МПСвЭПиТК

Программирование в виде релейно - контактных схем

Управление состоянием битов

Есть 7 базовых команд, используемых для управления состоянием отдельного бита. Это OUTPUT, OUTPUT NOT, SET, RESET, DIFFERENTIATE UP, DIFFERENTIATE DOWN и KEEP. Все эти команды ставятся самыми последними в командной строке и требуют битового адреса в качестве операнда. Хотя подробности приведены в гл. 5, данные команды (за исключением OUTPUT и OUTPUT NOT, которые уже описаны) описаны здесь из-за их важности в большинстве программ. Хотя данные команды служат для переключения в 0 или 1 выходные биты области IR (т.е. выдать или убрать выходной сигнал с внешнего устройства), их также можно использовать для управлением состоянием других битов в области IR или в других областях.

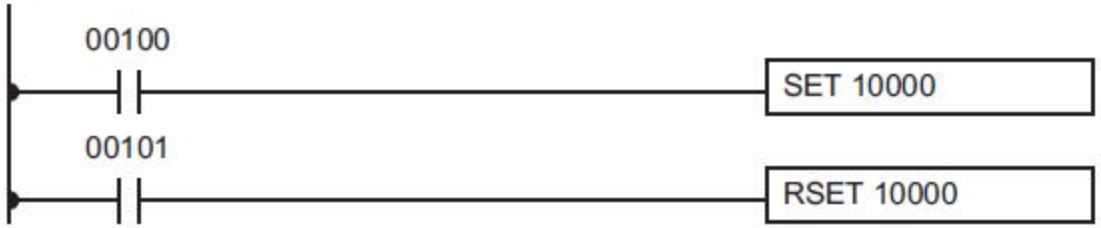
Установить в 1 (SET) и Сбросить в 0 (RESET)

Команды SET и RESET очень похожи на команды OUTPUT и OUTPUT NOT за исключением того, что они изменяют состояние битовых операндов только при условиях исполнения = 1. Данные команды не влияют на состояние операндов, если их условия исполнения = 0.

SET включит операнд в 1 при условии исполнения = 1, но в отличие от команды OUTPUT SET не выключит его в состояние 0 при условии исполнения = 0

RESET выключит битовый операнд в 0 при условии исполнения = 1, но в отличие от команды OUTPUT RESET не включит его в состояние 1, когда условие исполнения = 0.

В следующем примере IR 10000 будет = 1 при IR 00100 = 1 и останется = 1, пока IR 00101 не станет =1, несмотря на состояние IR 00100. Когда IR 00101 станет = 1, RESET сбросит IR 10000 в 0.

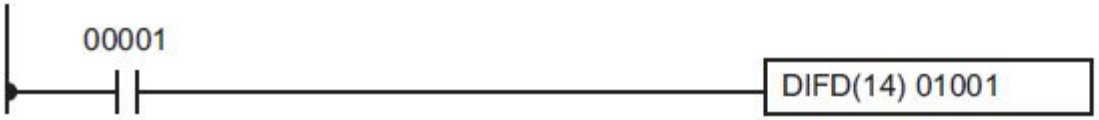


Адрес	Инструкция	Операнд
00000	LD	00100
00001	SET	10000
00002	LD	00101
00003	RSET	10000

Включить на 1 цикл (DIFFERENTIATE UP и DIFFERENTIATE DOWN)

Команды DIFFERENTIATE UP (включить на цикл при условии 0/1) и DIFFERENTIATE DOWN (включить на цикл при условии 1/0) служат для включения битового операнда в 1 на 1 цикл. Команда DIFFERENTIATE UP включает операнд в состояние 1 после того, как условие исполнения изменился с 0 на 1; Команда DIFFERENTIATE DOWN включает битовый операнд в состояние 1 после того, как условие исполнения изменится с 1 на 0; Обе этих команды требуют только одной строки в мнемокоде.

Адрес	Инструкция	Операнд
00000	LD	00000
00001	DIFU(13)	01000



Адрес	Инструкция	Операнд
00000	LD	00001
00001	DIFU(14)	01001

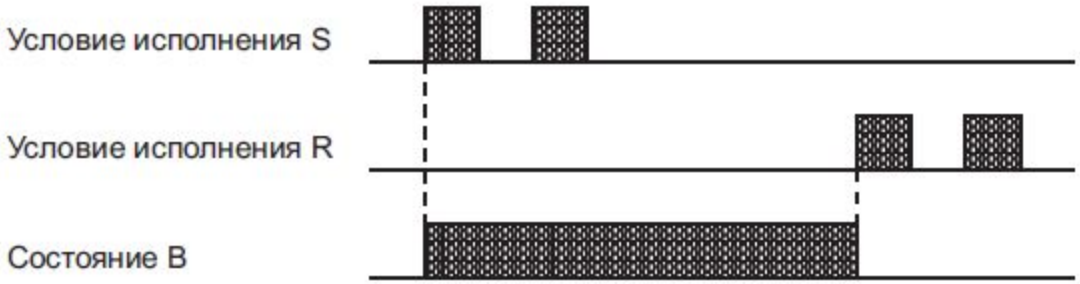
Здесь, IR 01000 будет включен в 1 на 1 цикл после включения IR 00000 в состояние 1. На следующем цикле после выполнения DIFU (13) IR 01000 будет выключено в 0 независимо от состояния IR 00000. Командой DIFFERENTIATE DOWN IR 01001 будет включен в 1 на 1 цикл после включения IR 00001 в состояние 0 (до того времени IR 01001 будет = 0), и будет выключен в 0 после выполнения DIFD (14).

Сохранить (KEEP)

Команда KEEP служит для сохранения состояния битового операнда и требует двух входов условий исполнения. К команде KEEP проводятся две командные линии. Когда условие исполнение на первой линии = 1, битовый операнд становится = 1. . Когда условие исполнения на второй линии = 1, битовый операнд становится = 0. Битовый операнд команды KEEP будет сохранять состояние 0 или 1 даже если он находится на секции схемы с командой INTERLOCK.

В следующем примере HR 0000 будет включен в 1 когда IR 00002 = 1 и IR 00003 = 0. HR 0000 останется в состоянии 1 до тех пор, когда либо IR 00004 либо IR 00005 включатся в 1. Как у всех команд, требующих более одной командной строки, командные строки кодируются перед командой, которой они управляют.

Адрес	Инструкция	Операнд
00000	LD	00002
00001	AND NOT	00003
00002	LD	00004
00003	OR	00005
00004	KEEP(11)	HR 0000

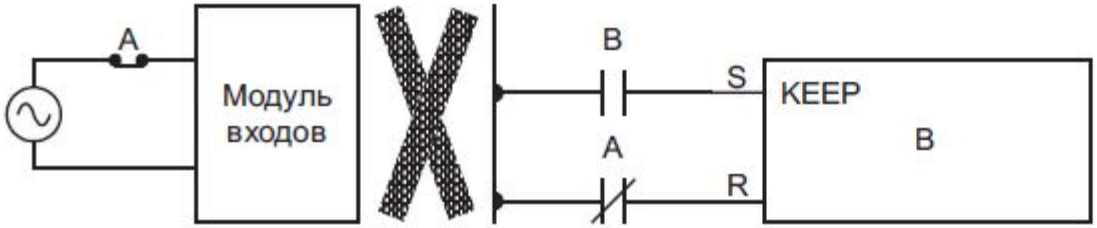


Флаги

Флагов, на которые оказывает воздействие данная команда, нет.

Предосторожности

Будьте осторожны, когда линией сброса KEEP(11) управляется нормально закрытое внешнее устройство. Не используйте входной бит с инверсным условием для линии сброса (R) KEEP(11), когда входное устройство работает на переменном токе. Задержка в снятии напряжения постоянного тока на ПК (по сравнению с переменным током устройства) может вызывать сброс указанного бита KEEP(11). Ситуация показана ниже.



4.5 Рабочие биты (внутренние реле)

Часто трудно запрограммировать условия так, чтобы сразу получить условия исполнения. Эти трудности легко преодолеваются тем, что используются определенные биты для переключения других команд. Такое программирование достигается с помощью рабочих битов. Иногда для этих целей требуются целые слова. Такие слова называются рабочими словами.

Рабочие биты не передаются с или на ПК. Они выбраны программистом для облегчения программирования. Биты входов/выходов и другие биты специального назначения нельзя использовать в качестве рабочих битов. Все биты в области IR, не выделенные в качестве битов входов/выходов, и некоторые неиспользуемые биты зоны AR можно использовать в качестве рабочих битов. Это облегчает разработку и написание программы, а также отладку.

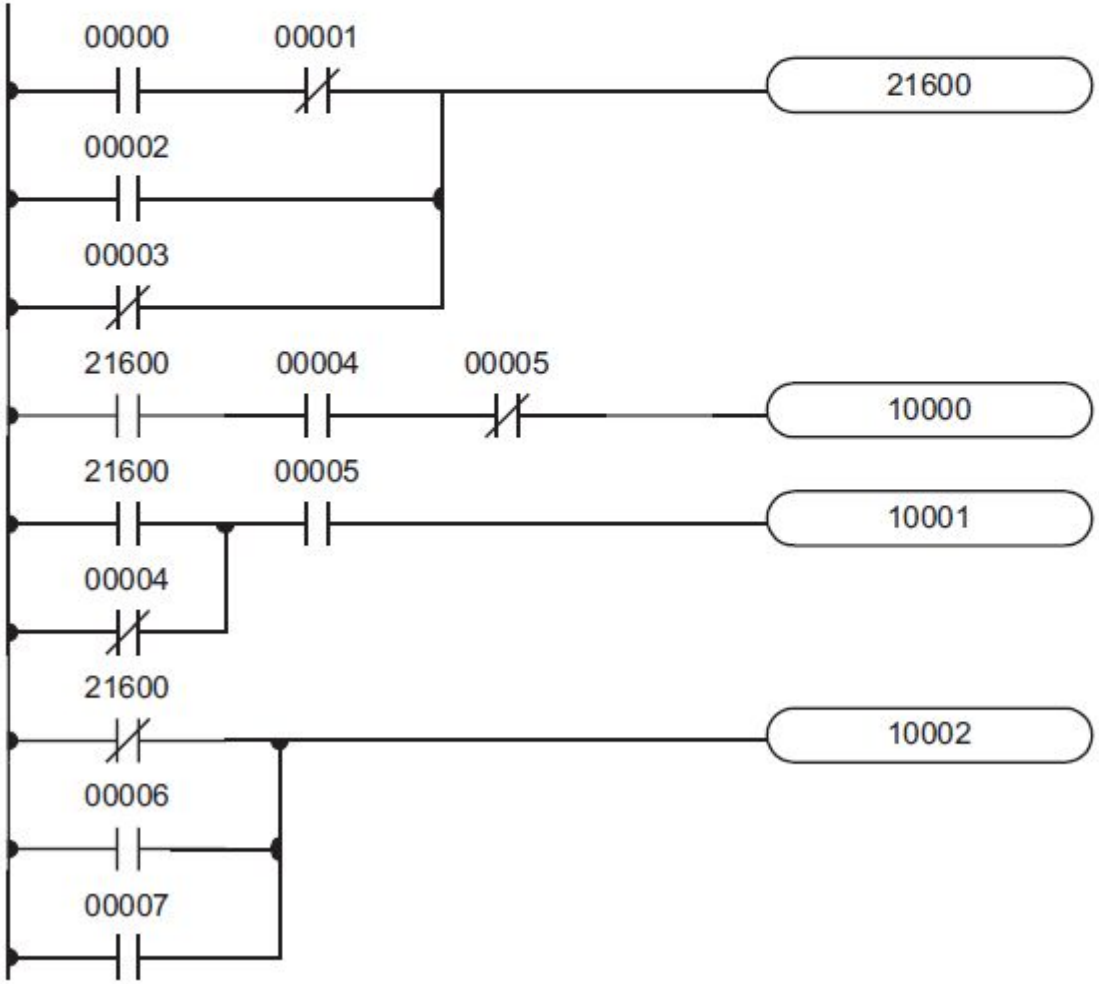
Применение рабочих бит

Примеры в данном пункте показывают два основных способа применения рабочих бит. Они могут служить примером для почти неограниченного числа способов применения рабочих бит. Если возникают проблемы при программировании, нужно рассмотреть возможность применения рабочих бит для упрощения программирования.

Рабочие биты часто используются с командами OUTPUT, OUTPUT NOT, DIFFERENTIATE UP, DIFFERENTIATE DOWN и KEEP. Сначала рабочие биты используются в качестве операндов для этих команд, чтобы позднее использовать их в качестве условий для выполнении команд. Рабочие биты можно также использовать и с другими командами, напр. SHIFT REGISTER (регистр сдвига) (SFT(10)).

Сокращение сложных условий

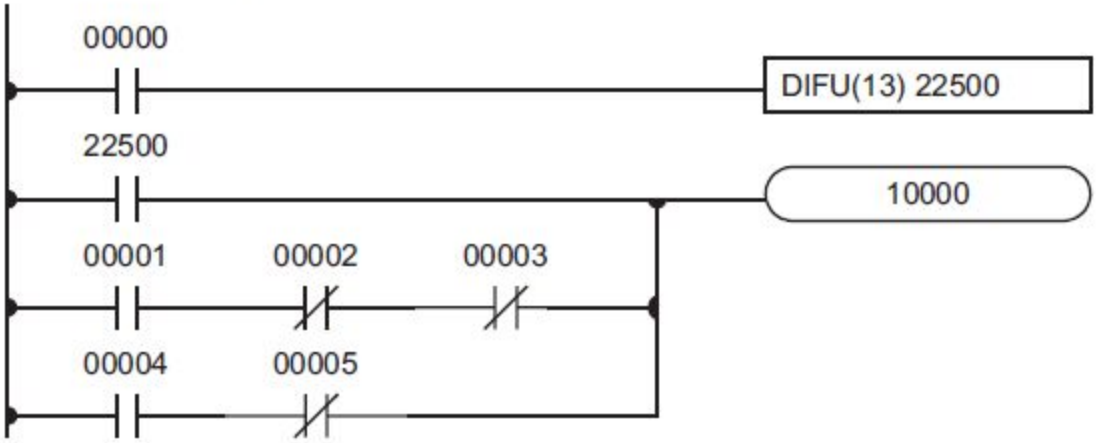
Рабочие биты можно использовать для упрощения программы, когда некоторая комбинация условий часто используется с другими условиями. В следующим примере IR 00000, IR 00001, IR 00002 и IR 00003 объединяются в логический блок, который сохраняет результирующее условие исполнение в IR 216000. Далее IR 21600 объединяется в логические блоки с различными другими условиями для задания условий срабатывания IR 10000, IR 10001 и IR 10002, т.е. включение выходов, приписанных к этим битам, в 1 или 0.



Условия при работе с фронтами

Рабочие биты можно также использовать для работы с фронтами для некоторых, но не для всех условий, требуемых для выполнения команд. В данном примере IR 10000 должен оставаться 1, пока IR 00001 = 1, а IR 00002 и IR 00003 = 0, или пока IR 00004 = 1 и IR 00005 = 0. Он должен будет = 1 только на 1 цикл каждый раз, когда IR 00000 включается в 1 (если только одно из предыдущих условий не находится постоянно в состоянии 1).

Это легко запрограммировать применением IR 22500 в качестве рабочего бита как операнда для команды DIFFERENTIATE UP (DIFU(13)). Когда IR 00000 включается в 1, IR 22500 включится в 1 на время 1 цикла и в следующем цикле сбросится в 0 командой DIFU(13). Если другие условия, управляющие IR 10000, не 1, рабочий бит IR 22500 включит на время одного цикла IR 10000 в 1.

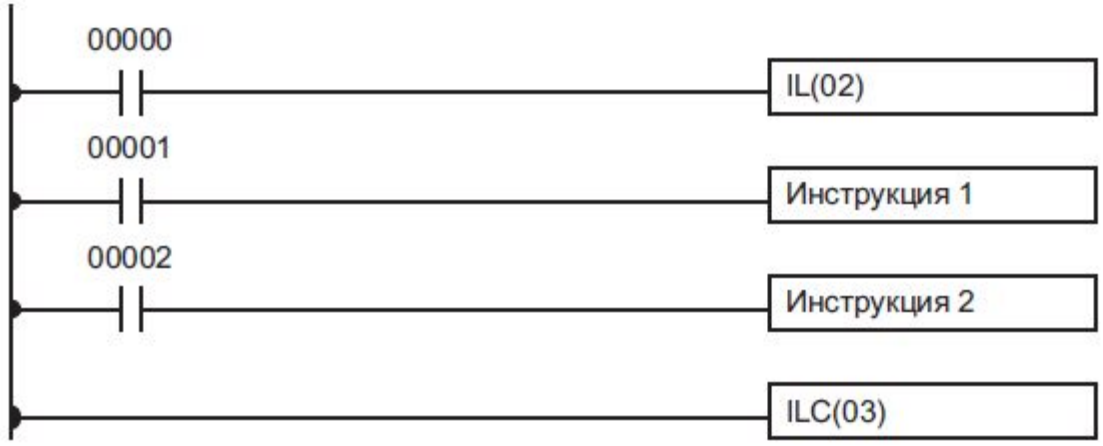


“Сгруппировать” - INTERLOCK

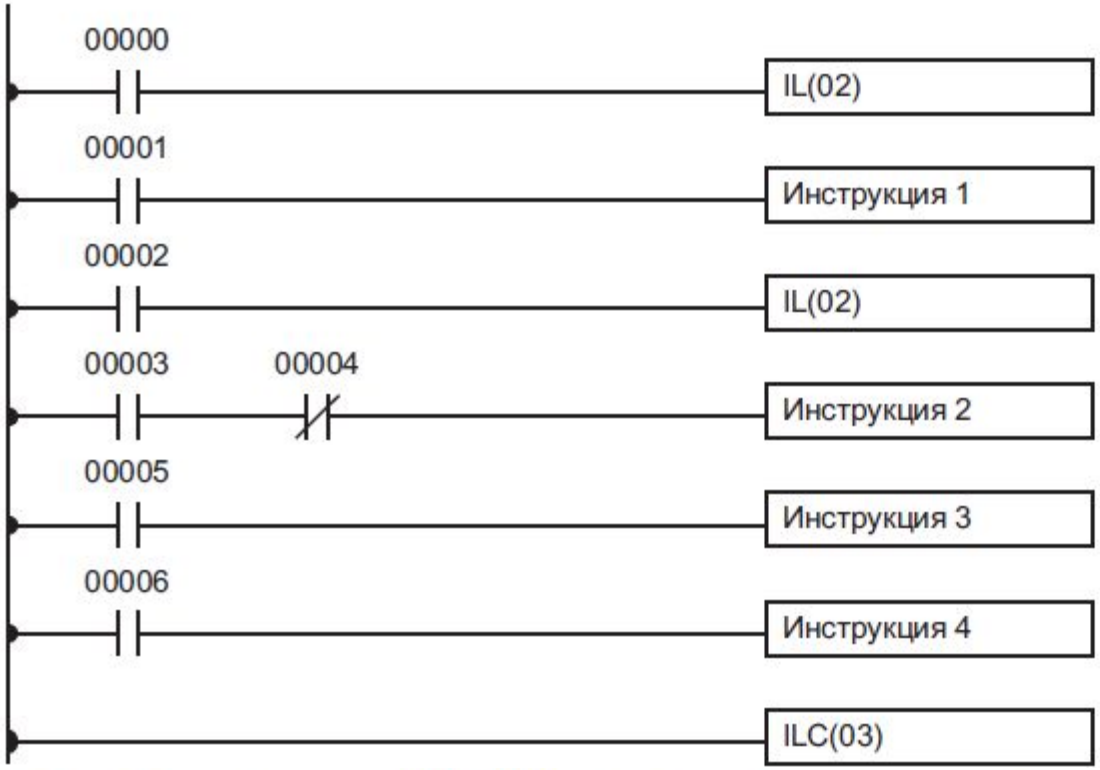
Проблемы сохранения условий исполнения в точках ветвления можно решить также командами INTERLOCK (сгруппировать) IL(02) и INTERLOCK CLEAR (разгруппировать) IL(03) для полного устранения точек ветвления, но позволив указанным условиям исполнения управлять группами команд. Команды INTERLOCK и INTERLOCK CLEAR всегда используются совместно.

Когда команда INTERLOCK помещена перед секцией релейно-контактной схемы, условие исполнения для команды INTERLOCK управляет исполнением всех команд до команды INTERLOCK CLEAR. Если условие исполнения для команды INTERLOCK = 0, все выходные (“правосторонние”) команды до INTERLOCK CLEAR будут выполняться с условием 0 (т.е. сброс всей секции РКС). Влияние, которое они оказывают на конкретные команды, смотрите гл. 5 -11 INTERLOCK и INTERLOCK CLEAR.

Схему В можно откорректировать также с помощью INTERLOCK. Здесь условия исполнения до точки ветвления ставятся в командную строку для команды INTERLOCK , все строки от точки ветвления записываются как отдельные командные строки и добавляется еще одна команда INTERLOCK CLEAR. Обратите внимание, что ни INTERLOCK, ни INTERLOCK CLEAR не требуют операнда.



Как показано на следующей схеме, внутри блока можно применить более одной команды INTERLOCK. Каждая действует до первой INTERLOCK CLEAR.



Если в данной схеме IR 00000 = 0 (т.е. условие исполнение для первой команды INTERLOCK = 0), команды 1 -4 будут выполнены с условием исполнения 0 и далее произойдет переход к команде, следующей за INTERLOCK CLEAR . Если в данной схеме IR 00000 = 1, состояние IR 00001 будет загружено как условие исполнения команды 1, затем состояние IR 00001 будет загружено как условие исполнения второй команды INTERLOCK. Если IR 00002 = 0, команды 2 - 4 будут исполнены с условием 0. Если IR 00002 = 1, IR 00003, IR 00005, IR 00006 определят первое условие исполнения на командных строках.

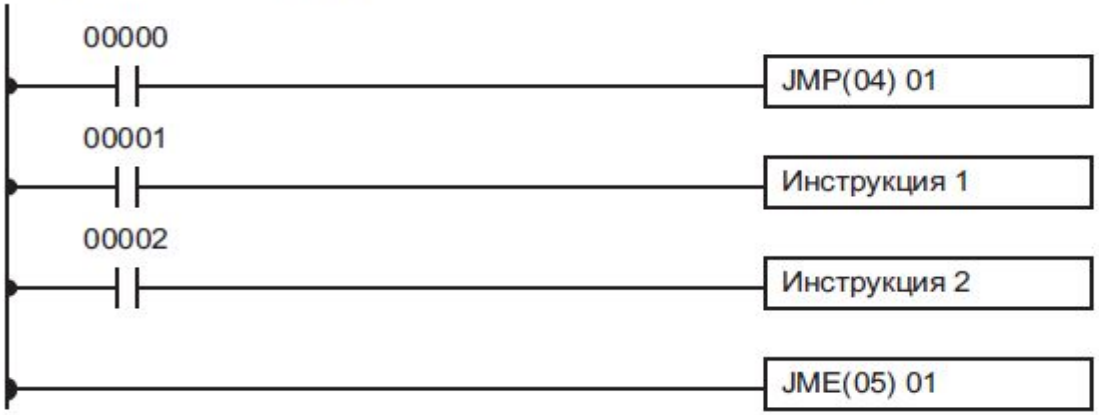
Переходы

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

Переходы создаются командами JMP(04) и JME(05) (Конец перехода). Если условие исполнения для команды JUMP = 1, программа выполняется, как будто данной команды не существует. Если условие исполнения для команды JUMP = 0, программа переходит сразу к команде JUMP END, не меняя состояний выходов между JUMP и JUMP END.

Всем командам JUMP и JUMP END присвоены номера 00 - 99. Есть 2 типа переходов. Номер перехода определяет тип перехода.

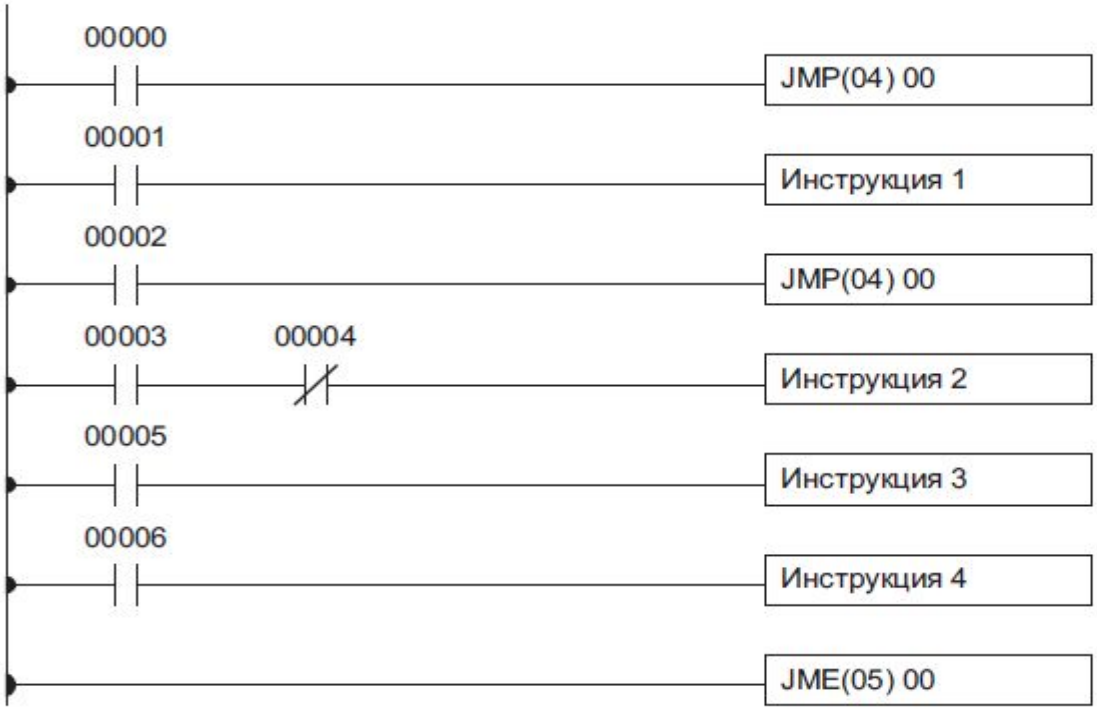
Переход можно определить номерами 01 - 99 только один раз, т.е. каждый номер может использоваться один раз с командой JUMP и один раз JUMP END. Когда выполнена команда JUMP, сразу происходит переход к JUMP END с тем же номером, как будто команд JUMP с другими номерами не существуют. Схема B из примера с TR и INTERLOCK может быть переписана с использованием команд JUMP, как показано ниже. Хотя в качестве номера перехода используется номер 01, можно использовать любой номер 00 - 99, если они не используются в других местах программы. JUMP и JUMP END не требуют операндов, а JUMP END не имеет и условий исполнения в командной строке.



Другой тип перехода создает JUMP с номером 00. С помощью JUMP с номером 00 можно создать сколь угодно много команд, и JUMP с номером 00 можно использовать без JUMP END между ними. Возможно даже для всех команд JUMP 00 перенести исполнение программы к одному JUMP END 00, т.е. для всех команд JUMP 00 требуется только одна команда JUMP END 00. Когда в качестве номера перехода задан 00, Исполнение программы переходит к ближайшей команде JUMP END 00. Хотя, как и при всех переходах, между JUMP 00 и JUMP END 00 состояние не изменяется и команды не выполняются, программа должна искать ближайшую JUMP END 00, чем немного продлевается время исполнения.

Исполнение программы с несколькими командами JUMP 00 и одной JUMP END 00 аналогично исполнению программы с командой INTERLOCK. Следующая схема аналогична той, которая исполнялась с командой INTERLOCK , но переписана с командами JUMP.

Выполнение программы будет отличаться, (например, в предыдущей сбрасывались некоторые секции программы , а переходы не влияют на состояние битов между JUMP и JUMP END).



Команды таймеров и счетчиков

TIM и TIMH(15) являются командами декрементирующего таймера, включающегося в 1 с задержкой, которым требуются номера TC и заданное значение (SV). STIM(69) используется для управления интервальными таймерами, которые используются для вызова подпрограмм прерываний.

CNT - команда декрементирущего счетчика, а команда CNTR(12) - команда реверсивного счетчика. Обе команды требуют номер TC и заданного значения (SV). Обе подключены к нескольким командным линиям, которые служат в качестве входного сигнала (сигналов) и сброса. CTBL(63), INT(89) и PRV(62) используются для управления высокоскоростным счетчиком. INT(89) служит также для остановки выдачи импульсов.

Номер TC нельзя задать дважды, т.е. если он уже задан в качестве определителя в какой-либо команде таймера или счетчика, его нельзя использовать снова. Если номер TC задан, его можно использовать столько раз, сколько требуется, в качестве операнда в командах, кроме задания таймера и счетчиках.

В CQM1 номера TC имеют диапазон 000..511, в CPM1 диапазон 000..127. Префикс не требуется при использовании номера TC в качестве определителя в команде задания таймера или счетчика. Когда номер TC задан как таймер, его можно использовать с префиксом TIM для использования в качестве операнда в определенных командах. Префикс TIM применяется независимо от команды таймера, которой задавался таймер. Когда номер TC задан как счетчик, его можно использовать с префиксом CNT для использования в качестве операнда для определенных команд. Префикс CNT применяется также независимо от команды счетчика, которой задавался счетчик.

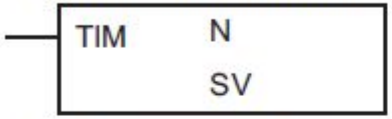
Номера ТС можно задать как операнды, требующие либо битовых, либо словных данных. Когда номер ТС задан в качестве операнда, требующего битовых данных, данный номер ТС получает доступ к биту, выполняющему функцию “флаг завершения”, который указывает, что заданное время отработано/счет завершен, т.е. бит, нормально 0, включится в 1, когда заданное значение (SV) истечет. Когда номер ТС задан в качестве операнда, требующего данных в виде слова, данный номер ТС получает доступ к ячейке памяти, которая содержит текущее значение таймера или счетчика. Таким образом, текущее значение таймера или счетчика можно использовать в качестве операнда для команды CMP(20) или других команд, для которых разрешена область ТС. Это делается назначением номера ТС, служащего для задания этого таймера или счетчика, для доступа к области памяти, в которой хранится текущее значение (PV).

Обратите внимание, что “TIM 000” используется и для задания команды TIM с номером 000, и для задания флага завершения для этого таймера, и для задания текущего значения этого таймера. Значение этого термина в контексте должно быть ясным, т.е. первое всегда - команда, второе всегда - битовый операнд и третье всегда словный операнд. То же самое верно для всех остальных номеров ТС с префиксами TIM или CNT.

Задание (SV) может вводится константой или адресом слова в области данных. Если слово области IR, присвоенное блоку входов, задано в качестве адреса слова, блок входа можно подключить таким образом, чтобы задание (SV) можно было вводить извне с цифровых переключателей или других аналогичных устройств. Таймеры и счетчики, подключенные подобным образом, могут получить задание только из внешнего источника в режимах RUN или MONITOR. Все заданные значения, включая заданные извне, должны быть в двоично-десятичном виде.

5.15.1 TIM - таймер

Обозначение на схеме



Область операндов

N	номер таймера	#
SV	заданное значение (слово, BCD)	IR, SR, AR, HR, LR, DM, #

Ограничения

Заданные значения лежат в диапазоне 000.0..999.9. Десятичная точка не вводится.

Каждый номер TC можно использовать в качестве определителя только для одной команды таймера или счетчика. В CQM1 номера TC могут быть в диапазоне 000..511, в CPM1 в диапазоне 000..127.

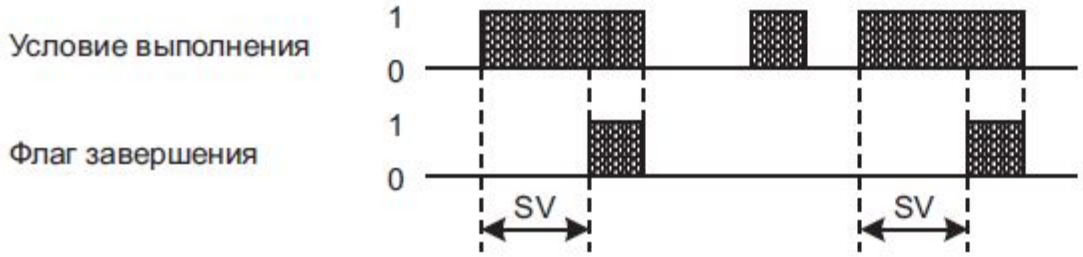
TC 000..TC 015 (TC 000..TC 003 в CPM1) рекомендуется использовать в команде задания TIM только если они требуются для команды TIMH(15).

Описание

Таймер запускается, когда условие срабатывания устанавливается в 1 и сбрасывается (в заданное значение), когда условие срабатывания = 0. После запуска TIM отсчитывает время, вычитая по дискрете (в 0.1 с) от задания.

Если условие исполнения остается в 1 достаточно долго для отсчета текущего значения до нуля, флаг завершения данного номера устанавливается в 1 и остается в 1 до сброса таймера (т.е. когда условие станет = 0).

Следующий рисунок иллюстрирует соотношение между условием исполнения TIM и Флага завершения, связанного с ним.



Предупреждения

Таймеры в секциях INTERLOCK сбрасываются, когда условие исполнения IL(02) = 0. Таймеры также сбрасываются при сбросе питания. Если нужен таймер, который не сбрасывался бы при таких условиях, биты импульсов времени в области SR можно подсчитать для задания таймеров, использующих CNT. Подробности см. 5-15-2.

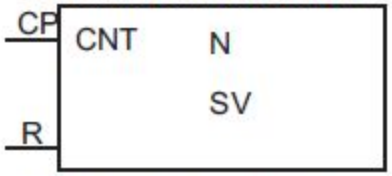
Флаги

ER: Заданные значения не в двоично-десятичном виде.

Косвенно адресуемое слово DM не существует. (Содержимое слова *DM не в двоично-десятичном виде, или превышена граница области DM).

CNT - Счетчик

Обозначение на схеме



Область операндов

N	номер счетчика	#
SV	заданное значение (слово, BCD)	IR, SR, AR, HR, LR, DM, #

Ограничения

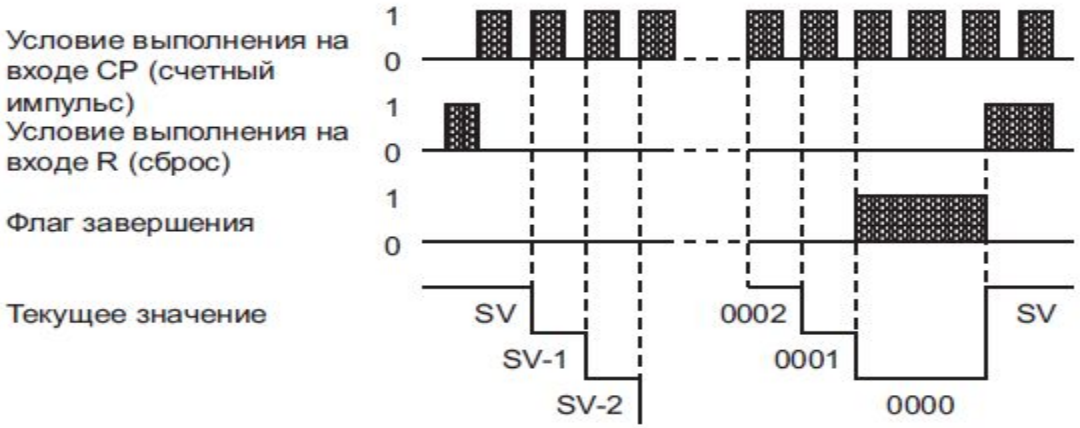
Каждый номер ТС можно использовать в качестве определителя только для одной команды таймера или счетчика. В CQM1 номера ТС задаются в диапазоне 000..511, в CPM1/CPM1A/SRM1 в диапазоне 000..127.

Описание

CNT служит для отсчета вниз от заданного значения, когда условие исполнения на счетном входе (CP), изменяется из 0 в 1, т.е. текущее значение будет декрементировано (уменьшено на 1) при текущем условии исполнения счетного выхода = 1 и условии исполнения в прошлом цикле = 0. Если условие исполнения не изменяется или изменяется из 1 в 0, текущее значение счетчика не изменяется. Флаг завершения счетчика устанавливается в 1, когда текущее значение достигает 0 и остается 1 до сброса счетчика.

Счетчик сбрасывается входом сброса (R). Когда R изменяется из 0 на 1, текущее значение сбрасывается до задания. Пока R = 1, текущее значение не декрементируется. Отсчет вниз от задания начнется снова, когда R станет = 0. Текущее значение не будет сброшен в секциях INTERLOCK или при прерывании питания.

На следующем рисунке показаны изменения условий исполнения, Флаг завершения и текущее значение счетчика. Разная высота линии текущего значения служит только для того, чтобы показать изменение текущего значения.



Предосторожности

Выполнение программы продолжится, если даже задание введено не в двоично-десятичном виде, но тогда задание будет некорректным.

Флаги

ER: Задание не в двоично-десятичном виде.

Косвенно адресуемое слова DM не существует. (Содержимое слова *DM не в двоично-десятичном виде, или превышена граница области DM)

Пример

В следующем примере CNT служит для создания расширенных таймеров путем счета битов импульсов часов в области SR.

CNT 001 считает, сколько раз бит часов частотой 1 сек (SR 25502) переключается из 0 в 1. Здесь IR 00000 используется для управления числом импульсов при работе CNT.

Поскольку в данном примере задание для CNT 001 = 700, флаг завершения для CNT 002 включается в 1, когда истекут 1 x 700 раз или 11 минут и 40 сек. Это приведет к включению IR 01602 в 1.