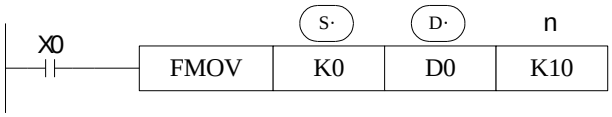



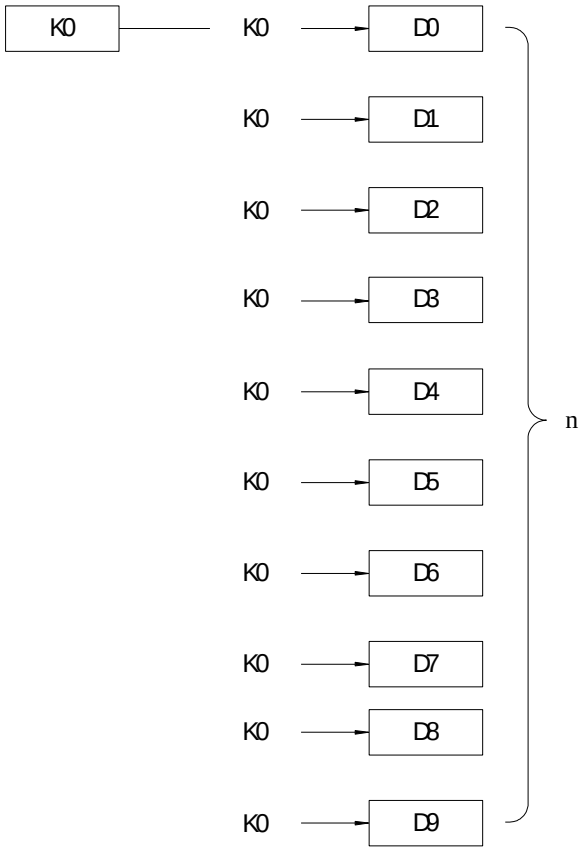




***<u>Блок FMOV</u>***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K



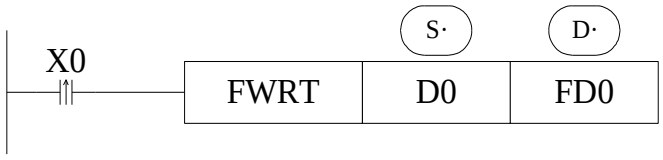

При срабатывании катушки X0 в регистры D[0] – D[9] будет занесено значение 0. То есть, S – является источником, с которого берется значение, D – первый регистр, в который запишется значения из S, n – количество регистров для записи.




***<u>Блок FWRT – запись в флэш-память</u>***

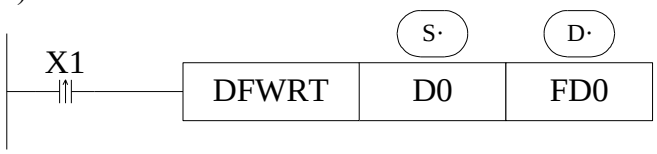
Аргументы: DX、DY、DM、DS、T、C、D、FD、K

1) запись одного слова:



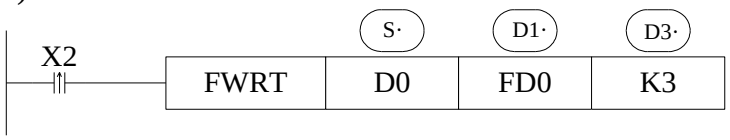

Данная функция запишет значение регистра D[0] в FD[0]

2) запись 2 слов




Данная функция запишет D[0],D[1] в FD[0],FD[1]

3) множественная запись



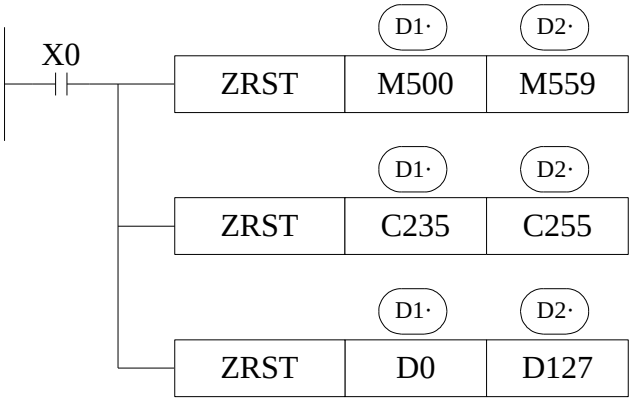

В данном случае произойдет следующее: значение D[0] запишется в FD[0], значение D[1] запишется в FD[1], значение D[2] запишется в FD[2]

Примечание:

1) FWRT позволяет записывать данные во флэш-память, данная память не зависит от питания, в ней рекомендуется хранить все технические параметры системы
2) для записи во флэш-память требуется много времени, порядка 150 мс. Данную операцию не рекомендуется часто применять
3) флэш-память можно перезаписывать около 1 000 000 раз. Для вызова функции записи рекомендую использовать блоки LDP, LDF, то есть с одиночными сигналами.

***Блок ZRST***

Аргументы: все битовые и словесные аргументы

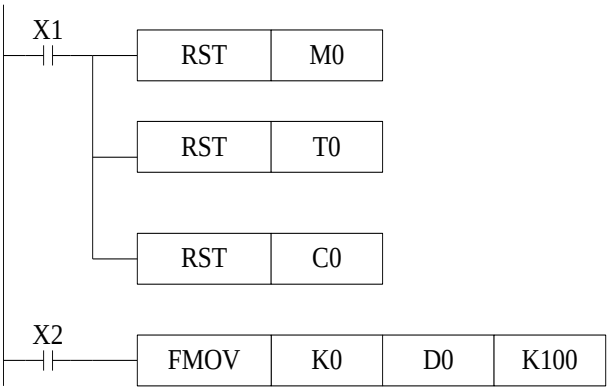



Сброс зоны катушек M500~M599

Сброс зоны слов (знач. счетчиков ) C235~C255

Сброс значения хранимых регистров D0~D127

То есть данная инструкция сбрасывает блок регистров. Для сброса одиночных регистров существуют следующие блоки:




Сброс M0

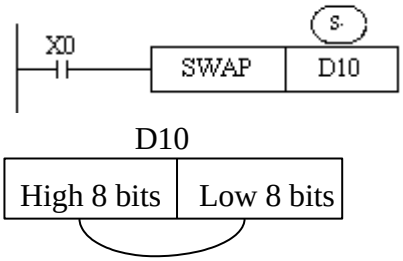
Сброс текущего значения T0

Сброс C0

Запись K0 в D0~D99

***Блок SWAP***

Аргументы: DX、DY、DM、DS、T、C、D 、FD





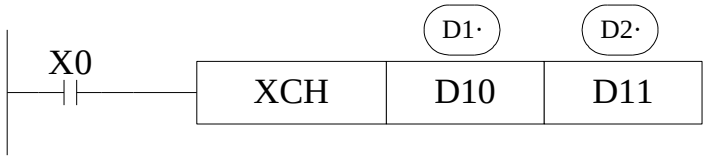


Меняет местами 8 старших регистров и 8 младших регистров D10

***Блок XCH***

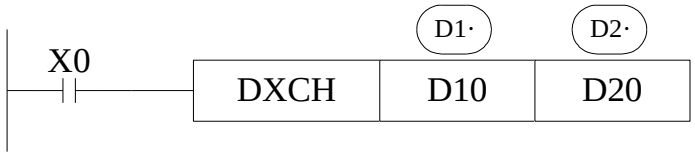
Аргументы: DX、DY、DM、DS、T、C、D 、FD

16-ти битная инструкция




Заносит значение D1 в D2, а D2 в D1.

32-х битная инструкция




Меняет значения местами D10 – D20, D11 – D21

# Операции с данными

***Блок ADD – операция сложения***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K


(D10) + (D12) → (D14)

Данная функция суммирует (арифметическое сложение) значения регистров S1 и S2, затем записывает их в D.

1) суммировать можно как положительные аргументы, так и отрицательные (-5 + 3 = -2)
2) если результат сложения равен 0 – установится флаг M8020, если полученное значение выйдет за пределы 323,767 (для 16-ти битного регистра) или за пределы 2, 147, 483, 647 (для 32 битного регистра), то установится флаг M8022, если же результат выйдет за пределы -323,767 или -2,147,483,647, то установится флаг M8021.
3) если не использовать третий операнд, то результат будет заносится в первый операнд S1
4) для сложения 32 – битных регистров используйте DADD

***Блок SUB – операция вычитания***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K

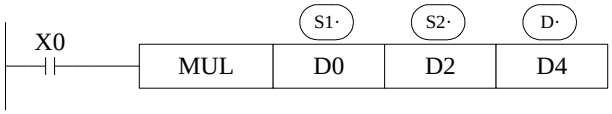

（D10）－（D12）→（D14）

Функция аналогична операции сложения. Из регистра S1 вычитается значение регистра S2 и заносится в D. Возможна также краткая запись, то есть используются только S1 и S2. Флаги ошибок те же.

***<u>Блок MUL – операция умножения</u>***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K

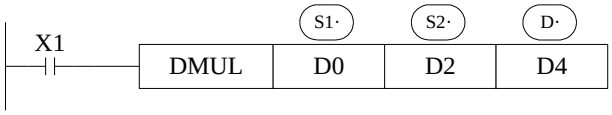
16-ти битная операция




BIN BIN BIN
（D0）× (D2) → （D5, D4）
16 bits 16 bits → 32 bits

С примера видно, что в результате перемножения 2-х 16-ти битных регистров получается один 32-х битный регистр. Старший бит результата содержит знак значения 0 – положительное, 1 – отрицательное.

32-х битная операция



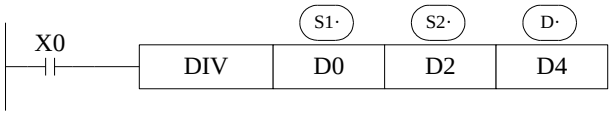

BIN BIN BIN
（D1, D0）× (D3, D2) →
（D7, D6, D5, D4）
32 bits 32 bits → 64 bits

В результате перемножения, получаем 64-х разрядное значение, которое не будет возможности показать на мониторе или панели оператора, так что рекомендую использовать тип float.

***<u>Блок DIV – операция деления</u>***

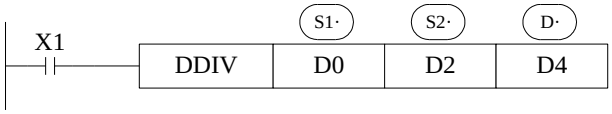
Аргументы: DX、DY、DM、DS、T、C、D、FD、K

16- ти битная операция




Здесь S1 является делимым, S2 – делителем, а в D заносится результат деления.

32 –х битная операция



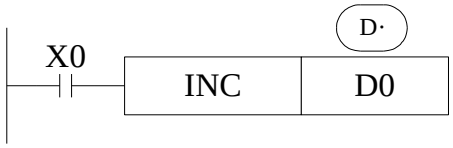

| Dividend | Divisor | Result | Result |
|---|---|---|---|
| BIN | BIN | BIN | BIN |
| (D1,D0) | ÷ (D3,D2) | （D5,D4）… | (D7,D6) |
| 32 bits | 32 bits | 32 bits | 32 bits |

Если значение делителя равно 0, то произойдет ошибка, и выполнение блока прекратится. Деление, как и в предыдущем варианте целочисленное, то есть остаток отбрасывается. Старший бит, как и при умножении указывает знак значения 0 – плюс, 1 – минус. Если делитель, или делимое со знаком минус, то и результат будет со знаком минус.

***<u>Блоки INC и DEC</u>***

Аргументы: DX、DY、DM、DS、T、C、D、FD

Данные блоки служат для увеличения или уменьшения значения регистра на 1 при каждом вызове такого блока.



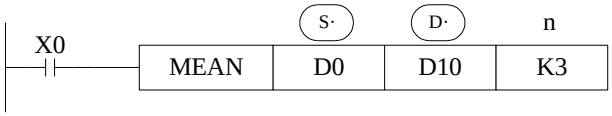

（D0）＋1→(D0)


（D0）－1→(D0)

Когда инструкция 16-ти битная, и значение регистра достигает 32 676, то следующее значение при инкременте станет -32 676. при декрементировании происходит обратная ситуация, то есть при достижении -32 676 следующее значение будет 32 676. Для работы с 32-х битными значениями используются DINC и DDEC.

***<u>Блок MEAN – среднее арифметическое</u>***

Аргументы: DX、DY、DM、DS、T、C、D、FD



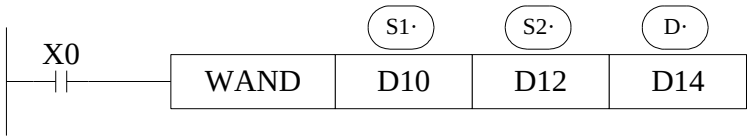

$$\frac{(D0) + (D1) + (D2)}{3} \longrightarrow (D10)$$

Если значение n выходит за допустимый диапазон – 1-64, то сгенерируется ошибка.

***<u>Блоки WAND, WOR, WXOR</u>***

Аргументы: DX、DY、DM、DS、T、C、D、FD,K

1) выполняет побитно операцию AND

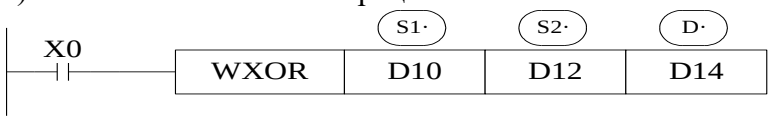



0&0=0 0&1=0
1&0=0 1&1=1

2) выполняет побитно операцию OR

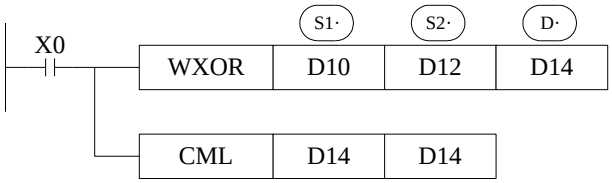

0 or 0=0 0 or 1=1
1 or 0=1 1 or 1=1

3) выполняет побитно операцию XOR



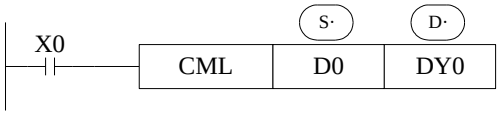

0 xor 0=0 0 xor 1=1
1 xor 0=1 1 xor 1=0

Данная инструкция, в сочетании с инструкцией CML, выполняет действия, похожие на логическую операцию XOR NOT

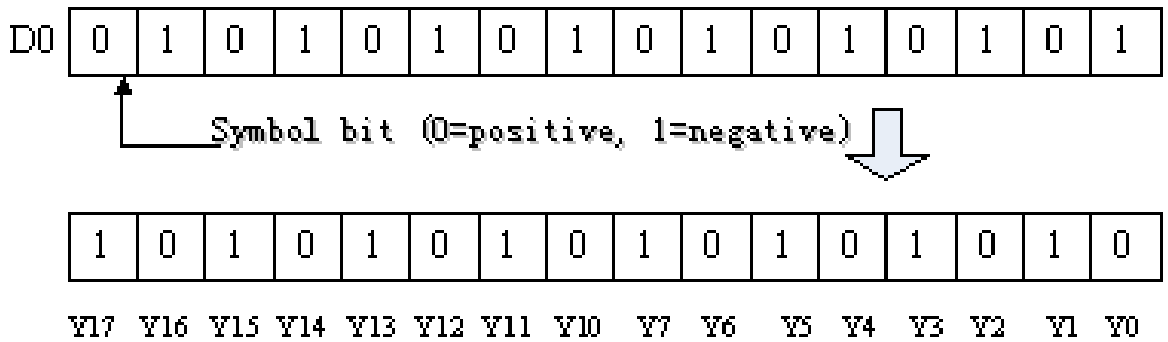



***<u>Блок CML</u>***

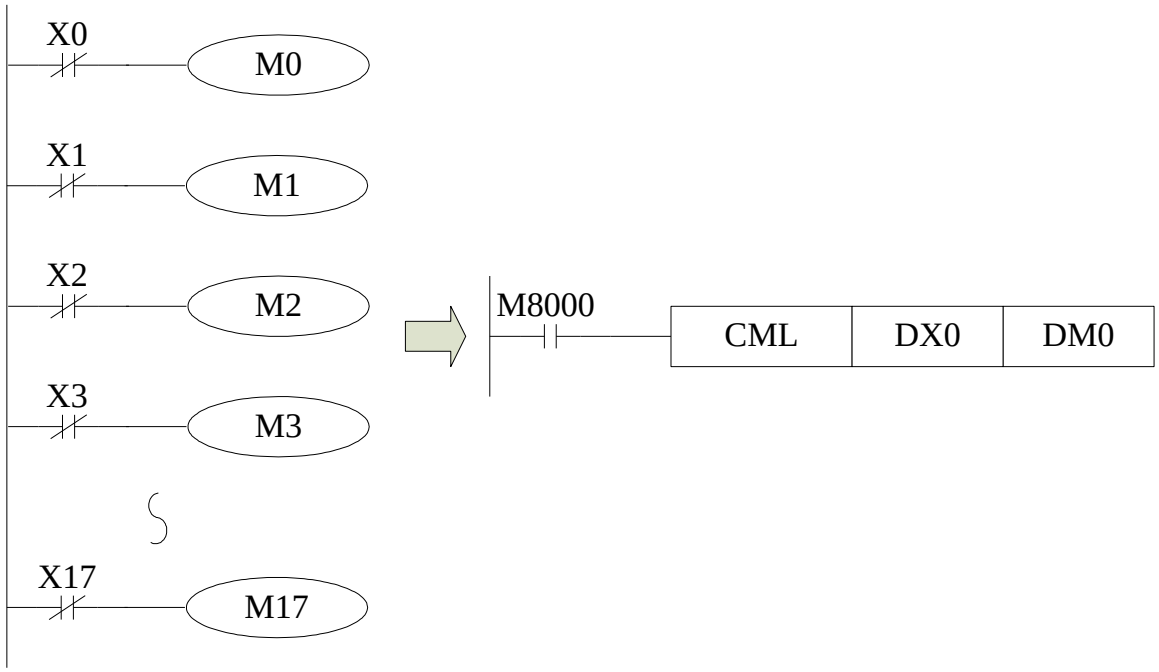
Аргументы: DX、DY、DM、DS、T、C、D、FD




Данный блок выполняет следующую функцию: побитно прочитав аргумент S, он записывает побитно значения в D, предварительно инвертируя каждый бит. То есть 0 станет 1, а 1 – 0.



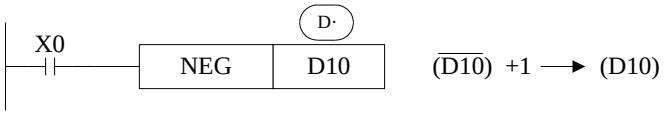

Чтение инвертированных входов




Данный пример показывает, что левая и правая части имеют одинаковый функционал, однако выражение справа имеет более компактный вид.

***<u>Блок NEG</u>***

Аргументы: DX、DY、DM、DS、T、C、D、FD



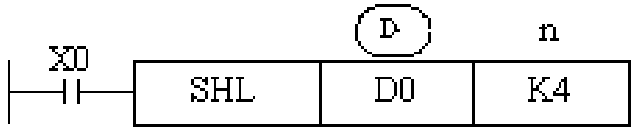

Данный блок производит инвертирование указанного аргумента в D. То есть 0 становится 1, 1 – 0. Когда инвертирование завершается, старшая единица прибавляется к битовому значению числа. В результате получаем полностью обратное значение, заданному в аргументе.
Если эта инструкция используется в основном цикле, она будет выполняться каждый скан.

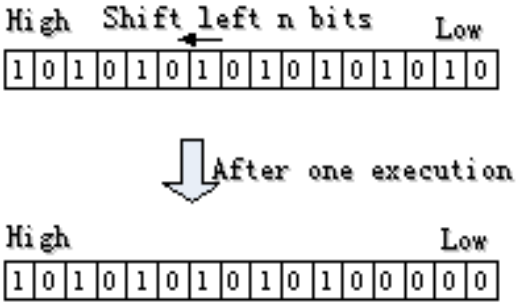
# Инструкции сдвигов (смешений)

***<u>Блок SHR и SHL</u>***

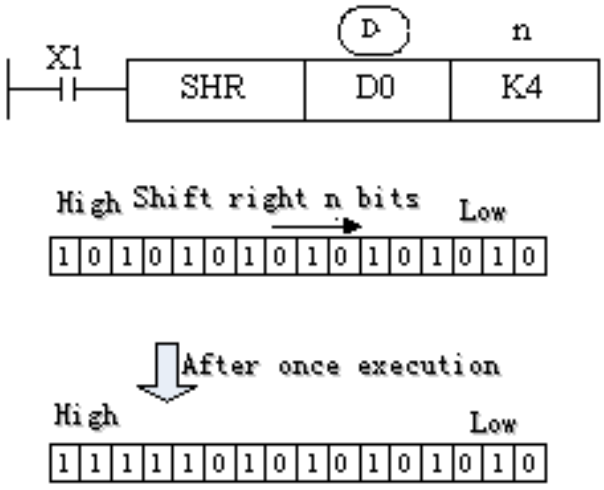
Аргументы: DX、DY、DM、DS、T、C、D、FD

Арифметический сдвиг влево






Арифметический сдвиг вправо

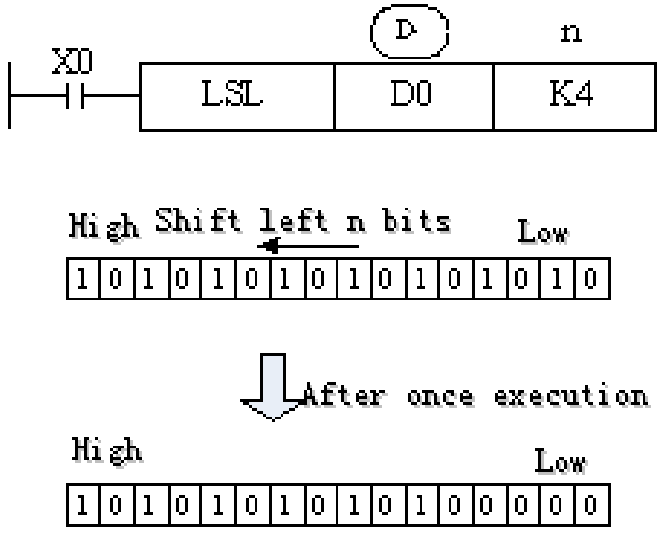



Примечания. Данные инструкции выполняются каждый цикл. В случае с 32- х битными значениями, блоки работают аналогично.

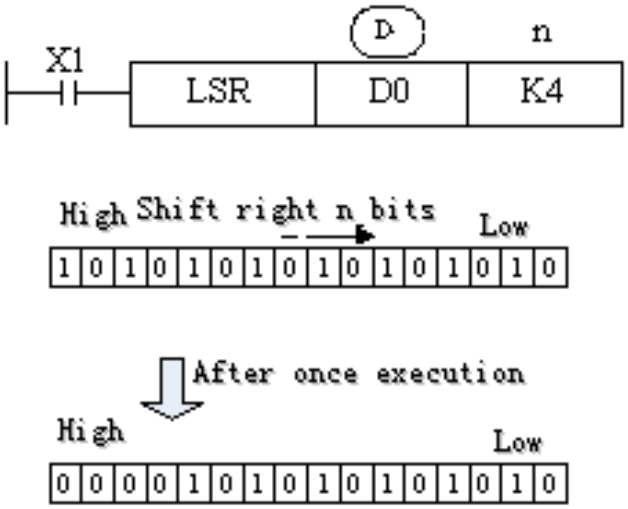
***<u>Блок LSL и LSR</u>***

Аргументы: DX、DY、DM、DS、T、C、D、FD

Логический сдвиг влево



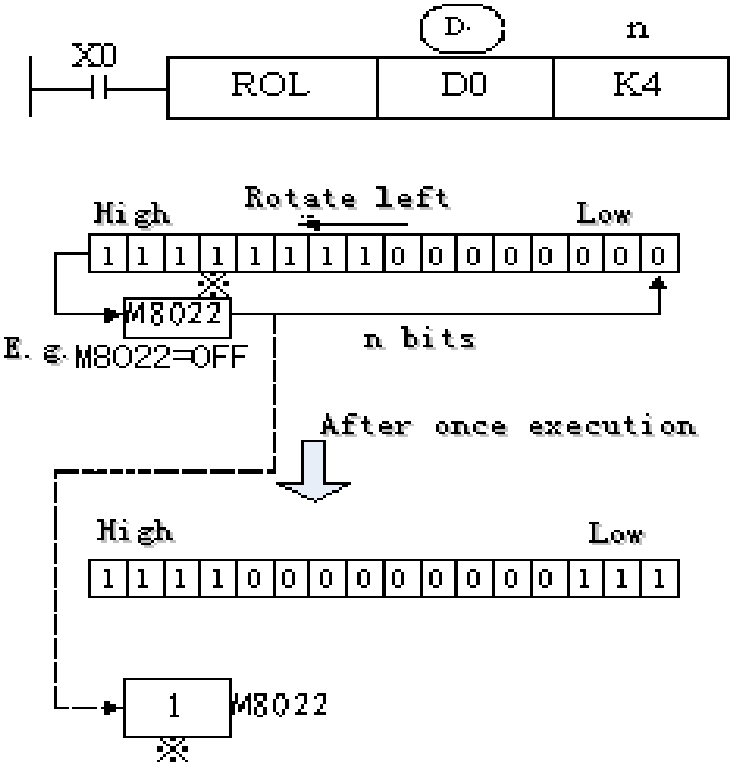
Логический сдвиг вправо




Все остальное аналогично арифметическим сдвигам.

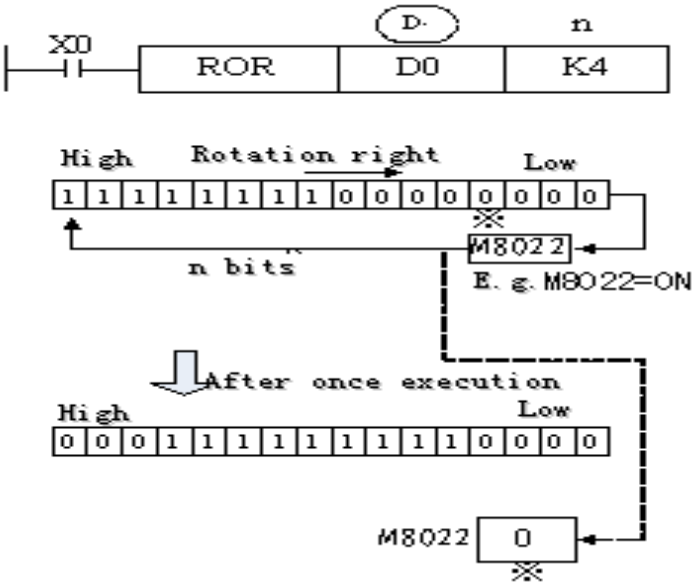
***<u>Блоки ROL и ROR</u>***

Аргументы: DX、DY、DM、DS、T、C、D、FD
Кольцевой сдвиг влево




Когда катушка X0 поменяет свое значение с 0 на 1, произойдет кольцевой сдвиг влево.

Кольцевой сдвиг вправо

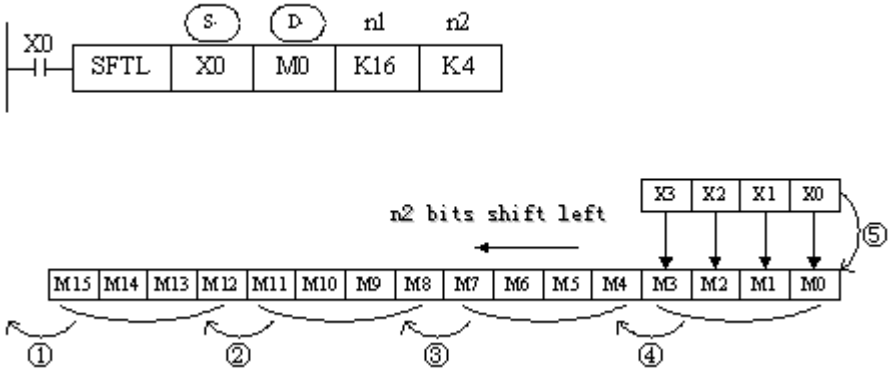



Желательно не использовать данные блоки в каждом цикле программы. Работа с 32-х разрядными значениями аналогична.

***<u>Блоки SFTL и SFTR</u>***

Аргументы: DX、DY、DM、DS、T、C、D、FD

SFTL - влево

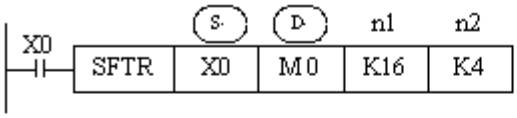



① M15~M12→overflow
② M11~M 8→M15~M 12
③ M 7~M 4→M11~M8
④ M 3~M 0→M7~M4
⑤ X 3~X 0→M3~M0

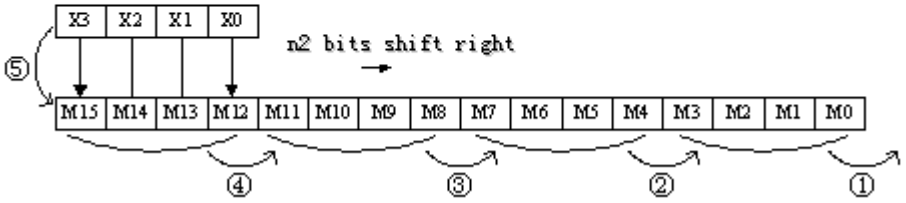
Данные блоки делают несколько копий источника S длиной n2 и записывают данные копии, в сегмент, начиная с адреса D длиной n1.

Как видно из примера, первый аргумент X0, длина блока – 4, то есть первый блок x0,x1,x2,x3. Второй аргумент M0, длина сегмента 16, следовательно получаем 4 пары (16 : 4 = 4). Данная функция при срабатывании X0 поместит в M0,M4,M8,M12 значение X0, в M1,M2,M3, M4 – X1 и т.д.

SFTR – вправо

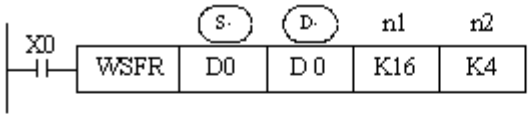



① M 3~M 0→overflow
② M 7~M 4→M3~M0
③ M11~M 8→M7~M4
④ M15~M12→M11~M8
⑤ X 3~X 0→M15~M12

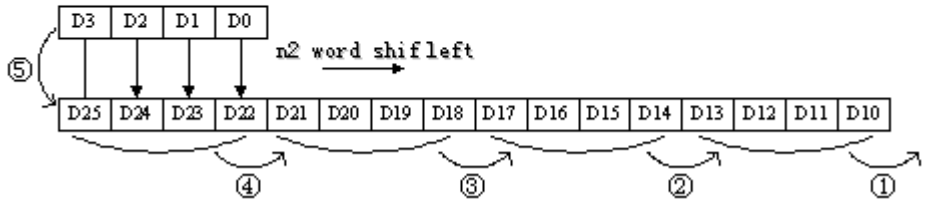


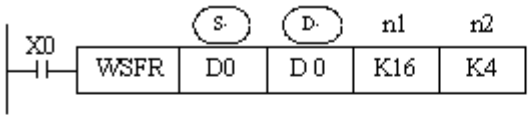

***<u>Блоки WSLF и WSFR</u>***

Аргументы: DX、DY、DM、DS、T、C、D、FD

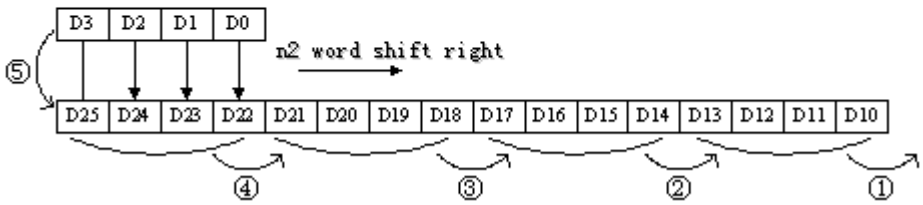



① D25~D22→overflow
② D21~D18→D25~D22
③ D17~D14→D21~D18
④ D13~D10→D17~D14
⑤ D 3~D 0→D13~D10




① D13~D10→overflow
② D17~D14→D13~D10
③ D21~D18→D17~D14
④ D25~D22→D21~D18
⑤ D 3~D 0→D25~D22

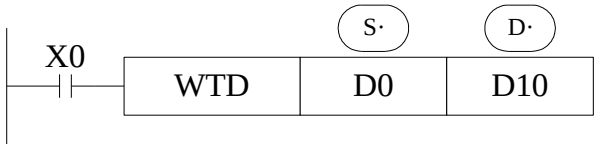



Данные блоки аналогичны предыдущим, разве что вместо катушек работа происходит с хранимыми регистрами.

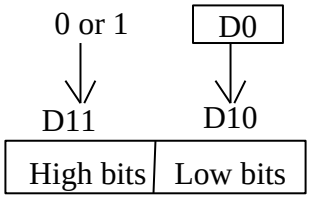
## 5.7 Преобразование форматов данных

***Блок WTD***

Аргументы: DX、DY、DM、DS、T、C、D、FD



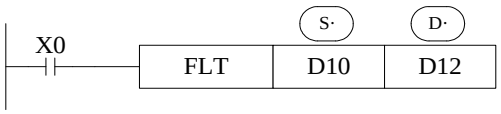

（D0） → （D11，D10）
Single word Double word




Данный блок преобразует 16-ти битное число в 32 – битное. Если число четное, то старший бит содержит 0, если отрицательное – 1.

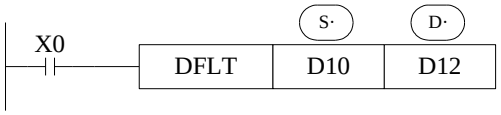
***Блок FLT (DFLT, FLTD)***

Аргументы: DX、DY、DM、DS、T、C、D、FD



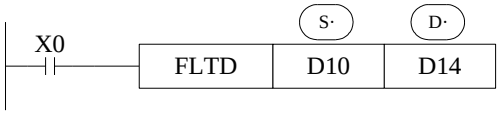

(D10) → (D13,D12)
BIN integer Binary float point

Данный блок преобразует 16-ти битное целое значение в 32-х вещественное.




（D11,D10） → （D13,D12）
BIN integer Binary float point

Данный блок преобразует 32-х битное целое в 32-х битное вещественное значение



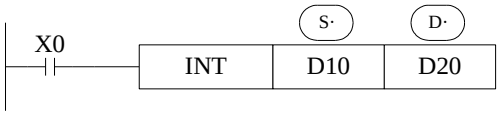

（D13,D12,D11,D10） → （D17,D16,D15,D14）
BIN integer Binary float point

Данный блок преобразует 64-х битное целое в 64-х битное вещественное значение.

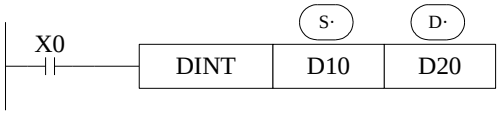
Данные команды противоположны командам INT. Константы K(десятичное число) и H(шестнадцатеричное) приводятся к типу автоматически.

***Блок INT***

Аргументы: DX、DY、DM、DS、T、C、D、FD




（D11,D10） → (D20)

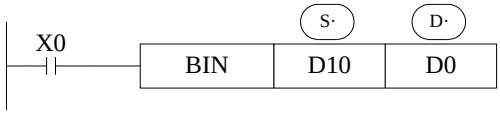



（D11,D10） → (D20, D21)

Данные блоки приводят значение к типу INT, первый блок 16-ти битный, второй 32-х битный.

***Блок BIN***

Аргументы: DX、DY、DM、DS、T、C、D、FD




Границы данных：0~9,999 или 0~99, 999, 999 .
Для преобразования этому блоку необходимо привести число к формату BCD→ затем BIN

Если аргумент S не является BCD, блок не выполнится и сработает флаг M8067. Константа K автоматически приводится к данному типу.

***Блок BCD***

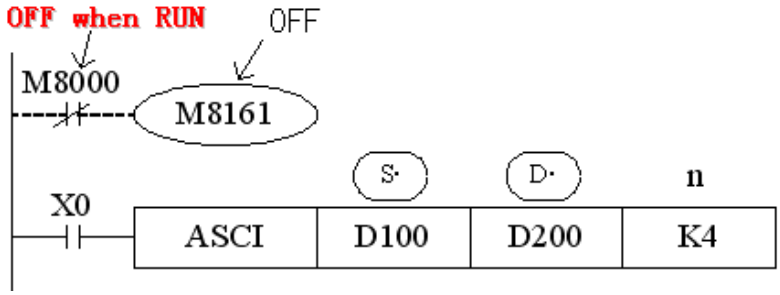
Аргументы: DX、DY、DM、DS、T、C、D、FD


Данный блок приводит тип BIN к типу BCD. Входной аргумент должен находиться в рамках, для 16-ти разрядных значений, от 0 до 9999., для 32-х разрядных в пределах от 0 до 99999999.
Эта команда позволяет выводить данные на 7-ми сегментный монитор.

***Блок ASCII***

Аргументы: DX、DY、DM、DS、T、C、D、FD

16-ти битное преобразование, когда M8161=OFF




Данный блок преобразовывает каждый бит источника S в шестнадцатеричный код в аргумент D, данные автоматически размешаются в старшие 8 бит и младшие 8 бит аргумента D.
Преобразование в буквенно-цифровой формат начинается с параметра n.
Пример последовательности преобразования:
Назначим начальные данные

(D100)=0ABCH

(D101)=1234H

(D102)=5678H

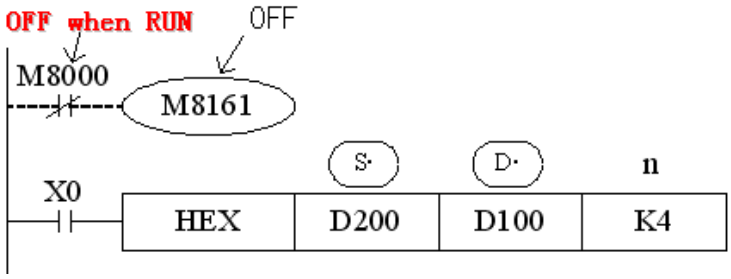
[0]=30H [1]=31H [5]=35H
[A]=41H [2]=32H [6]=36H
[B]=42H [3]=33H [7]=37H
[C]=43H [4]=34H [8]=38H

| D \ n | K1 | K2 | K3 | K4 | K5 | K6 | K7 | K8 | K9 |
|---|---|---|---|---|---|---|---|---|---|
| D200 down | [C] | [B] | [A] | [0] | [4] | [3] | [2] | [1] | [8] |
| D200 down | | [C] | [B] | [A] | [0] | [4] | [3] | [2] | [1] |
| D201 down | | | [C] | [B] | [A] | [0] | [4] | [3] | [2] |
| D201 up | | | | [C] | [B] | [A] | [0] | [4] | [3] |
| D202 down | | | | | [C] | [B] | [A] | [0] | [4] |
| D202 up | | | | | | [C] | [B] | [A] | [0] |
| D203 down | | | | | | | [C] | [B] | [A] |
| D203 up | | | | | | | | [C] | [B] |
| D204 down | | | | | | | | | [C] |

***Блок HEX***

Аргументы: DX、DY、DM、DS、T、C、D、FD

16-ти битный режим включается, когда M8161=OFF

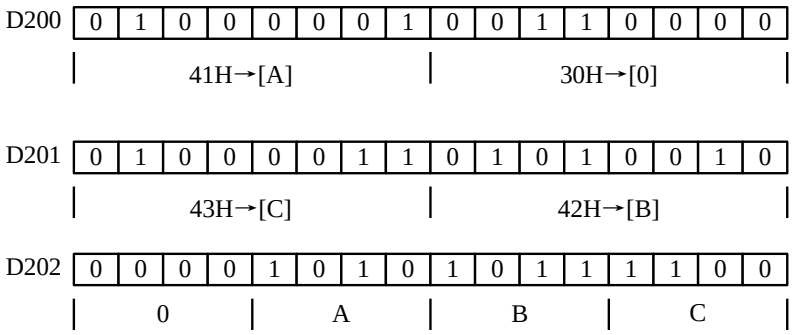



Блок преобразовывает младшие и старшие 8 бит в HEX код. Адрес каждый раз смещается на 4 бита. Преобразование начинается с аргумента, указанного в n.

Последовательность выполнения блока следующая:

| (S· ) | ASCII | HEX Conversion |
|---|---|---|
| D200 up | 30H | 0 |
| D200 up | 41H | A |
| D201 down | 42H | B |
| D201 up | 43H | C |
| D202 down | 31H | 1 |
| D202 up | 32H | 2 |
| D203 down | 33H | 3 |
| D203 up | 34H | 4 |
| D204 | 35H | 5 |

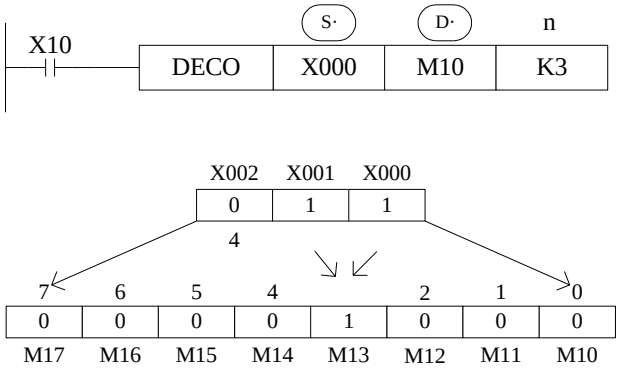
| n \ (D· ) | D102 | D101 | D100 |
|---|---|---|---|
| 1 | Not change to be 0 | | ···0H |
| 2 | | | ··0AH |
| 3 | | | ·0ABH |
| 4 | | | 0ABCH |
| 5 | | ···0H | ABC1H |
| 6 | | ··0AH | BC12H |
| 7 | | ·0ABH | C123H |




***Блок DECO***

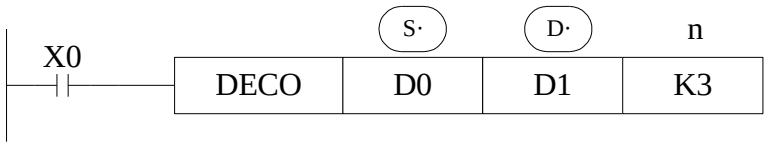
Аргументы: DX、DY、DM、DS、T、C、D、FD

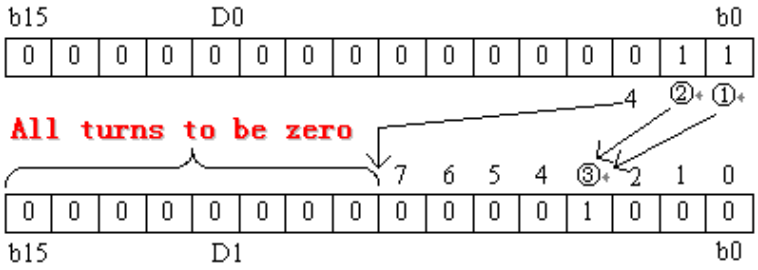
Когда параметр D – катушка, и n<=16




- Начальный адрес источника 3, стартуем с катушки M10, количество бит - 3 (M13) is 1. Если все источники 0, M10 равно 1
- когда n=0, блок не работает, за пределами n=0~16, то инструкция не выполнится
- когда n=16, если кодирующая команда “D” катушка, её номер 2^8=256。
- когда на вход передается 0, инструкция не выполнится, активируйте S, что бы активировать выход.

Когда D это слово, и n меньше 4:




Исходные младшие n бит (n<=4), кодируются, если n<=3 старшие биты кодируются нулем. Когда n=0, данная инструкция не выполнится.

***Блок ENCO***

Аргументы: DX、DY、DM、DS、T、C、D、FD

***Блок ENCOL***

Аргументы: DX、DY、DM、DS、T、C、D、FD

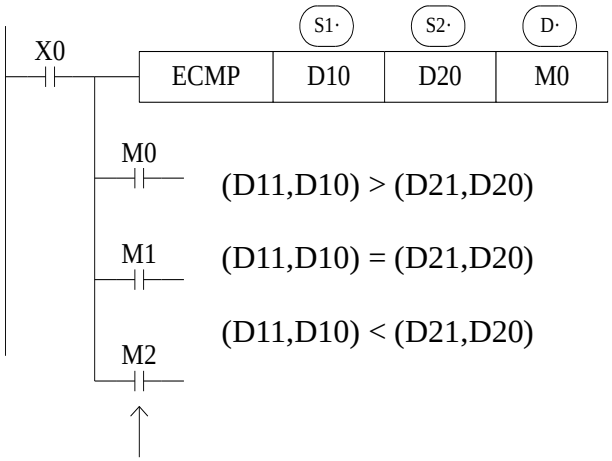
# Вещественные операции

| Блок | Функция |
|---|---|
| ECMP | Строгое сравнение вещественных чисел |
| EZCP | Сравнение блока вещественных чисел |
| EADD | Сложение вещественных чисел |
| ESUB | Вычитание вещественных чисел |
| EMUL | Умножение вещественных чисел |
| EDIV | Деление вещественных чисел |

| | |
|---|---|
| ESQR | Вычитание квадратного корня |
| SIN | Синус |
| COS | Косинус |
| TAN | Тангенс |

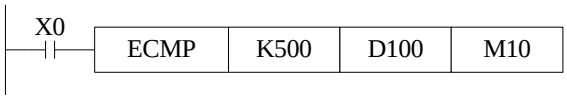
***<u>Блок ECMP</u>***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K



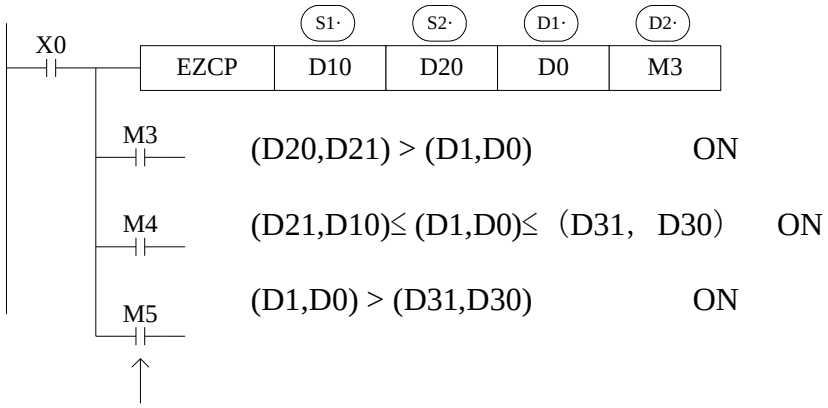

（D11,D10） : （D21,D20）→M0,M1,M2

Для сравнения значений регистров можно использовать блоки LD<,LD>=,DLD<> и т.д. Но, если в данные блоки передать вещественное число, то его дробная часть будет отброшена, и сравниваться будут только целые части. Это ведут к потере точности и запаздыванию реагирования на какие-то события.
Данный блок позволяет сравнивать вещественные значения. В параметры S1 и S2 передаются значения для сравнения, а в параметре D указывается начальный флаг результата.
С предыдущего примера – M0 установится в 1, если параметр S1> S2, M1 установится в 1, если S1= S2, M2 установится в 1, если S1<S2.
Также можно сравнивать и константы:




（K500）：（D101，D100）→M10,M11,M12

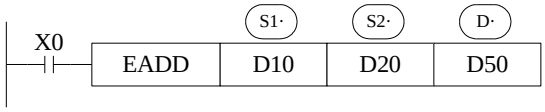
Аргументы: DX、DY、DM、DS、T、C、D、FD、K
Данный блок производит сравнение области значений




Результаты сравнения сохранятся, даже если команда сравнения будет отключена. Если в качестве параметров используются константы, они автоматически будут приведены к вещественному типу.

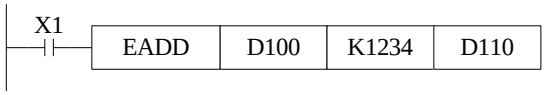
***Блок EADD***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K




（D11,D10） + (D21,D20) → (D51,D50)

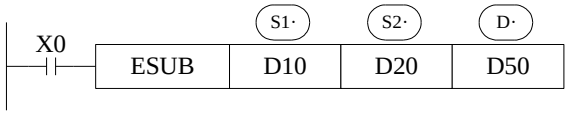
Данный блок производит алгебраическое сложение двух вещественных значений. Есть возможность использовать блок без третьего аргумента, тогда результат суммы будет заноситься в первый аргумент. Если используются константы, они автоматически преобразуются в вещественный тип.




（K1234） + ( D101,D100) → (D111,D110)

***Блок ESUB***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K

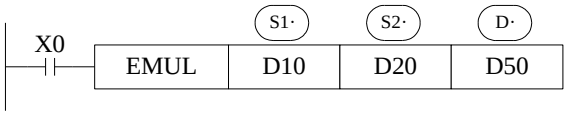



(D11,D10) － (D21,D20) → (D51,D50)

Данный блок производит арифметическое вычитание. Характеристики и принцип работы аналогичны с предыдущим блоком.

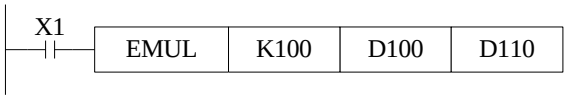
***Блок EMUL***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K




（D11，D10）×（D21,D20）→(D51,D50)

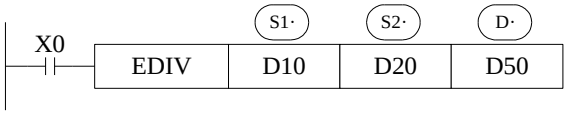
Данный блок производит умножение 2 вещественных значений. Если используются константы – они автоматически приводятся к вещественному типу.




(K100) × (D101,D100) → (D111,D110)

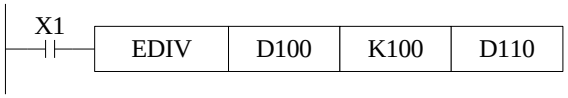
***<u>Блок EDIV</u>***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K




（D11,D10）÷（D21,D20）→（D51,D50）

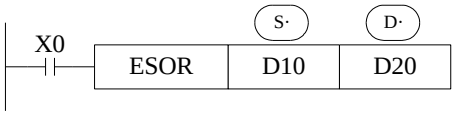
Данный блок производит деление аргумента S1 на S2 и заносит результат в аргумент D. Если параметр S2 =0, то произойдет ошибка и блок не выполнится.
Если используются константы, они приводятся к вещественному типу до выполнения данного блока.




(D101,D100) ÷ (K2346) →(D111,D110)

***<u>Блок ESQR</u>***

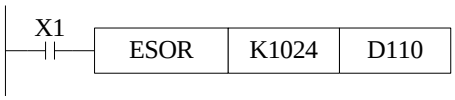
Аргументы: DX、DY、DM、DS、T、C、D、FD、K




(D11,D10)→（D21,D20）

Данный блок позволяет вычислить квадратный корень переданного значения. При использовании констант, они автоматически приводятся к вещественному типу

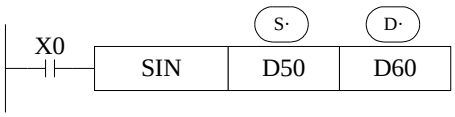
（K1024）→（D111，D110）




Если результат равен нулю, флаг нуля активен. Если в качестве аргумента передано отрицательное значение, произойдет ошибка и блок не выполнится.

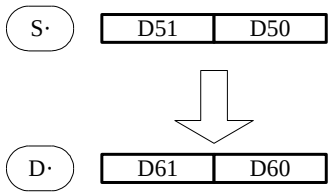
***<u>Блок SIN</u>***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K




(D51,D50) → (D61,D60)SIN

Данный блок рассчитывает синус значения, заданного в аргументе S (в радианах).

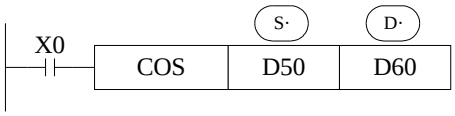



RAD переменная（угол×Π/180）
Заданная вещественным значением

SIN переменная
Вещественный тип

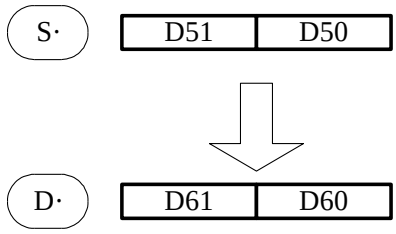
***<u>Блок COS</u>***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K




(D51,D50)RAD → (D61,D60)COS

Этот блок вычисляет математический косинус аргумента, заданного в S (в радианах).




RAD
переменная（угол×Π/180）
Вещественный тип

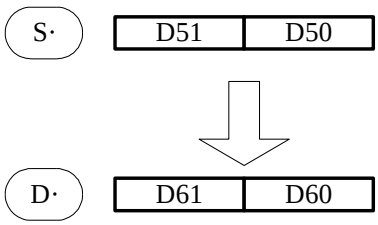
COS переменная
Вещественный тип

***<u>Блок TAN</u>***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K


(D51,D50)RAD → (D61,D60)TAN

ий тангенс аргумента, заданного в S (в радианах).




RAD переменная（угол×П/180）
Вещественный тип

TAN переменная
Вещественный тип

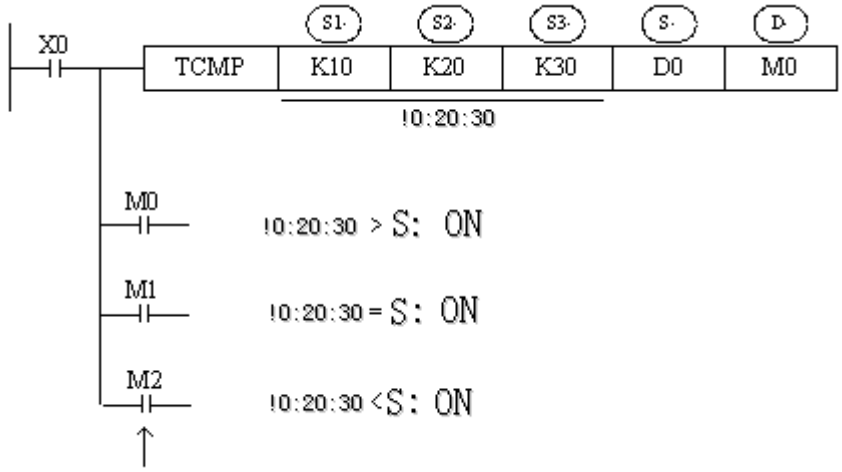
## 5.9 Операции со временем (часы, секунды, минуты)

| Обозначения | Функция |
|---|---|
| TCMP | Сравнение времен |
| TZCP | Сравнение области времен |
| TADD | Сложение времени |
| TSUB | Вычитание времени |
| TRD | Чтение времени с контроллера |
| TWR | Установка времени на контроллере |

***Блок TCMP***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K

Сравнивает введенные данные с установленным временем



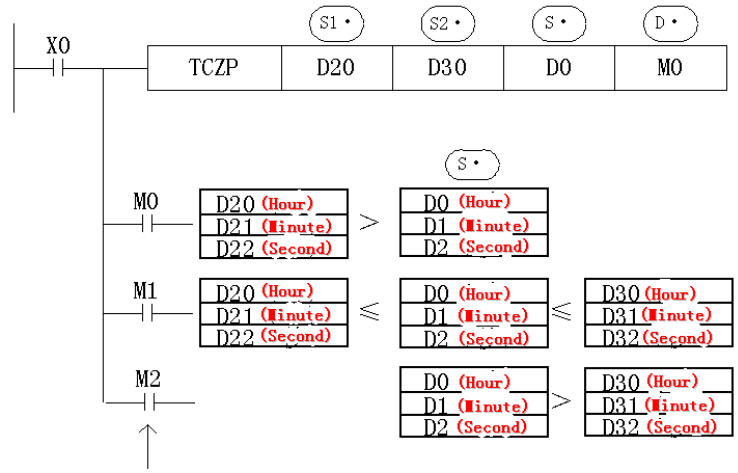

Результаты сравнения сохраняются даже при неработающем блоке TCMP. Данный блок сравнивает два блока. В данном примере:

S1 – часы D0 - часы
S2 – минуты и D1 - минуты
S3 – секунды D2 – секунды

Час должен быть в диапазоне 0-23
Минуты должны быть в диапазоне 0-59
Секунды должны быть в диапазоне 0-59

***Блок TZCP***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K
Данный блок сравнивает два определенных времени с текущим временем.



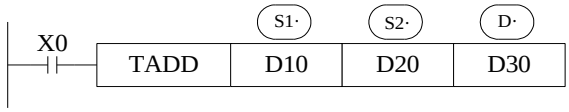

В блоке задается 3 времени (час, минута, секунда) для сравнения. Результаты начинаются с флага M0.

(S1·), (S1·)+1, (S1·)+2
(S2·), (S2·)+1, (S2·)+2
(S·), (S·)+1, (S·)+2
(D·), (D·)+1, (D·)+2

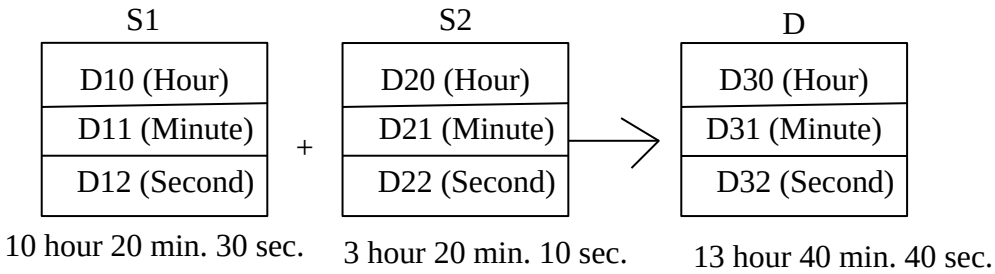
Временные блоки должны быть в рамках:
- час – 0-23
- минута 0-59
- секунда 0-59

***<u>Блок TADD</u>***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K




(D10, D11, D12)+（D20, D21, D22）→（D30, D31, D32）

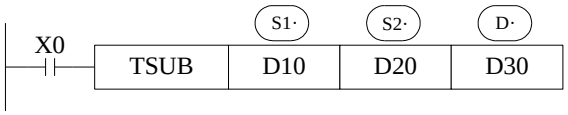



10 hour 20 min. 30 sec. 3 hour 20 min. 10 sec. 13 hour 40 min. 40 sec.

Данный блок производит сложение времени. В качестве аргументов в него передаются первые адреса 3-х временных блоков. Аргументы S1 и S2 складываются, и результат заносится в D. Если результат сложения больше 24 часов, то срабатывает флаг-регистр M8022, а результат будет приведен к 24 часам. Когда результат сложения равен 0 (0 часов, 0 минут, 0 секунд) – установится нулевой флаг.

***Блок TSUB***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K



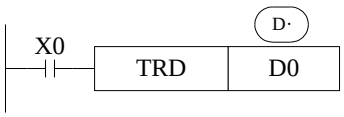

(D10, D11, D12)－（D20, D21, D22）→（D30, D31, D32）

| S1 | | S2 | | D |
|---|---|---|---|---|
| D10 (Hour) | | D10 (Hour) | | D10 (Hour) |
| D11 (Minute) | – | D11 (Minute) | = | D11 (Minute) |
| D12 (Second) | | D12 (Second) | | D12 (Second) |
| 10 hour 20 min. 30 sec. | | 3 hour 20 min. 10 sec. | | 7 hour 0 min. 20 sec. |

Данный блок вычисляет разницу двух времен. Если в результате вычитания получится меньше 0 часов, 0 минут и 0секунд, то сработает флаг M8021, результатом будет 0 часов, 0 минут, 0 секунд. Если результат равен 0, то сработает нулевой флаг.
Данные должны входить в следующие границы:
- часы -0-23
- минуты 0-59
- секунды 0-59

***Блок TRD***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K

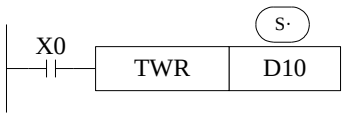



Данный блок считывает с контроллера текущие время и дату, и записывает их в 7 регистров, начиная с D.
На контроллере текущие время – дата находятся в регистрах D8013~D8019.

| Device | Meaning | Values | Device | Meaning |
|---|---|---|---|---|
| D8018 | Year | 1~99 | D0 | Year |
| D8017 | Month | 1~12 | D1 | Month |
| D8016 | Date | 1~31 | D2 | Date |
| D8015 | Hours | 0~23 | D3 | Hours |
| D8014 | Minutes | 0~59 | D4 | Minutes |
| D8013 | Seconds | 0~59 | D5 | Seconds |
| D8019 | Day | 0 (Sat.)~6 (Sun.) | D6 | Day |

***Блок TWR***

Аргументы: DX、DY、DM、DS、T、C、D、FD、K




Данный блок устанавливает (настраивает) текущие дату – время на контроллере, используя регистры, которые начинаются с адреса, указанного в аргументе S.

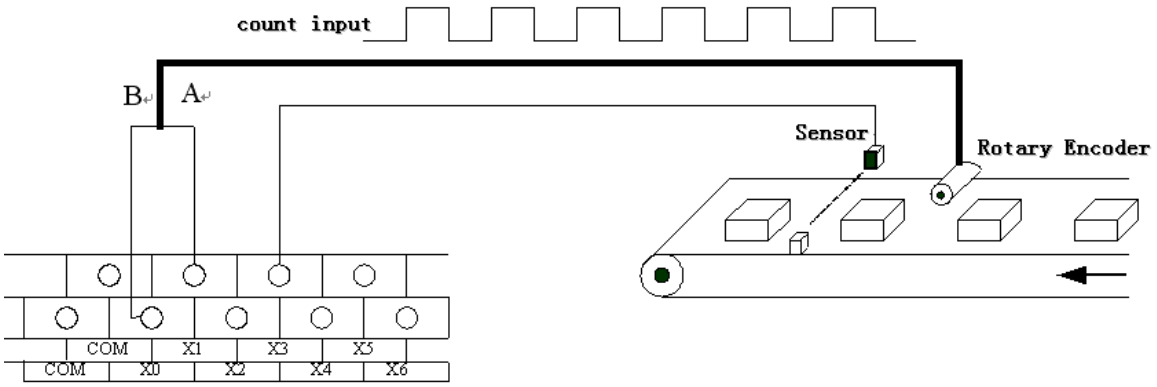
| Device | Meaning | Values | Device | Meaning |
|---|---|---|---|---|
| D0 | Year | 1~99 | D8018 | Year |
| D1 | Month | 1~12 | D8017 | Month |
| D2 | Date | 1~31 | D8016 | Date |
| D3 | Hours | 0~23 | D8015 | Hours |
| D4 | Minutes | 0~59 | D8014 | Minutes |
| D5 | Seconds | 0~59 | D8013 | Seconds |
| D6 | Day | 0 (Sat.)~6 (Sun.) | D8019 | Day |

# ***Специальные функции***

6.1 Высокоскоростные счетчики
6.2 Импульсные выходы
6.3 Инструкции модбас
6.4 Свободный формат передачи данных
6.5 PWM модулятор импульсов
6.6 Испытание с частотой
6.7 Точное время
6.8 Функции прерывания
6.9 CANBUS протокол связи (для серии XC5)

## 6.1 Высокоскоростные счетчики

Все ПЛЦ серии XC поддерживают функции высокоскоростного счета. В зависимости от вида счетчика, функции делятся на режимы приращения, с выбором направления (инкремент, декремент), для двухфазных счетчиков. Максимальная частота 200 КГц.




## Назначенные входы для высокоскоростных счетчиков

В таблице ниже дается перечень высокоскоростных счетчиков для разных ПЛК

| Модель ПЛК | | Высокоскоростные счетчики | | |
|---|---|---|---|---|
| | | Инкрементирующий тип | Импульс+ направления счета | двухфазный |
| XC3 серия | XC3-14 | 4 | 2 | 2 |
| | XC3-24/XC3-32 | 5 | 3 | 3 |
| | XC3-48/XC3-60 | 4 | 2 | 2 |
| XC5 серия | XC5-32 | 2 | 1 | 1 |
| | XC5-48/XC5-60 | 5 | 3 | 3 |

Для определения нужного счетчика сошлитесь на следующую таблицу.
Если вход X на контроллере не применяется как вход высокоскоростного счетчика, он может применяться как обычный вход.

[U]---вход для счета импульсов [Dir]---направление счета (OFF тогда +, ON тогда -)
[A]---A первая входная фаза [B]---B вторая входная фаза

**XC3-48、XC3-60 PLC модели**

| | Инкрементирующий тип | | | | | | | | | | Импульс+ направления счета | | | | | двухфазный | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | C600 | C602 | C604 | C606 | C608 | C610 | C612 | C614 | C616 | C618 | C620 | C622 | C624 | C626 | C628 | C630 | C632 | C634 |
| X000 | U | | | | | | | | | | U | | | | | B | | |
| X001 | | | | | | | | | | | Dir | | | | | A | | |
| X002 | | U | | | | | | | | | | U | | | | | B | |
| X003 | | | | | | | | | | | | Dir | | | | | A | |
| X004 | | | U | | | | | | | | | | | | | | | |
| X005 | | | | U | | | | | | | | | | | | | | |

**XC3-24、XC3-32 и XC5-48、XC5-60 PLC модели**

| | Инкрементирующий тип | | | | | | | | | | Импульс+ направления счета | | | | | двухфазный | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | C600 | C602 | C604 | C606 | C608 | C610 | C612 | C614 | C616 | C618 | C620 | C622 | C624 | C626 | C628 | C630 | C632 | C634 |
| X000 | U | | | | | | | | | | U | | | | | B | | |
| X001 | | | | | | | | | | | Dir | | | | | A | | |

**XC3-14 PLC модель**

| | Инкрементирующий тип | | | | | | | | | | Импульс+ направления счета | | | | | двухфазный | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | C600 | C602 | C604 | C606 | C608 | C610 | C612 | C614 | C616 | C618 | C620 | C622 | C624 | C626 | C628 | C630 | C632 | C634 |
| X000 | U | | | | | | | | | | U | | | | | B | | |
| X001 | | | | | | | | | | | Dir | | | | | A | | |
| X002 | | U | | | | | | | | | | | | | | | | |
| X003 | | | U | | | | | | | | | | | | | | | |
| X004 | | | | | | | | | | | | | U | | | | B | |
| X005 | | | | U | | | | | | | | | Dir | | | | A | |

**XC5-32 PLC модель**

| | Инкрементирующий тип | | | | | | | | | | Импульс+ направления счета | | | | | двухфазный | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | C600 | C602 | C604 | C606 | C608 | C610 | C612 | C614 | C616 | C618 | C620 | C622 | C624 | C626 | C628 | C630 | C632 | C634 |
| X000 | U | | | | | | | | | | U | | | | | B | | |
| X001 | | | | | | | | | | | Dir | | | | | A | | |
| X002 | | | | | | | | | | | | | | | | | | |
| X003 | | U | | | | | | | | | | | | | | | | |

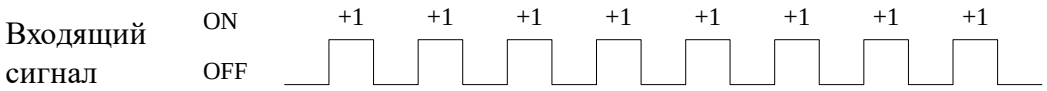
Высокоскоростные счетчики, которые не поддерживают 4 частоты в двухфазных счетчиках, перечислены в таблице ниже

| PLC модель | | Высокоскоростные счетчики с 4-мя частотами |
|---|---|---|
| XC3 серия | XC3-14 | C630 |
| | XC3-24/ XC3-32 | C632 |
| | XC3-48/ XC3-60 | C630 |
| XC5 серия | XC5-32 | - |
| | XC5-48/ XC5-60 | C632 |

# Тип входных для высокоскоростных счетчиков разных видов

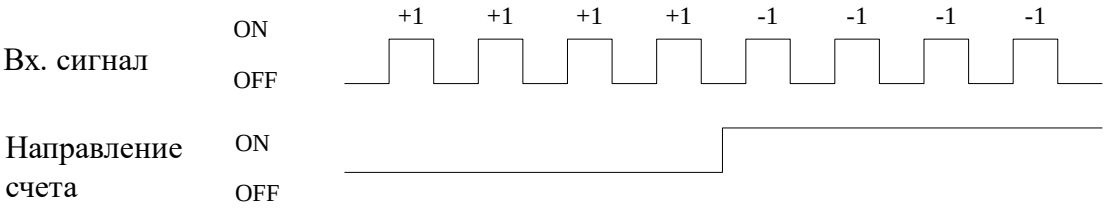
Инкрементирующий тип:

При данной модификации значение счетчика увеличивается при поступлении сигнала на вход контроллера:



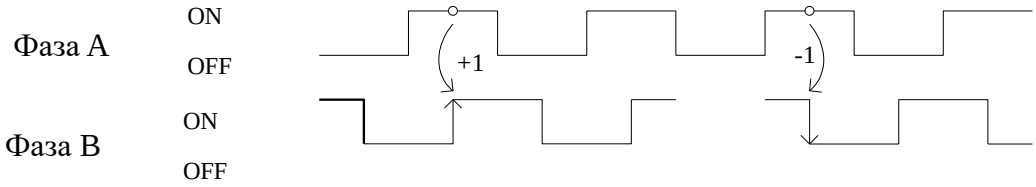

Импульс + направления счета:

При данной модификации используются 2 входа – один для подачи импульсов, второй для направления счета




Двухфазный счетчик:

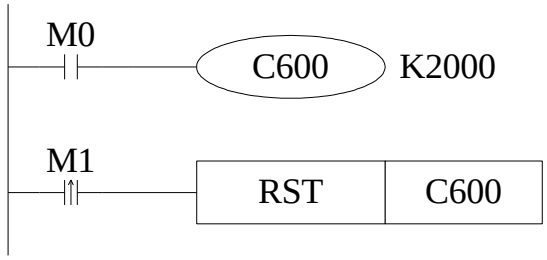
У двухфазного счетчика значение может инкрементироваться или декрементироваться в зависимости от разности сигналов 2-х фаз




Максимальное значение счетчика:

Значение счетчика находится в пределах : K-2,147,483,648 ~ K+2,147,483,647. Если значение выходит за пределы верхней границы, то следующее значение будет равно -2,147,483,648, если за нижнюю границу - 2,147,483,648.

Сброс.

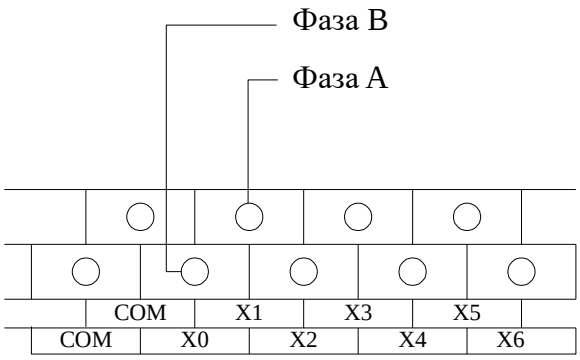



В данном примере при активации катушки M0 запустится счетчик C600 входа X0. Когда M1 сменит свое значение с 0 на 1, счетчик C600 будет выключен, а его значение обнулится.

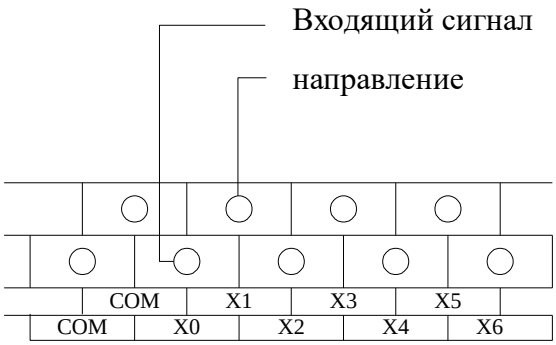
Подсоединение к входному терминалу (вход анна контроллере).

В примере рассмотрено подключение счетчика C600.

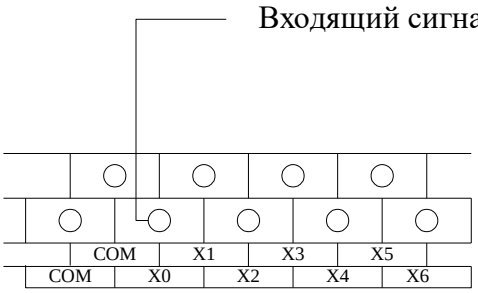
Двухфазный счетчик




импульс + направления счета



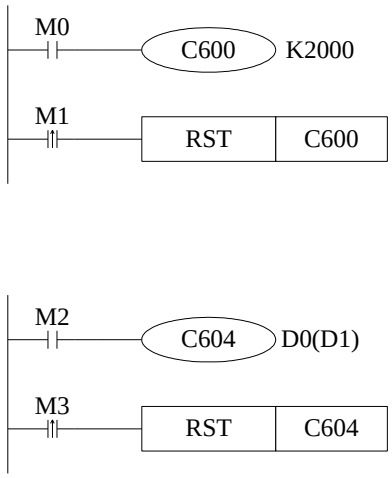

инкрементирующий тип




Примеры программы

Ниже, мы на примере XC3-60 PLC рассмотрим как в программе обработать сигналы с высокоскоростного счетчика ：

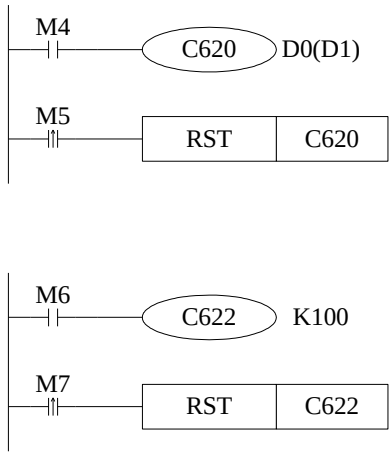
**Инкрементирующий тип**




- когда M0 будет включена, счетчик C600 начнет принимать сигналы с X0
- когда M1 сменит значение с 0 на 1 выполнится инструкция RST и счетчик будет сброшен

- Когда M2 включится, начнет работать счетчик C604,привязанный на вход X4. В данном примере величина значения счетчика устанавливается регистром данных и используется косвенная адресация. Катушка M3, при срабатывании сбрасывает счетчик C604

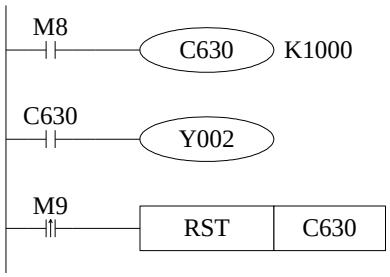
**Импульс + направления счета**




- Когда включится катушка M4, счетчик C620 начнет принимать сигналы с входа X0, в зависимости от состояния входа X1 будет производиться инкремент/ декремент значения счетчика. Если на входе нет сигнала - значение увеличивается, если есть – уменьшается.
- Когда сработает катушка M6, счетчик C622 начнет принимать сигналы с входа X2, и в зависимости от значения X3 будет увеличивать/ уменьшать значение. Если X3 – 0, значение инкрементируется, X3 – 1 – декрементируется.

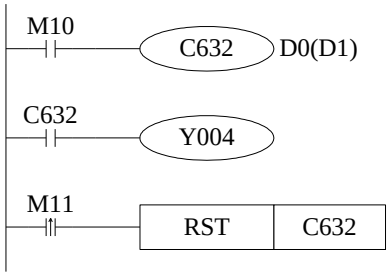
**AB двухфазный тип**

Двухфазный счетчик может инкрементировать/ декрементировать значение в зависимости от скачков (уровней, прыжков) фаз A и B.




- Когда M8 сработает, C630 начинает считывать импульсы с X000 (B фаза), X001(A фаза) через перерывы.
- Если сработала M9, выполнится сброс счетчика

- Когда текущее значение счетчика C630 превысит заданное (1000), установится выход Y2, если значение счетчика меньше заданного, выход не будет работать



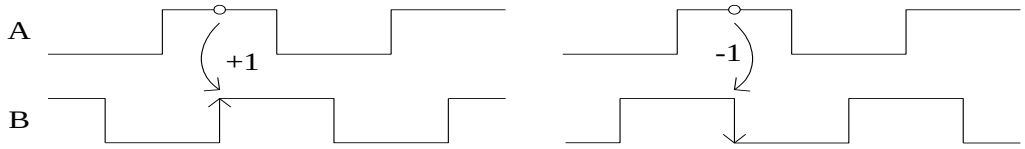

- Когда M10 активна, C632 начинает счет. Сигналы принимаются с X002 (B фаза)、X003(A фаза).
- Катушка M11 сбрасывает счетчик
- Если текущее значение превышает указанное, то Y4 включается, если нет – то Y4 выключено.

Если фаза A находится в промежутке между OFF→ON и если на фазе B нет сигнала, производится инкрементирование значения счетчика, если на фазе B единица, то производится декрементирование.

**Частота срабатывания**

Высокоскоростных счетчиков, которые поддерживают 2 типа частоты срабатывания, два. По умолчанию они настроены на 4-х частотное срабатывание. Счетчики, поддерживающие 2 типа, показаны ниже:

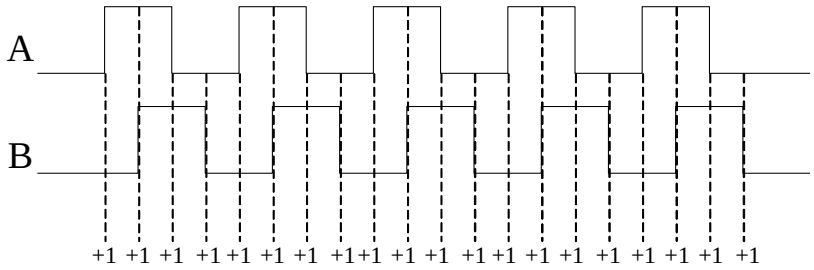
Частота срабатывания 1, двухфазный счетчик имеет следующий вид:



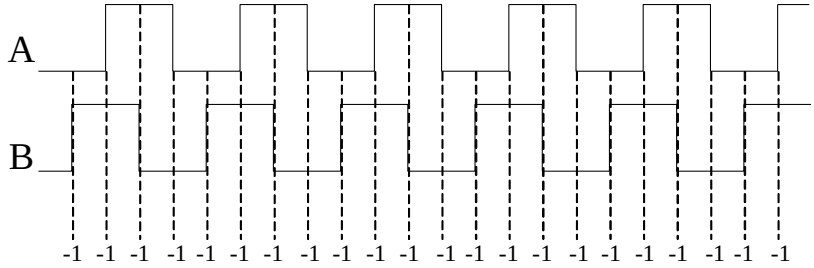

Инкремент значения  декремент значения

Частота срабатывания 4; для данной модификации счетчик имеет вид:

Инкремент значения счетчика



Декремент счетчика



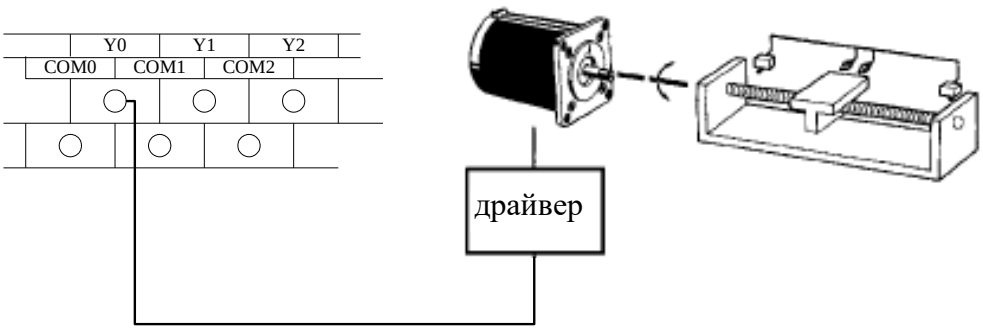

Переключение работы счетчиков по частоте срабатывания

| | | |
|---|---|---|
| FD8241 | Частота срабатывания для C630 | 1 is 1 time frequency,<br>4 is 4 times frequency, |
| FD8242 | Частота срабатывания для C632 | 1 is 1 time frequency,<br>4 is 4 times frequency |
| FD8243 | Частота срабатывания для C634 | 1 is 1 time frequency,<br>4 is 4 times frequency |

# 6.2 Выходные импульсы

Контроллеры серии XC3 и XC5 имеют 2 импульсных выхода. Используя инструкции (блоки) программирования, вы можете реализовать одиночный (или несколько) импульсный выход с увеличивающейся/ уменьшающиеся частотой, с конкретным временем достижения этой частоты, подачей положительных/ отрицательных импульсов. Максимальная частота 400 КГц.

Шаговый двигатель



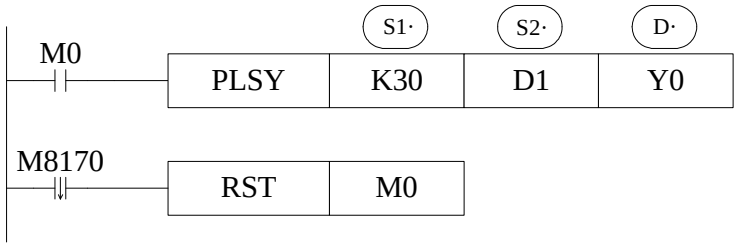

Примечания:
1) если вы хотите использовать импульсные выхода, вы должны использовать транзисторные ПЛК или транзисторно-релейные XC3-14T-E, XC3-60RT-E.
2) XC5-32 ПЛК имеет 4 импульсных выхода Y0、Y1、Y2、Y3.

Типы и обработка высокоскоростных выходов в программе.

Один импульсный выход без возможности установки времени разгона/ торможения.
- частота 0-400 КГц
- выхода Y0-Y1
- тип выходного сигнала – продолжительный или ограниченный
- максимальное число импульсов:
  16 битная инструкция 0~K32767
  32 битная инструкция 0~K2147483647
- инструкции (блоки) PLSF, PLSY.

PLSY – генерирует заданное число импульсов с заданной частотой
PLSF – генерирует продолжительные импульсы с переменной частотой

PLSY инструкция:




Также для генерации ограниченного числа импульсов используется 32-х битная инструкция DPLSY

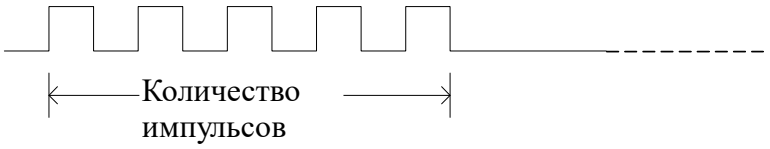
(S1·) Заданная частота. Аргументы: K、TD、CD、D、FD

(S2·) Заданное количество импульсов. Аргументы: K、TD、CD、D、FD

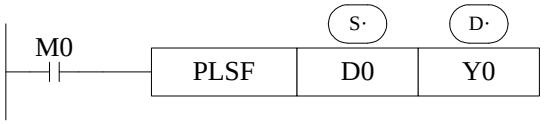
(D·) Назначенный выход (он посылает импульсы), может быть Y000 или Y001

Когда M0 включится, блок PLSY сгенерирует количество импульсов, заданное в параметре D1, с частотой 30 Гц в выход Y0. Когда текущее количество импульсов сравнится с заданным, M8170 выключится, что приведет к остановке работы блока PLSY, и сбросу катушки M0.

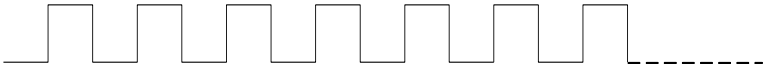
Ограниченное количество импульсов




- **PLSF инструкция:**

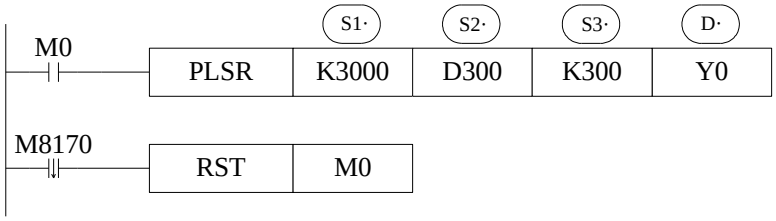


- Генерирует последовательные импульсы с переменной частотой
- 32-х битный аналог [DPLSF].
- Параметр частоты. Аргументы: K、TD、CD、D、FD
  Границы: 200~400KHz (если установленная частота меньше 200Hz, устанавливается 200Hz)
- установка Y порта генерирующего импульсы, могут использоваться Y0 или Y1
- С непостоянной частотой указанной в D0 импульсы подаются на Y0
- Импульсы аккумулируются в регистре D8170

Продолжительные импульсы 

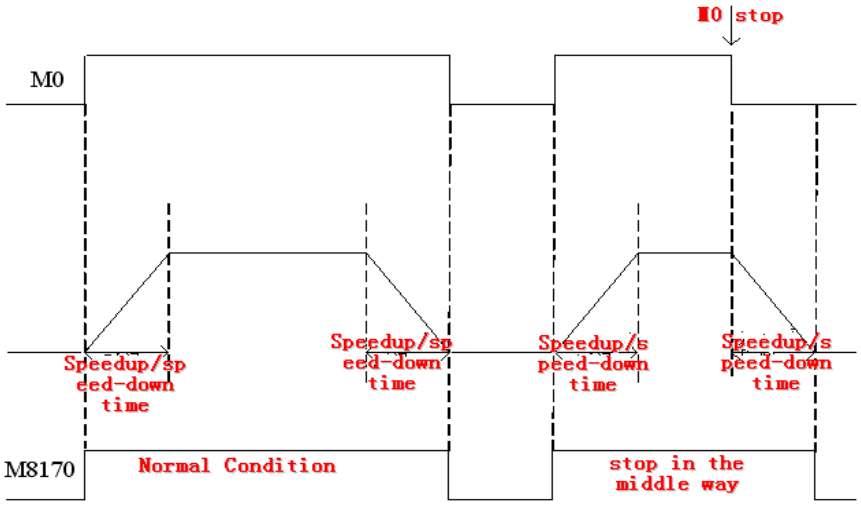
Один импульсный выход с возможностью установки времени разгона/ торможения
1) диапазон частот 0~400KHz
2) время разгона/ торможения приблизительно 5000 мс
3) выходы Y0 или Y1
4) выходная характеристика: ограниченное количество импульсов
5) количество импульсов:
   - 16-ти битная инструкция 0~K32767
   - 32-х битная инструкция 0~K2147483647

6) блок (инструкция) PLSR, она генерирует заданное количество импульсов, с заданной частотой и заданным временем разгона.

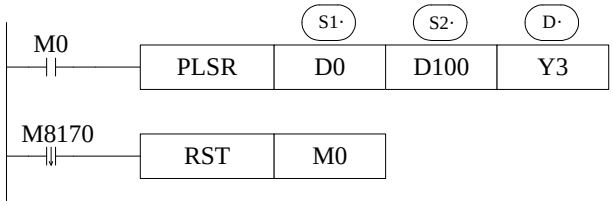



- Генерация определенного числа импульсов с заданной частотой; использование 32 битной инструкции [DPLSR]。
  Заданная частота. Аргументы: K、TD、CD、D、FD
  Общее количество импульсов. Аргументы: K、TD、CD、D、FD
  Время разгона/ установки. Аргументы: K、TD、CD、D、FD
  Установленный выход, может быть Y0 или Y1

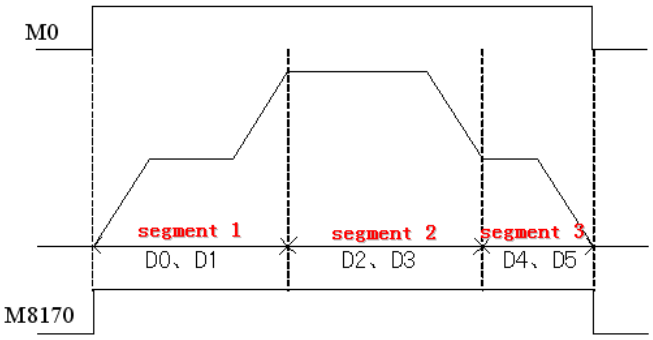
Когда катушка M0 включиться, начнет работать блок PLSR с заданной частотой, и временем разгона. При работающем блоке катушка M8170 будет находиться во включенном состоянии. Когда текущее количество импульсов сравняется с заданным, выдача импульсов прекратится, катушка M8170 выключится, что приведет к сбросу катушки M0. Смотрите график ниже. Если до достижения заданного количества импульсов сбросить катушку M0, катушка M8170 тоже сбросится.




Выдача сегментов импульсов в одну фазу



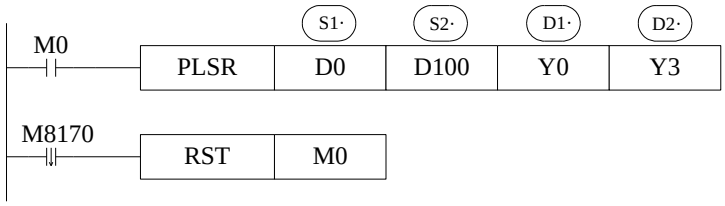

- Данная инструкция генерирует заданное количество импульсов с заданной частотой.
  Область начинается с Dn или FDn стартового адреса. В данном примере, D0 устанавливает частоту импульсов для первого сегмента, D1 количество импульсов первого сегмента, D2 устанавливает частоту импульсов для 2-го сегмента, D3 устанавливает количество импульсов для 2-го сегмента, ……если значение переменных Dn, Dn+1 равно 0, это является окончанием сегментов.
  Вы можете использовать 24 сегмента. Аргументы: D, FD
  Время разгона/ торможения. Она используется при разгоне первого сегмента, 2-го до достижения заданной частоты, короче используется для всех сегментов, и в итоге для остановки. Аргументы: K, TD, CD, D, FD
  Указывается выход, могут использоваться Y0 или Y1
- При использовании 32-х битной инструкции DPLSR, D0、D1 устанавливают частоту сегмента 1、D2、D3, устанавливают количество импульсов сегмента 1, D4、D5 устанавливают частоту сегмента 2、D6、D7 устанавливают количество импульсов для сегмента 2……




Двойной импульсный выход с заданной скоростью разгона/ торможения.

- Частота: 0~400KHz
- Время разгона/ торможения: максимум 5000ms
- выходы: Y0 или Y1
- выход для направления: любой Y
- выходная модификация: ограниченное число импульсов
- количество импульсов: 16-ти битная инструкция: 0~K32767
  32-х битная инструкция: 0~K2147483647

- инструкция (блок): PLSR



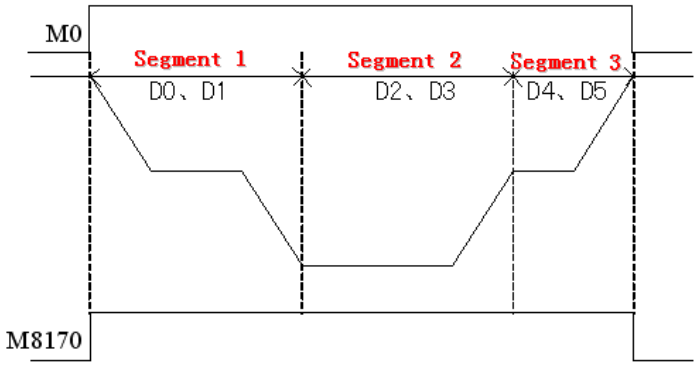

- Генерируется определенное количество импульсов с заданной частотой, временем разгона/ торможения, сигналом направления (например, шаговый двигатель).

S1· Область включает в себя Dn или FDn стартовые адреса. В данном примере, D0 устанавливает максимальную частоту сегмента 1, D1 количество импульсов для сегмента 1. D2 устанавливает максимальную частоту для сегмента 2, D3 устанавливает количество импульсов сегмента 2, …… если Dn、Dn+1 равны 0, этот сегмент является конечным. Вы можете использовать 24 сегмента. Аргументы: D, FD.

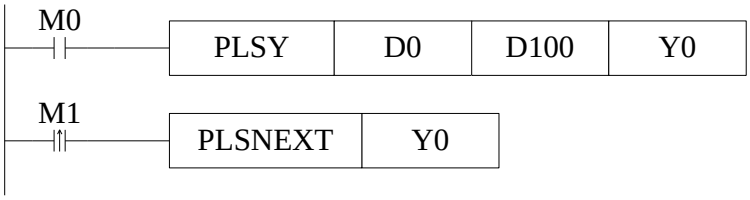
S2· Время разгона/ торможения используется для достижения максимальной частоты с момента старта. Также используется в каждом сегменте и для торможения в конце последнего сегмента. Аргументы: K、TD、CD、D、FD

D1· Выход для подачи импульсов, могут использоваться Y000 или Y001

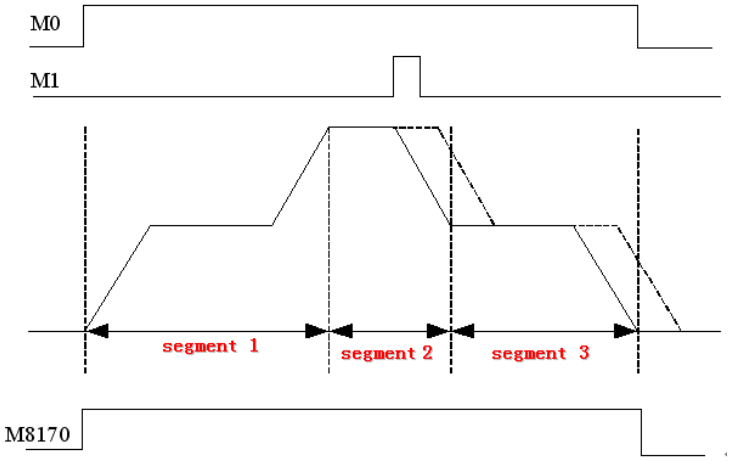
D2· Выход для установки сигнала направления (может быть любой свободный Y). Данный параметр устанавливается в зависимости от знака количества импульсов в первом сегменте, если - +, то Y включен, иначе, если минус – выключен.




**Переключатель сегмента импульсов [PLSNEXT]**

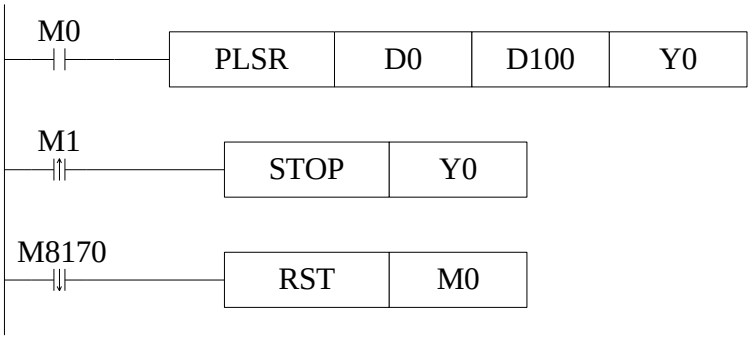



Если выходной сигнал достигает максимальной частоты текущего сегмента, выход стабилен, если M1 сменит свое состояние с 0 на 1, будет введен следующий выходной импульс с заданным временем разгона/ торможения.
Во время процесса разгона/ торможения данная инструкция не может выполняться.



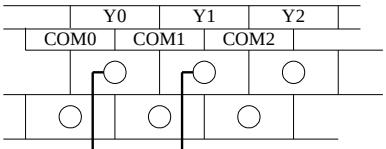

----------(Прерывистая линия) показывает реальную кривую выходного импульса.

**Немедленная остановка подачи импульсов [STOP]**




- Если M0 сменит свое значение с 0 на 1, PLSY активируется и Y000 начнет генерировать импульсы с заданной частотой D0, D001 количеством импульсов, D100 временем разгона/ торможения, когда количество импульсов достигнет заданного количества сработает флаг остановки (стопа) M8170. если сработает катушка M0, сработает инструкция STOP, и Y0 немедленно прекратит генерацию импульсов.

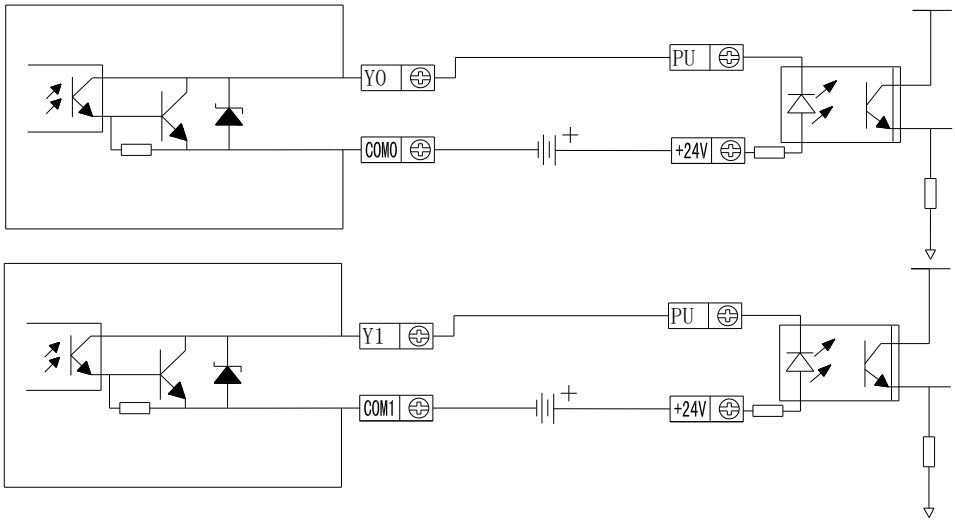
- **Подсоединение выходов**




Выход Y0: порт импульсов 0 (одна фаза)
Выход Y1: порт импульсов 1 (одна фаза)

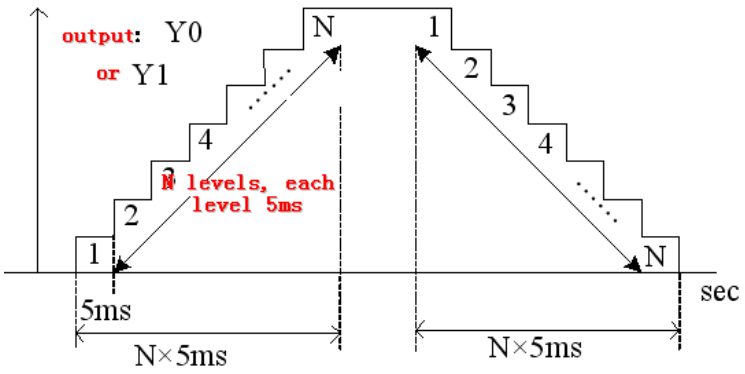
Ниже, на графике, показано подсоединение шагового двигателя к контроллеру

Контроллер Драйвер шагового двигателя




Примечания.

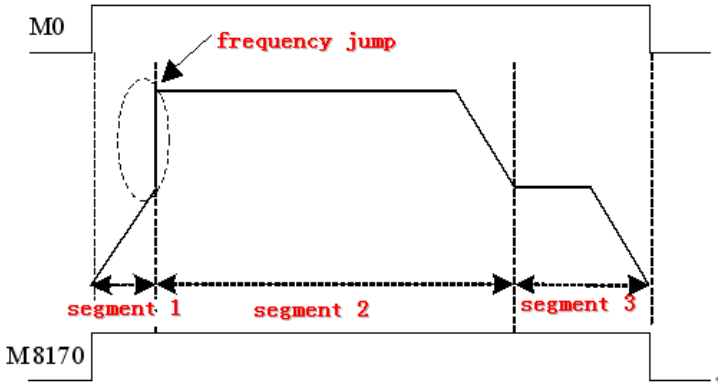
1. концепция шагового увеличения/ уменьшения частоты




В процессе разгона/ торможения автоматически рассчитывается количество шагов для достижения максимальной частоты для сегмента. Данная величина является постоянной. Максимальный шаг – 15К (не зависимо – происходит торможение или разгон). Когда значение превышает 15, принимается значение равное 15К. минимальный шаг частоты –

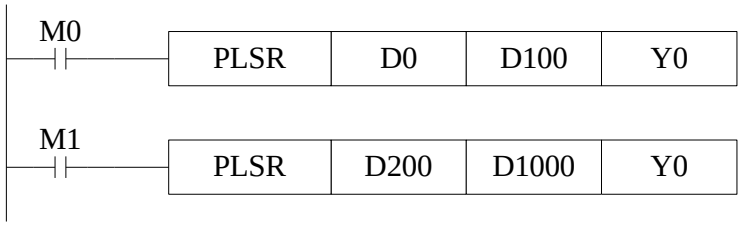
10Гц, если значение меньше 10, то рассчитывается 10 Гц.

**2.Частотный скачек в сегментном выходном сигнале**



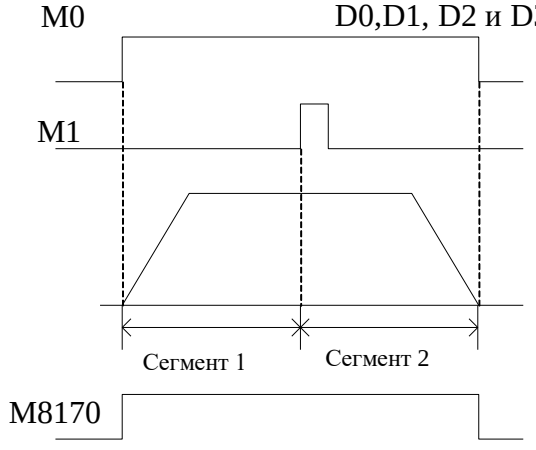

В процессе подачи сегмента импульсов, если генерация импульсов уже началась, но не успела достичь максимальной частоты для данного сегмента, то при переходе на следующий сегмент произойдет частотный скачек (данная ситуация может произойти в случае большого значения частоты и маленького количества импульсов для данного сегмента).
Что бы избежать скачка, не устанавливайте слишком маленькое значение параметра разгона/ торможения.

3. В листинге одной программы нельзя использовать два блока PLSR (PLSY, DPLSY…) с одним и тем же выходом, выдающем импульсы.
Приведенная ниже инструкция нерабочая




Фиксированная остановка.
Совместное использование блоков PLSR и PLSNEXT позволяют получить функцию остановки с фиксированной длиной.

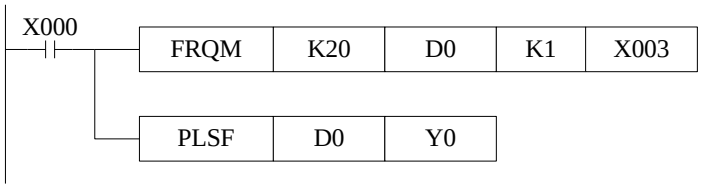
В представленной программе регистры D0,D1, D2 и D3 устанавливают параметры 2-х сегментов.




включение катушки M1 приведет к реализации функции

остановки с фиксированной длиной.

Проверка отношений
Выходящий частотный импульс Y0 приравнивается к значению входа X3. Если тестируется входной импульсный канал X3, частота импульсного выхода приравнивается Y0.




Спецификация специальных катушек и регистров импульсных выходов.

Перечень катушек

| ID | Высокочастотный Сигнал ID | Функция | Расшифровка |
|---|---|---|---|
| M8170 | импульс_1 | Установочный флаг | 1 при рабочем блоке до достижения заданного количества импульсов |
| M8171 | | 32-х битный флаг перегрузки | 1 при рабочем блоке до достижения заданного количества импульсов |
| M8172 | | Флаг направления | 1 когда выход направления включен |
| M8173 | импульс_2 | Установочный флаг | 1 при рабочем блоке до достижения заданного количества импульсов |
| M8174 | | 32-х битный флаг перегрузки | 1 при рабочем блоке до достижения заданного количества импульсов |
| M8175 | | Флаг направления | 1 когда выход направления включен |
| M8176 | импульс_3 | Установочный флаг | 1 при рабочем блоке до достижения заданного количества импульсов |
| M8177 | | 32-х битный флаг перегрузки | 1 при рабочем блоке до достижения заданного количества импульсов |
| M8178 | | Флаг направления | 1 когда выход направления включен |
| M8179 | импульс_4 | Установочный флаг | 1 при рабочем блоке до достижения заданного количества импульсов |
| M8180 | | 32-х битный флаг перегрузки | 1 при рабочем блоке до достижения заданного количества импульсов |
| M8181 | | Флаг направления | 1 когда выход направления включен |

Некоторые специальные регистры

| ID | Высокочастотный Сигнал ID | Функция | расшифровка |
|---|---|---|---|
| D8170 | Импульс_1 | Младшие 16-ть битов, аккумулирующие количество импульсов | |
| D8171 | | Старшие 16-ть битов, аккумулирующие количество импульсов | |
| D8172 | | Текущий сегмент (номер сегмента) | |
| D8173 | Импульс_2 | Младшие 16-ть битов, аккумулирующие количество импульсов | |
| D8174 | | Старшие 16-ть битов, аккумулирующие количество импульсов | |
| D8175 | | Текущий сегмент (номер сегмента) | |
| D8176 | Импульс_3 | Младшие 16-ть битов, аккумулирующие количество импульсов | |
| D8177 | | Старшие 16-ть битов, аккумулирующие количество импульсов | |
| D8178 | | Текущий сегмент (номер сегмента) | |
| D8179 | Импульс_4 | Младшие 16-ть битов, аккумулирующие количество импульсов | |
| D8180 | | Старшие 16-ть битов, аккумулирующие количество импульсов | |
| D8181 | | Текущий сегмент (номер сегмента) | |
| D8190 | импульс_1 | Младшие 16-ть битов, аккумулирующие количество импульсов | |
| D8191 | | Старшие 16-ть битов, аккумулирующие количество импульсов | |
| D8192 | импульс_2 | Младшие 16-ть битов, аккумулирующие количество импульсов | |
| D8193 | | Старшие 16-ть битов, аккумулирующие количество импульсов | |
| D8194 | импульс_3 | Младшие 16-ть битов, аккумулирующие количество импульсов | |
| D8195 | | Старшие 16-ть битов, аккумулирующие количество импульсов | |
| D8196 | импульс_4 | Младшие 16-ть битов, аккумулирующие количество импульсов | |

## 6.3 Коммуникационные функции

XC3-PLC、XC5-PLC позволяют вам подсоединять технические устройства и сеть. Они не только поддерживают простую сеть (протокол коммуникации Модбас, свободный протокол), но поддерживают и более сложную сеть. XC3-PLC、XC5-PLC имеют коммуникационные порты, которые позволяют связываться с другими устройствами (принтеры, инструменты), которые работают на собственных протоколах.
XC3-PLC、XC5-PLC контроллеры поддерживают протокол модбас, а XC5-PLC также поддерживают CANBUS.

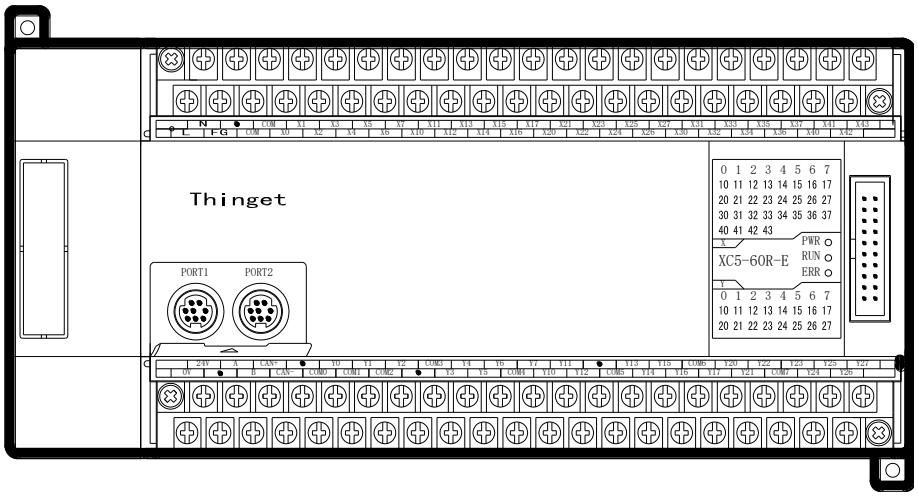
**COM порт**

В ПЛК серии XC3 имеется 2 порта(Port1、Port2) , а в серии XC5 - 3. Также в серии XC5 есть CAN COM порт.

COM 1 (порт1) – это порт для программирования, он используется для заливки программы на контроллер и для соединения с другими инструментами. Параметры (baud rate, data bit etc.) зафиксированы и не могут быть сброшены.

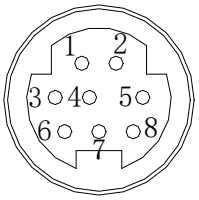
COM 2 (порт 2). Этот коммуникационный порт используется для заливки программы и для подсоединения устройств. Параметры (baud rate, data bit etc.) этого COM порта могут быть сброшены и установлены программно.

Блок расширения BD , может подключаться к другому COM порту. Этот COM порт может быть RS232 и RS485.




### 1、RS232 COM порт

**График разъемов COM 1（Port1）：**

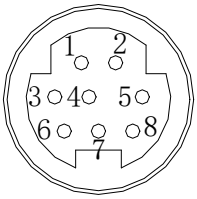



2：PRG
4：RxD
5：TxD
6：VCC
8：GND

Про RS485 COM порт, A является “+” сигналом、B это “-“ сигнал.
В XC серии PLC, с COM2 (порт2) можно использовать протоколы RS485 and RS232, но не одновременно

Mini Din 8 core socket (Hole)

- **The pin graph of COM 1（Port1）：**




4：RxD
5：TxD
8：GND

Mini Din 8 core socket (Hole)

CAN порт позволяет организовать CANbus соединение.
Более детально про функцию CAN соединения, можно прочесть из параграфа 6-8. CAN bus функция доступна ПЛК(XC5 series)”

**Параметры соединения**

| | |
|---|---|
| Станция | Номер Modbus станции: 1~254、255 (FF) это свободный протокол |
| Скорость передачи данных | 300bps~115.2Kbps |
| Размер сообщения | 8 bits данныхt、7 bits данных |
| Стоп биты | 2 stop bits、1 stop bit |
| Контроль | четный、нечетный、без проверки |

Параметры по умолчанию для порта COM 1：
Номер станции 1、скорость передачи 19200bps、8 бит данных、1 стоп бит、четный контроль

**Установка параметров**

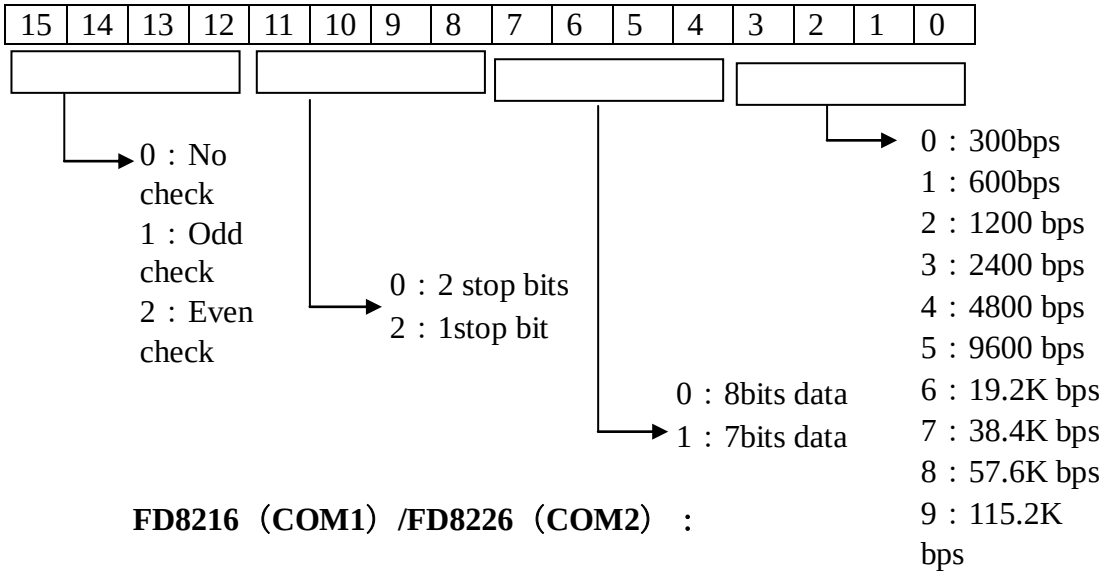
XC серия ПЛК позволяет устанавливать настройки COM порта

Как установить параметры соединения:

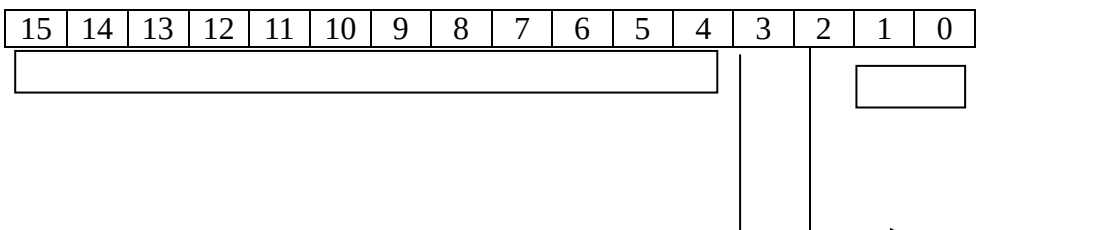
| | Регистр | Функция | Расшифровка |
|---|---|---|---|
| COM 1 | FD8210 | Метод соединения | 255 это свободный формат, **1~254 биты являются номером станции** |
| | FD8211 | Параметры связи | Baud rate, data bit, stop bit, check |
| | FD8212 | Таймаут посылки запроса | Время в мс, если равно 0 – то таймаута нет |
| | FD8213 | Время отклика на запрос | Время в мс, если равно 0, то нет |
| | FD8214 | Первый символ | Старшие 8 бит недействительны |
| | FD8215 | Последний символ | Старшие 8 бит недействительны |
| | FD8216 | Свободный формат связи | 8/16 bits cushion, with/without start bit, with/without stop bit |
| COM 2 | FD8220 | Метод соединения | 255 это свободный формат, **1~254 биты являются номером станции** |
| | FD8221 | Параметры связи | Baud rate, data bit, stop bit, check |
| | FD8222 | Таймаут посылки запроса | Время в мс, если равно 0 – то таймаута нет |
| | FD8223 | Время отклика на запрос | Время в мс, если равно 0, то нет |
| | FD8224 | Первый символ | Старшие 8 бит недействительны |
| | FD8225 | Последний символ | Старшие 8 бит недействительны |
| | FD8226 | Свободный формат связи | 8/16 bits cushion, with/without start bit, with/without stop bit |

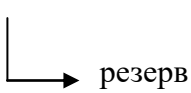
| | Методы установки параметров связи |
|---|---|

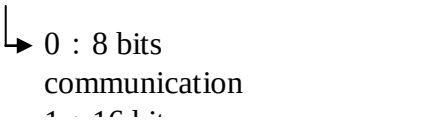
**FD8211（COM1）/FD8221（COM2）：**




**FD8216（COM1）/FD8226（COM2）：**






0: без стартового символа
1: со стартовым символом

0：без символа-конца
1：с символом-концом

### 6.3.1 Функции для связи по протоколу Модбас

**Функции связи**

XC серия поддерживает типы протокола Модбас – мастер и подчиненный.
Мастер: если выбран данный тип, то контроллер посылает запросы на другие контроллеры, устройства, а они отвечают на запросы.
Подчиненный: при данном типе контроллер отвечает на одиночные запросы других устройств-мастеров
По умолчанию XC-PLC настроен на Modbus slave.

Для данных контроллеров имеется следующая таблица связей:

Область катушек:

| Начальный адрес (**Dec.**) | M0 | X0 | Y0 | S0 | M8000 | T0 | C0 |
|---|---|---|---|---|---|---|---|
| Соответствие с Модбасом (**Hex.**) | 0 | 4000 | 4800 | 5000 | 6000 | 6400 | 6C00 |

Примечание: с таблицы видно взаимосвязи номера катушки и адреса этой катушки через протокол модбас. К примеру, если нам нужна катушка Y6, то в модбасе этот адрес будет равен 4800+6=Y6(4806).

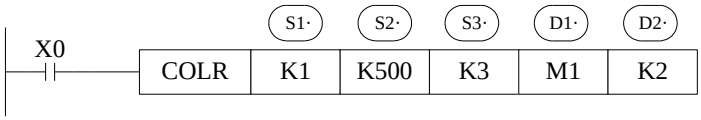
Область регистров данных:

| Начальный адрес (**Dec.**) | D0 | TD0 | CD0 | D8000 | FD0 | FD8000 |
|---|---|---|---|---|---|---|
| Соответствие с Модбасом (**Hex.**) | 0 | 3000 | 3800 | 4000 | 4800 | 6800 |

Примечание: расчет адреса аналогичен предыдущей таблице.

**Инструкции подключения**

**1.чтение катушек [COLR]**




- В протоколе модбас функция для чтения катушек 01H .
  Функция читает состояние катушек с назначенной станции.

S1· Номер удаленной станции. Аргументы: K、TD、CD、D、FD

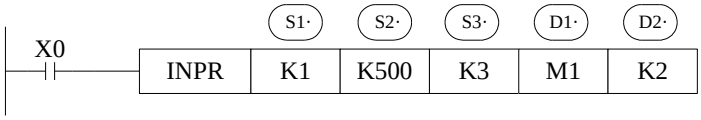
S2· Номер стартовой катушки. Аргументы: K、TD、CD、D、FD

S3· Количество катушек. Operands: K、TD、CD、D、FD

D1· Стартовый номер получателя на текущем контроллере. Аргумент: X、Y、M、S、T、C

D2· Номер порта. В рамках: K1~K2

**2、Чтение катушек входов [INPR]**



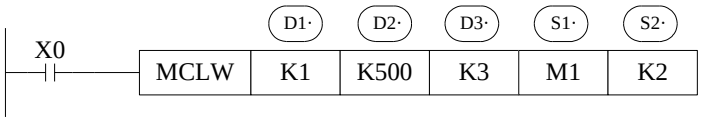

- В модбас функция для чтения катушек входов 02H
  Функция считывает состояние с заданной катушки, с заданной станции
  Номер удаленной станции. Аргументы: K、TD、CD、D、FD
  Номер удаленной стартовой катушки. Аргументы: K、TD、CD、D、FD
  Количество катушек. Аргументы: K、TD、CD、D、FD
  Стартовый номер катушки получателя на текущем контроллере.
  Аргументы: X、Y、M、S、T、C
  Номер порта. В рамках: K1~K2

Расшифровка инструкции: когда X0 включена, выполнится COLR или INPR инструкция. После окончания выполнения инструкций, установится соединение до последнего бита. Когда X0 выключена – операция не сработает. Если произойдет ошибка, запрос автоматически повторится. Если не ответа не будет в течении 10 раз, установится флаг ошибки. Пользователь должен убедится в работоспособности и реальности регистра.

**3、Установка одиночной катушки [COLW]**


- Функция записи единичной катушки в протоколе модбас находится под номером 05H
  Функция: устанавливает указанную катушку в указанной станции1.
  Номер удаленной станции. Аргументы: K、TD、CD、D、FD
  Номер удаленной катушки. Аргументы: K、TD、CD、D、FD
  Локальная катушка для записи прочитанного. Аргументы : X、Y、M、S、T、C
  Номер порта. Возможно: K1~K2

**4、Запись области катушек [MCLW]**




- В протоколе Модбас, функция для записи области катушек 0FH。
  Функция: устанавливает область удаленных катушек.

D1· Номер удаленной станции. Аргументы: K、TD、CD、D、FD

D2· Номер первой удаленной катушки. Аргументы: K、TD、CD、D、FD

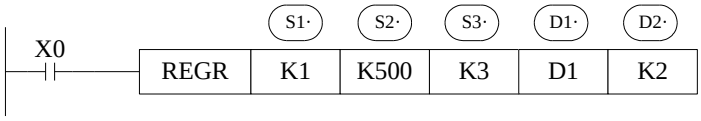
D3· Количество катушек. Аргументы: K、TD、CD、D、FD

S1· Номер первой локальной катушки. Аргументы: X、Y、M、S、T、C

S2· Номер порта. Возможно: K1~K2

Расшифровка инструкции: когда X0 включено, выполнится COLW или MCLW инструкция. После выполнения инструкции, установится соединение до последнего бита. Инструкция не работает, когда X0 выключен. Если произойдет ошибка присоединении – запрос автоматически повторится. После 10 раз установится флаг ошибки. Пользователь должен будет проверить качество соединения и реальность регистра.

**5、Чтение регистров данных [REGR]**




- Инструкция для чтения регистров данных в модбасе - 03H.
  Функция: читает указанные регистры данных с указанной удаленной станции.

S1· Номер удаленной станции. Аргументы: K、TD、CD、D、FD

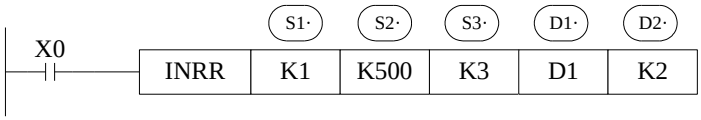
S2· Адрес первого удаленного регистра. Аргументы : K、TD、CD、D、FD

S3· Количество регистров. Аргументы : K、TD、CD、D、FD

D1· Номер первого локального регистра для записи значений. Аргументы: D

D2· Номер порта. Возможно: K1~K2

**6、Чтение входящих регистров [INRR]**




- Функция для чтения входящих регистров в модбасе - 04H.

Функция: читает состояние удаленных входящих регистров на указанной станции


Номер удаленной станции. Аргументы: K、TD、CD、D、FD

Стартовый номер удаленных регистров. Аргументы: K、TD、CD、D、FD

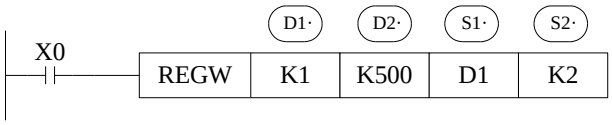
Количество регистров. Аргументы: K、TD、CD、D、FD

Адрес первого регистра получателя. Аргумент: D

Номер порта. Возможно: K1~K2

Расшифровка инструкции: когда X0 включен, выполняется REGR или INRR инструкция. После выполнения инструкции, устанавливается заключающий бит. Операция не выполнится, когда X0 выключен. Если произошла ошибка, сообщения отправится заново автоматически. После 10 раз неудачных запросов установится флаг ошибки. Пользователь должен проверить качество связи и достоверность читаемых регистров.

**7、Запись одного регистра [REGW]**




- Функция для записи одного регистра данных в модбасе - 06H

Функция: записывает данные в регистр на удаленной станции.

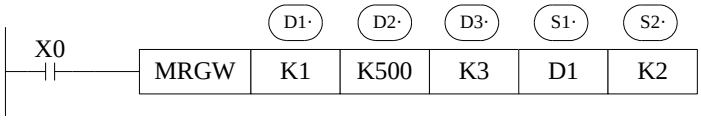

Номер удаленной станции. Аргументы: K、TD、CD、D、FD

Номер удаленного регистра. Аргументы: K、TD、CD、D、FD

Локальный регистр, с которого берутся данные. Аргументы: D

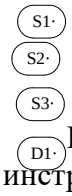
Номер порта. Границы: K1~K2

**8、Запись блока регистров [MRGW]**




- Функция, записывающая блок регистров в модбасе - 10H

Функция: записывает указанные регистры в регистры на удаленную станцию.




Номер удаленной станции. Аргументы: K、TD、CD、D、FD

Номер стартового регистра на удаленной станции. Аргументы: K、TD、CD、D、FD

Расшифровка инструкции: когда X0 включена, выполнится REGW или MRGW инструкция. После выполнения инструкций, установится конечный бит. При выключенном X0 инструкция не выполнится. Если при соединении произойдет ошибка, запрос автоматически повторится. При 10 ошибках связи установится флаг ошибки. Пользователь должен проверить качество соединения.