Руководство пользователя контроллера DL205

Том 2 из 2

D2-USER-M

Оглавление

Глава 6. Программирование с помощью барабанного командоаппарата

(только для DL250-1/DL260)

В этой главе…

Введение

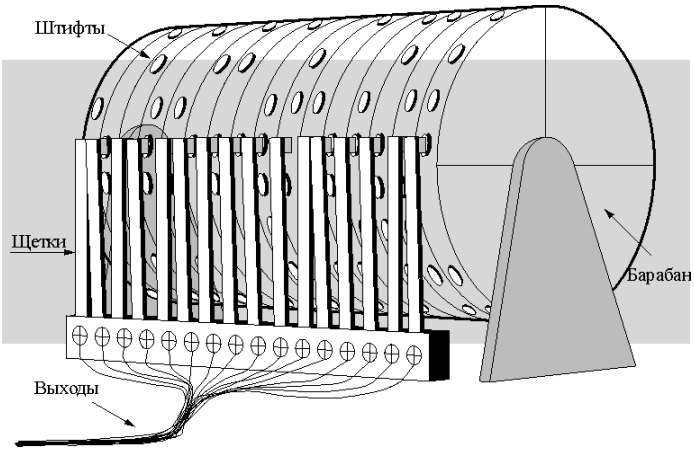
Назначение

230	240	250–1	260
×	×	✓	✓

Четыре команды процессоров DL250-1 и DL260 служат для программной имитации электромеханического барабанного командоаппарата. Команды предполагают некоторое расширение принципов работы электромеханического барабанного командоаппарата, о чем написано ниже.

Терминология барабанного командоаппарата

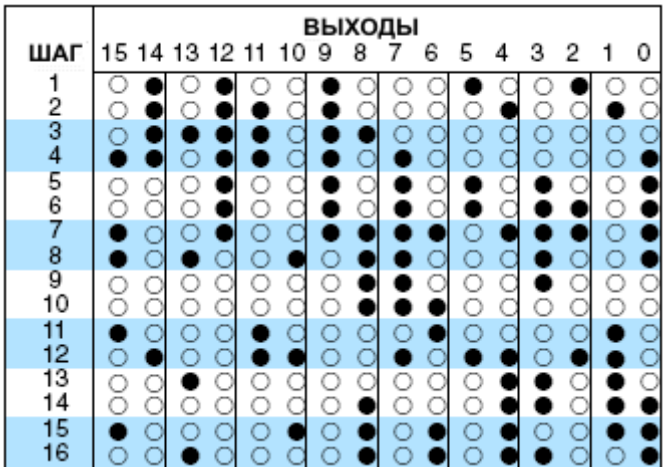
Команды барабанного командоаппарата наилучшим образом подходят для циклических процессов, состоящих из конечного числа шагов. Они могут простым образом выполнить работу программы релейной логики, состоящей из многих цепей. Барабанные командоаппараты позволяют сэкономить время, необходимое для разработки и отладки программы.
Глядя на реальный механический барабан (см. рисунок ниже), мы введем ряд терминов, связанных с командами барабанного (DRUM) командоаппарата.
На изогнутой поверхности механического барабана обычно располагаются штифты. Штифты размещаются в определенной **комбинации (pattern)**, формируя набор действий, необходимых для управления установкой. В заданные моменты времени двигатель или соленоид поворачивают барабан на строго определенный угол. При вращении неподвижные щетки касаются штифтов (есть = ВКЛ, нет = ВЫКЛ). При этом между барабаном и щетками устанавливается или разрывается электрический контакт. Таким образом, формируются электрические выходы барабанного командоаппарата. Эти выходы используются в качестве сигналов включения / выключения исполнительных механизмов.
За один оборот барабан проходит конечное число положений, называемых **шагами(steps)**. Каждый шаг представляет собой некоторое состояние процесса. При включении питания барабан **возвращается(reset)** к определенному шагу. Барабан переходит от одного шага к другому по времени или по событию. При определенных условиях оператор установки может вручную перевести барабан на один шаг вперед, используя механизм ручного управления барабаном (**jog**). Срабатывание контактов каждой щетки создает уникальную серию сигналов включения/ выключения, называемую **последовательностью(sequence)**. Эта последовательность сигналов служит для управления определенным исполнительным механизмом установки. Поскольку барабан круглый, с каждым оборотом последовательность автоматически повторяется. Прикладные задачи могут существенно отличаться друг от друга, поэтому один барабан может делать один оборот за секунду, а другой – за неделю.



Барабанные командоаппараты не только реализуют возможности механических барабанов, но и расширяют их. Например, они обладают возможностью мгновенного перевода к **предварительно заданному** шагу, что невозможно для механических барабанов. Функция перевода позволяет перейти от текущего шага к любому другому по команде!

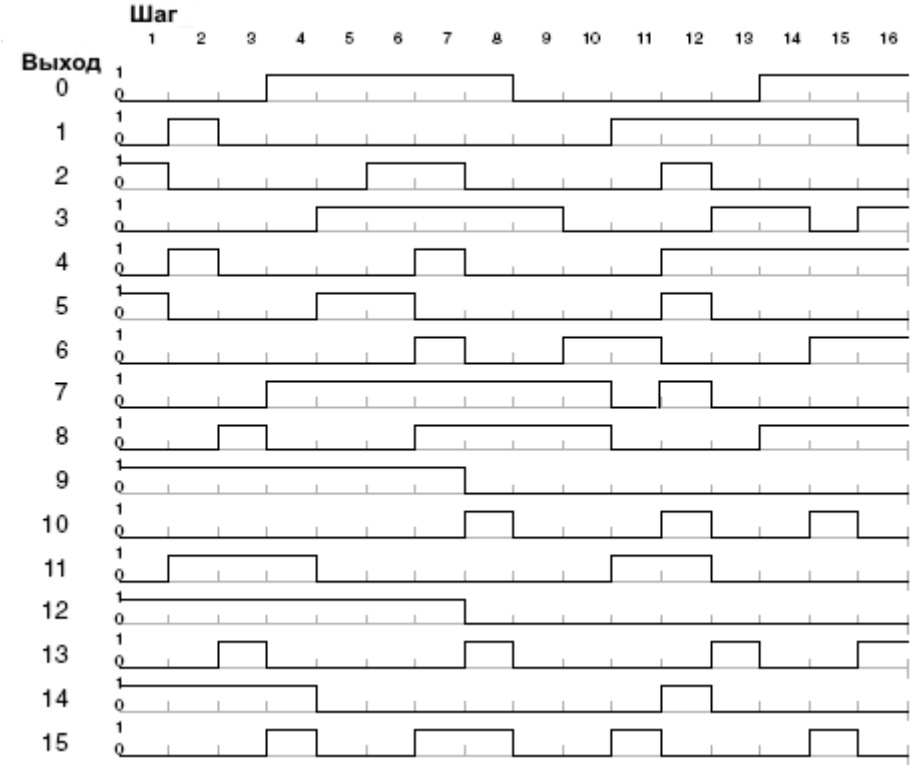
Представление барабанного командоаппарата в виде диаграммы

В этом руководстве, как и при редактировании в *Direct*SOFT, электронный барабанный командоаппарат представляется в виде диаграммы. Представьте, что полый цилиндр барабана разрезали между двумя рядами штифтов, после чего развернули. Теперь барабан будет выглядеть так, как показано на рисунке ниже. Каждый строка с номером от 0 до 15 представляет собой шаг (соответствует 16-битному слову). Каждый столбец представляет выход с номером от 1 до 16. Сплошные кружки соответствуют штифтам (состояние ВКЛ) механического барабана, а пустые кружки – отсутствию штифтов (состояние ВЫКЛ).



Выходные последовательности

Механический барабанный командоаппарат реализует последовательности изменений управляющих состояний на своих электрических выходах. На следующем рисунке изображена последовательность сигналов ВКЛ/ВЫКЛ, создаваемая барабаном с представленной выше конфигурацией. Сравните их, и вы увидите, что они эквивалентны! Если вы заметили это сходство, вам будет несложно понять принцип работы команд барабанного командоаппарата.



Переходы между шагами

Типы команд барабанного командоаппарата

Для процессорных модулей DL250-1 и DL260 существует четыре типа команд барабанного командоаппарата:

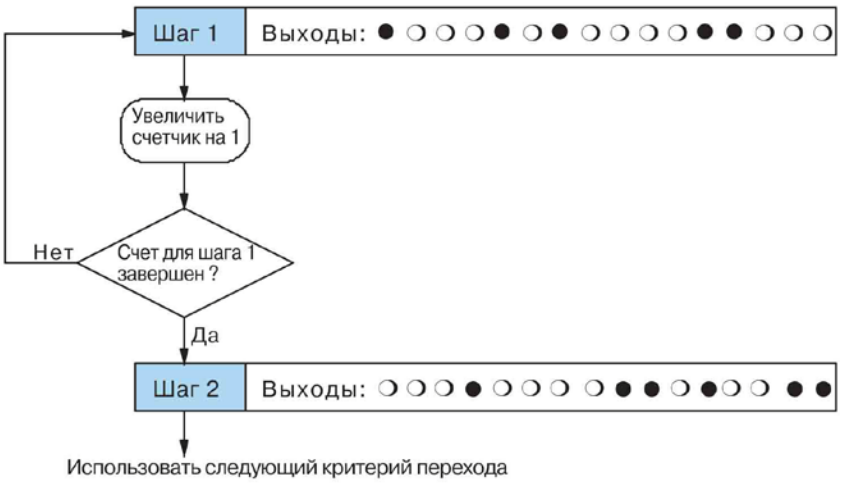
- Барабан с дискретными выходами и переходом между шагами по времени (DRUM)
- Барабан с дискретными выходами и переходом между шагами по времени и по событию (EDRUM)
- Маскированный барабанный командоаппарат с дискретными выходами и переходом по событию (MDRMD)
- Маскированный барабанный командоаппарат с выходом по словам и переходом по событию (MDRMW)

Все четыре типа команд барабанного командоаппарата поддерживают шаговые переходы по времени, и три из них поддерживают переходы по событию. Дополнительно можно определить выход в виде слова или отдельных битов и задать маску выходов (включение/выключение отдельных выходов).

У каждого барабанного командоаппарата 16 шагов и у каждого шага 16 выходов (см. рисунок ниже). Каждый из выходов может задавать значение X, Y, или реле C, обеспечивая гибкость программирования. Как показано, мы присваиваем шагу 1 произвольную уникальную конфигурацию выходов (○= ВЫКП, ●= ВКЛ). При программировании команд барабанного командоаппарата вы также определяете назначение выходов и состояния ВКЛ/ВЫКЛ на этот момент. Все шаги используют одно и то же назначение выходов, но у каждого шага может быть своя собственная уникальная конфигурация.

Переходы шагов только по времени

Барабанные командоаппараты переходят от шага к шагу в зависимости от времени и/или внешнего события (входа). Для каждого шага при вводе команды барабанного командоаппарата назначается свое собственное условие перехода. Рисунок выше иллюстрирует переход от шага к шагу только по времени.



Перейдя к Шагу 1, барабан остается в нем в течение некоторого времени (указанного в программе). Временной масштаб таймера задается программно от 0.01 до 99.99 секунд. Таким образом, выбирается разрешение таймера, или, другими словами, длительность одного **такта**. Для всех шагов используется общий масштаб, но каждый шаг характеризуется различным количеством тактов, которое задается программно. Когда количество тактов, заданное для Шага 1, отсчитано, барабан переходит к Шагу 2. Вслед за этим сразу же меняются состояния выходов, поскольку теперь они определяются комбинацией "штифтов" Шага 2.

Время, в течение которого командоаппарат остается на определенном шаге, рассчитывается по формуле:

Длительность шага = 0.01с * Временной масштаб * количество тактов шага.

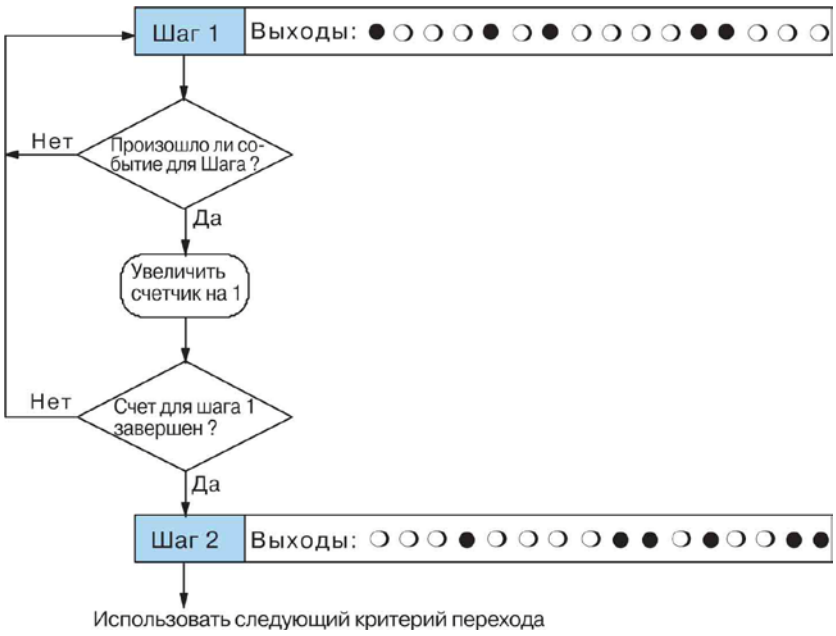
Например, если вы задаете в программе временной масштаб (длительность такта) 5 секунд и назначаете 12 тактов для Шага 1, длительность Шага 1 после перехода к нему барабанного командоаппарата составит 60 секунд. Максимальная длительность каждого шага определяется по следующей формуле:

Макс. длительность шага = 0.01 с x 9999 x 9999 =999 800 с = 277.7 часа = 11.6 дней.

ПРИМЕЧАНИЕ: При первоначальном выборе разрешения временного масштаба длительность одного такта на практике удобно указывать равной 1/10 длительности самого короткого шага барабанного командоаппарата. Это позволяет оптимизировать длительность шага 10% приращениями. Другие шаги с меньшей длительностью можно оптимизировать даже с меньшими (в процентном отношении) приращениями. Кроме того, обратите внимание на то, что команда барабанного командоаппарата выполняется только один раз за цикл ЦПУ. Следовательно, бессмысленно задавать временной масштаб барабана намного меньше времени сканирования процессора.

Переходы по времени и по событию

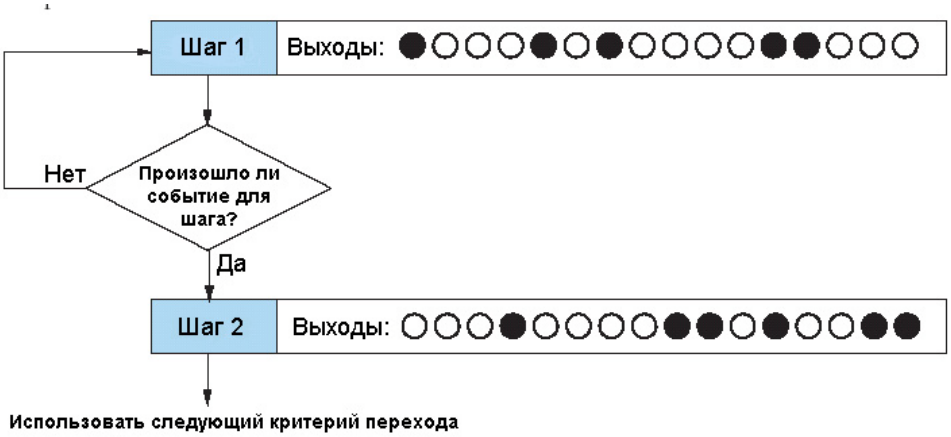
Переходы между шагами могут также осуществляться как по времени, так и/или по событию. На рисунке ниже показано, как происходят переходы в этом случае.



При переходе к Шагу 1 барабанный командоаппарат устанавливает комбинацию состояний выходов, показанную на рисунке. Затем он приступает к опросу внешнего входа, указанного в программе для данного Шага. В качестве входов событий можно выбрать дискретные точки типа X, Y или C. Предположим, что в качестве входа события для Шага 1 было выбрано X0. Пока X0 остается в состояние ВЫКЛ, барабан остается на Стадии 1. Когда X0 находится в состоянии ВКЛ, удовлетворяется критерий события и начинается приращение таймера. Отсчет времени происходит до тех пор, пока удовлетворяется критерий события (X0=ВКЛ). Когда количество тактов для Шага 1 достигнуто, барабанный командоаппарат переходит к Шагу 2. Сразу же вслед за этим состояния выходов изменяются в соответствии с комбинацией, определяемой Шагом 2.

Переходы только по событию

Для перехода к следующему шагу необязательно указывать одновременно и время, и событие в качестве условий перехода. Для каждого шага барабанного командоаппарата в отдельности можно указать либо время, либо событие, либо оба этих критерия одновременно. Например, можно сделать так, чтобы переход от Шага 1 происходил по событию, от Шага 2 – только по времени, а от Шага 3 – и по времени, и по событию. Более того, вы можете выбрать для использования только часть из 16-ти шагов и только часть из 16-ти выходов.



Назначение счетчиков

Каждая инструкция барабанного командоаппарата использует ресурсы четырех счетчиков процессора. При использовании команды барабанного командоаппарата в программе указывается номер первого счетчика. Три следующих счетчика используются командоаппаратом автоматически. Бит первого счетчика устанавливается по завершении цикла барабанного командоаппарата и обнуляется по сбросу барабанного командоаппарата. Текущее значение счетчика и состояние его бита точно отражают ход выполнения команды барабанного командоаппарата, и их можно контролировать в программе релейной логики.

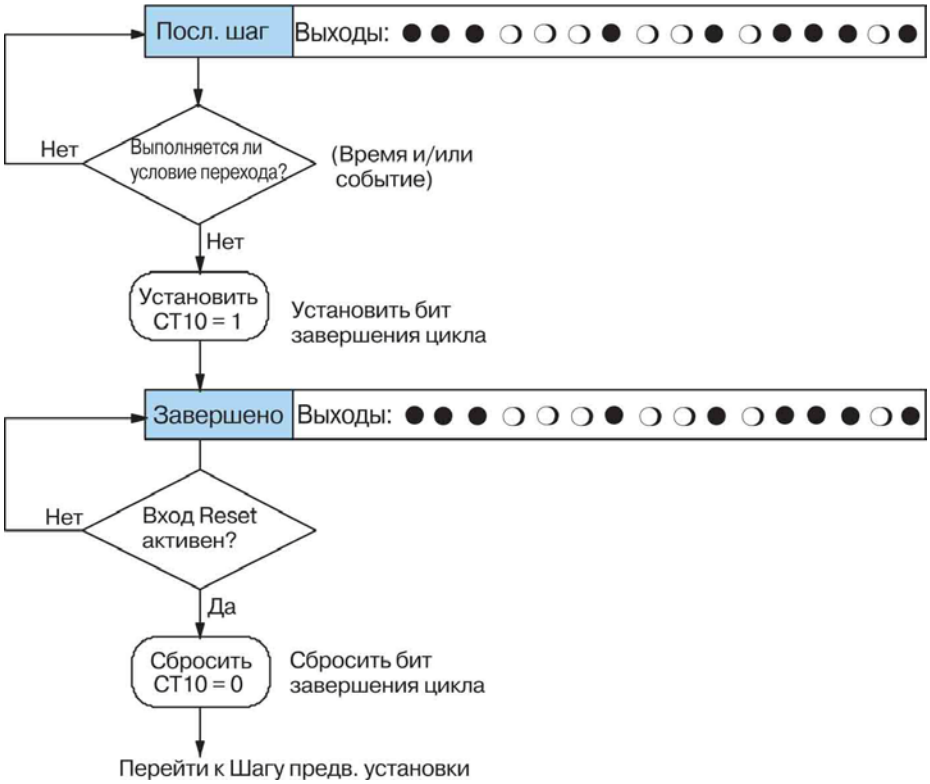
Предположим, что мы программируем счетчик на 8 шагов и выбрали для этих целей счетчик CT10 (следует помнить о восьмеричной нумерации счетчиков). Использование счетчиков показано в таблице справа. Правый столбец содержит типичные значения, пояснения к которым приводятся ниже.

Назначение счетчиков			
CT10	Число тактов шага (Текущее значение)	V1010	1528
CT11	Счетчик таймера	V1011	0200
CT12	Заданный шаг (Preset)	V1012	0001
CT13	Текущий шаг (Current)	V1013	0004

CT10 указывает, что с начала выполнения текущего Шага прошло 1528 тактов, а CT13 указывает, что текущим является Шаг 4. Если мы запрограммировали Шаг 4 на 3000 тактов, то сейчас Шаг завершен лишь наполовину. CT11 выполняет роль таймера, отсчитывая время с шагом 0,01 секунды. Изменение его самого младшего разряда соответствует интервалу 0.01 секунды. Значение 200 означает, что текущий Шаг длится уже 2 секунды (0.01 x 200 с соответствует 1528 тактам). Наконец, CT12 содержит номер Шага предварительной установки, указанный в команде барабанного командоаппарата. В нашем случае при активизации входа сброса барабанный командоаппарат переходит на Шаг 1. Значение CT12 может быть изменено только из релейной программы, либо после правки команды барабанного командоаппарата и перезапуска программы. Бит счетчика CT10 устанавливается по завершении цикла барабанного командоаппарата и обнуляется по сбросу командоаппарата.

Завершение последнего шага

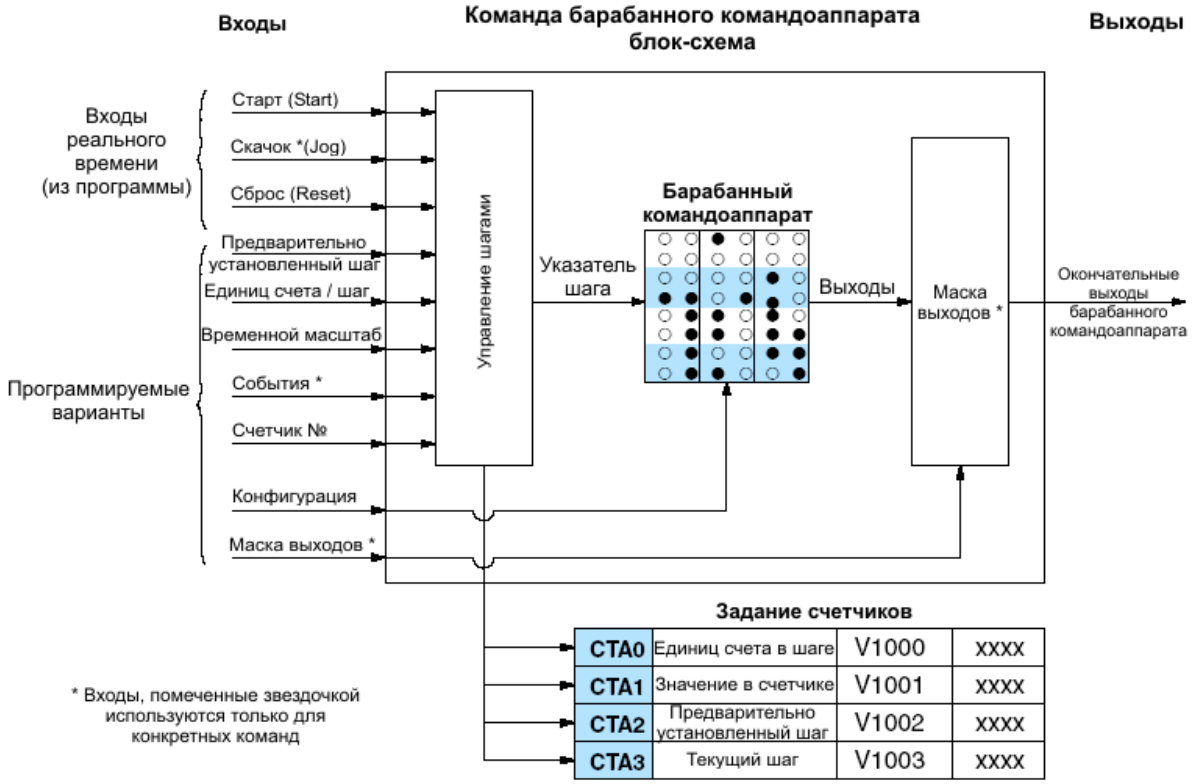
В качестве последнего шага в последовательности барабанного командоаппарата может быть шаг с любым номером, так как возможна реализация неполных барабанных командоаппаратов (см. рисунок ниже). Когда выполнены условия перехода для последнего шага, командоаппарат устанавливает бит счетчика, заданного в команде барабанного командоаппарата (например, CT10). Затем командоаппарат переходит в конечное состояние "цикл завершен". При этом сохраняется комбинация состояний выходов, установленная на последнем шаге. Когда цикл барабанного командоаппарата завершается, входы Start (Старт) и Jog (Скачок) перестают действовать. Барабанный командоаппарат выходит из состояния "цикл завершен", когда активизируется вход Reset (Сброс) (либо при переходе из режима программирования к режиму выполнения). Бит завершения цикла (например, CT10) при этом сбрасывается, и командоаппарат переходит непосредственно к предустановленному шагу.



Краткое описание барабанного командоаппарата

Структурная схема команд барабанного командоаппарата

Помимо параметров настройки самого барабана, команды барабанного командоаппарата используют различные дополнительные входы и выходы. Обратимся к рисунку ниже.



В команде барабанного командоаппарата используется ряд входов для управления шагами (в этом и состоит основное управление барабаном). Ниже перечислены входы и их функции:

Start (Старт) – вход Start действует, только если не активен вход Reset (Сброс). Когда Start включен, работает таймер барабанного командоаппарата (если для шага выбран переход по времени) и барабан контролирует вход события (если для шага выбран переход по событию). Когда Start выключен, барабанный командоаппарат остается в своем текущем состоянии (Reset должен оставаться выключенным), при этом сохраняется текущая комбинация выходов барабанного командоаппарата (состояний ВКЛ/ВЫКЛ).

Jog (Скачок) – вход Jog действует, только если отключен Reset (Start может быть включен или выключен). При каждом переходе входа Jog из состояния ВЫКЛ в состояние ВКЛ барабанный командоаппарат переводится на один шаг вперед (вход Jog поддерживается только командой EDRUM).

Reset (Сброс) - приоритет входа Reset выше, чем у входа Start. Когда Reset включен, барабанный командоаппарат возвращается к шагу предварительной установки. Когда Reset выключен, вход Start работает как обычно.

Preset Step (Шаг предварительной установки) – любой выбранный вами шаг от 1 до 16 (как правило, Шаг 1). Барабанный командоаппарат переходит к этому шагу, когда включается вход Reset,

либо при переходе ЦПУ в режим выполнения.

- **Counts /Step (Тактов на шаг)** – количество тактов, в течение которого барабан находится на каждом шаге. Количество тактов задается для каждого шага отдельно. Этот параметр программировать не обязательно.
- **Timer Value (Значение таймера)** – текущее значение таймера, отсчитывающего количество тактов для шага.
- **Counter # (Номер счетчика)** — номер счетчика задает первый из четырех последовательных счетчиков, используемых барабанным командоаппаратом для управления шагами. Он позволяют следить, как командоаппарат проходит свой цикл управления.
- **Events (События)** — дискретная точка типа X, Y, C, S, C, CT или SP служит в качестве входа для перехода между шагами. Для барабанных командоаппаратов с переходом по времени/событию программировать события необязательно.

ПРЕДУПРЕЖДЕНИЕ: Выходы барабана задействованы все время, пока ЦПУ находится в режиме выполнения. Вход Start не должен быть включен. Выходы не сбрасываются входом Reset. При переходе к режиму выполнения выходы барабана автоматически устанавливаются или сбрасываются в соответствии с конфигурацией текущего шага барабанного командоаппарата. При этом учитывается влияние используемой маски выходов.

Состояние регистров барабанного командоаппарата перед включением питания

Для приложения важен выбор шага, который будет начальным при включении питания и при переходе из режима программирования к режиму выполнения. Обратимся к следующей таблице. Если память счетчика настроена, как память с запоминанием (см. настройку - retentive range), барабанный командоаппарат сохранит свое прежнее состояние.

Номер счетчика	Функция	Инициализация при включении питания	
		Без сохранения значения	С сохранением значения
CTA(n)	Текущее количество тактов шага	Инициализация = 0	Используется предыдущее состояние (без изменений)
CTA(n + 1)	Значение счетчика времени	Инициализация = 0	Используется предыдущее состояние (без изменений)
CTA(n + 2)	Шаг предварительной установки	Инициализация = № шага предварительной установки	Используется предыдущее состояние (без изменений)
CTA(n + 3)	№ текущего шага	Инициализация = № шага предварительной установки	Используется предыдущее состояние (без изменений)

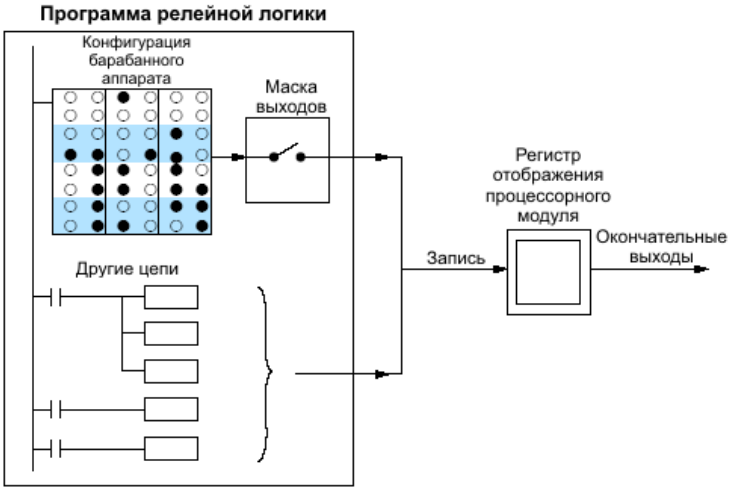
В приложениях с относительно короткой длительностью цикла барабанного командоаппарата требуется, как правило, сброс при включении питания, поэтому используется вариант без запоминания. В задачах с относительно большой длительностью цикла барабанного командоаппарата может потребоваться возобновление состояния, предшествующее прекращению работы, поэтому используется запоминание. По умолчанию всегда выбрано запоминание. Это означает, что оперативная V-память будет сохранять свое состояние при инициализации.

Работа маски выходов

Иногда при управлении выходами нужна большая гибкость, чем может обеспечить стандартная конфигурация выходов барабанного командоаппарата. Маска выходов позволяет отключить управление заданной конфигурации командоаппарата для выбранных выходов на определенных шагах, позволяя управлять этими выходами с помощью другой программы. Функция «маски выходов» поддерживается двумя из четырех типов команд барабанного командоаппарата:

- **MDRMD** — маскированные барабаны с дискретными выходами
- **MDRMW** — маскированные барабаны с выходами по словам, меняющимися по событию

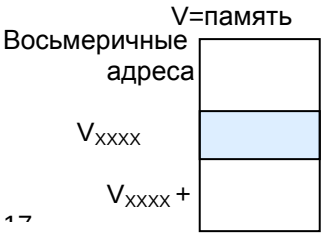
Маска выходов — это просто управление барабанным командоаппаратом с помощью побитового включения/выключения, записываемого в регистр отображения шестнадцати выходов (см. приведенный рисунок). Регистр отображения содержит формальный текущий статус всех точек ввода/вывода. В конце каждого сканирования процессор ПЛК использует состояние регистра отображения для записи в действительные точки выхода.



Практическое использование масок выходов для барабанного командоаппарата включает:

- **Вложенная последовательность** — конкретный выход может образовывать специальную последовательность «внутри» конкретного шага, в то время как остальные выходы барабанного командоаппарата остаются неизменными. Вместо использования дополнительных шагов мы с помощью маски отключаем выход и управляем им в течение шага с помощью внешней программы.
- **Ручное форсирование** — иногда на конкретном шаге нужно вручную управлять некоторыми выходами. Задание маски для соответствующих выходов барабанного командоаппарата позволяет ручным входам получить преимущество над программным управлением.

Для каждого шага задается собственное слово маски! Каждый бит слова маскирует соответствующую точку выхода. Как показано справа, значения маски будут храниться в 16-регистровой таблице V-памяти. В команде барабанного командоаппарата задается начальное положение таблицы. Например, таблица, которая начинается в V2000, простирается до V2017. У различных барабанов MDRMD или MDRMW должны быть независимые таблицы масок

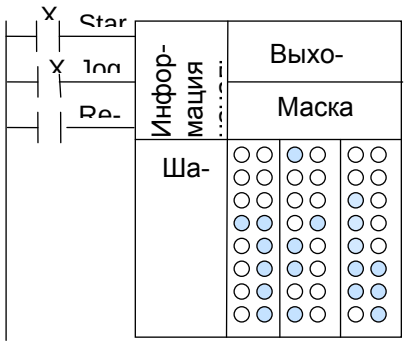


Если бит маски = 1, барабанный командоаппарат управляет точкой выхода. Если бит маски=0, командоаппарат не может записывать в регистр отображения, **поэтому выход остается в своем текущем состоянии.**

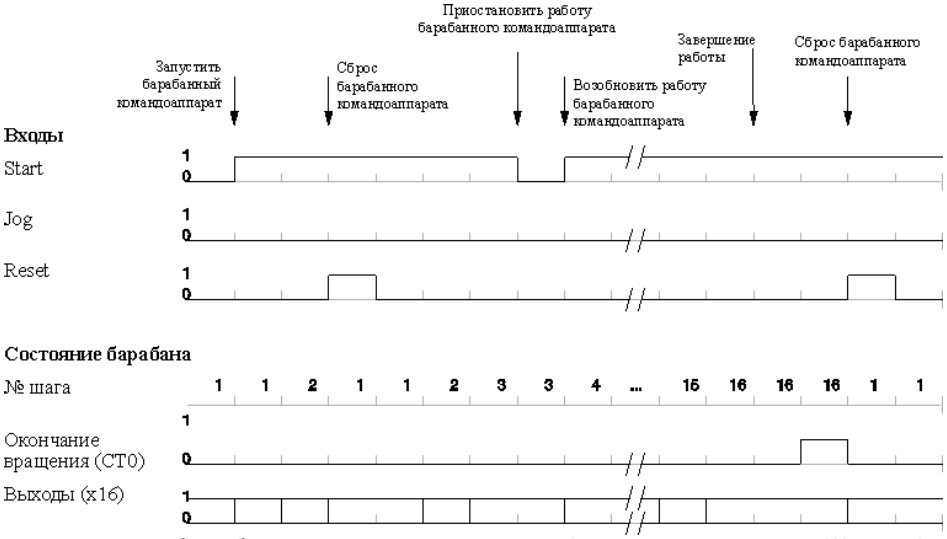
Управление барабанным командоаппаратом

Управляющие входы барабанного командоаппарата

Теперь мы готовы собрать воедино изложенные ранее принципы и продемонстрировать управление с помощью команд барабанного командоаппарата. На рисунке справа показана упрощенная общая команда барабанного командоаппарата. Выходы релейной программы управляют входами Start, Jog и Reset. Бит первого счетчика командоаппарата (CT0, например) сообщает об окончании цикла командоаппарата.



Ниже приводится временная диаграмма, на которой показана последовательность произвольного переключения входов барабанного командоаппарата с переходом шагов по времени и реакция командоаппарата. При переходе в режим выполнения ЦПУ записывает в Регистр номера шага номер Шага предварительной установки (как правило, Шаг 1). Когда включается вход Start, барабанный командоаппарат начинает свою работу, ожидая событие и /или запуская таймер (в зависимости от настройки). Когда барабанный командоаппарат переходит к Шагу 2, включается Reset при по-прежнему включенном входе Start. Поскольку Reset "старше" входа Start, барабанный командоаппарат возвращается к шагу предварительной установки (Шаг 1). Обратите внимание на то, что барабанный командоаппарат **удерживается** на шаге предварительной установки, пока включен Reset, и что этот шаг **не выполняется** (не реагирует на события и не ведет счет времени), пока Reset не будет выключен.
После того, как барабан переходит к Шагу 3, сразу же выключается вход Start, останавливая приращение таймера барабанного командоаппарата до тех пор, пока Start не включится снова.

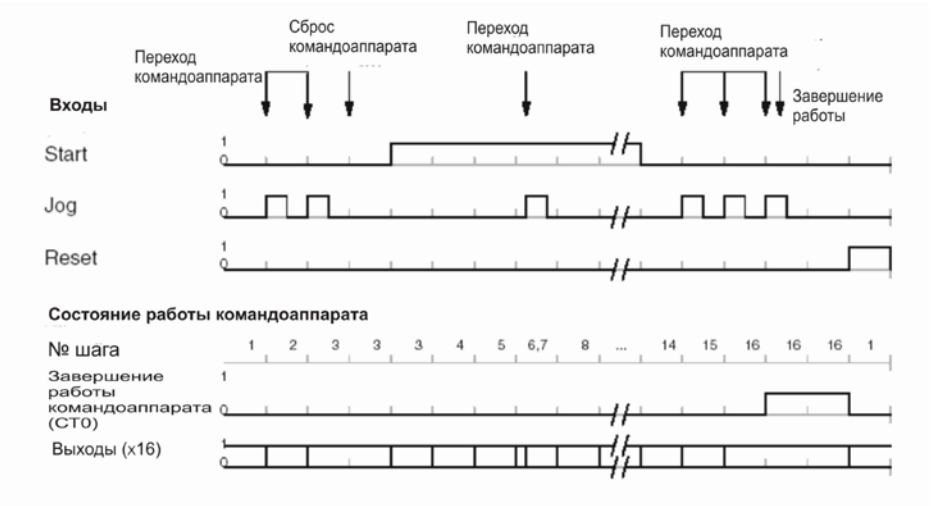


Когда завершается последний шаг барабанного командоаппарата (в нашем примере - Шаг 16), устанавливается бит завершения цикла (CT10), и номер шага остается равным 16. Когда включается вход Reset, бит завершения цикла барабанного командоаппарата (CT10) сбрасывается, и барабанный командоаппарат переводится в Шаг предварительной установки.

ПРИМЕЧАНИЕ: на временной диаграмме все шаги имеют одинаковую длительность. На практике длительность шагов может существенно различаться в зависимости от выбранного для них количества тактов.

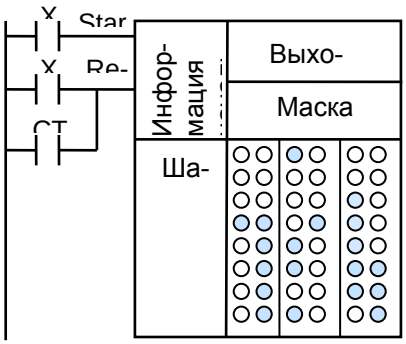
На приведенном ниже рисунке показано, как работает вход Jog у барабанных командоаппаратов с переходами по событиям. Вход Start может быть как включен, так и выключен (Reset при этом должен быть выключен). Два скачка переводят командоаппарат на Шаг 3. После этого включается вход Start, и барабанный командоаппарат начинает работать как обычно. В течение Шага 6 на входе Jog возникает другой сигнал. Он переводит командоаппарат на Шаг 7, сбрасывая таймер в 0. Командоаппарат начинает выполнять немедленно Шаг 7, поскольку Start уже включен. На Шаг 8 командоаппарат переходит обычным образом.

Когда барабанный командоаппарат переходит к Шагу 14, выключается вход Start. Два дополнительных сигнала Jog переводят командоаппарат на Шаг 16. Обратите внимание на то, что для перевода командоаппарата из Шага 16 к состоянию "завершение цикла", требуется третий сигнал Jog. Наконец, поступает сигнал Reset, который переводит барабанный командоаппарат к Шагу предварительной установки и сбрасывает бит завершения цикла.



Барабанные командоаппараты с самовозвратом

Для приложений часто требуются барабанные командоаппараты, которые автоматически начинают работать снова после завершения цикла. Эта возможность легко реализуется с помощью бита окончания цикла командоаппарата. На рисунке справа заданы начальные установки команды барабанного командоаппарата для CT0, поэтому мы выполняем логическое ИЛИ для бита завершения (CT0) и входа Reset. После выполнения последнего шага командоаппарат устанавливает CT0, что переводит командоаппарат на предварительно установленный шаг с одновременным сбрасыванием CT0. Контакт X1 по-прежнему используется для ручного сброса.



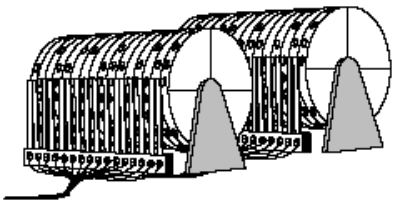
Инициализация выходов барабана

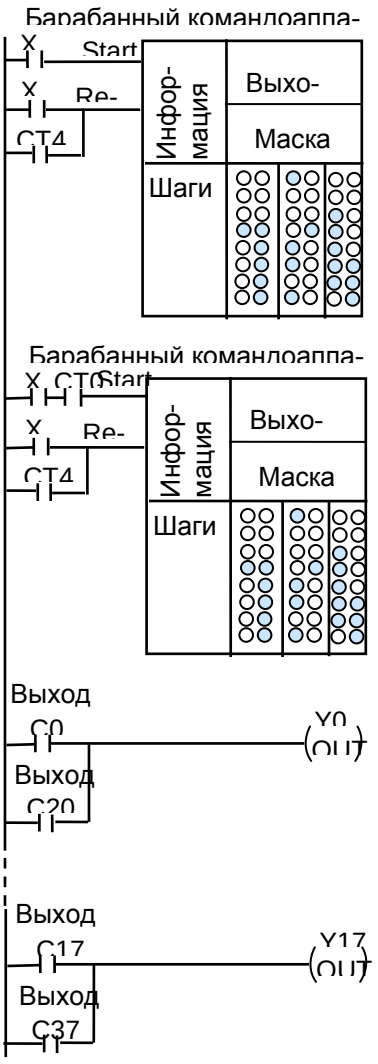
Если процессор находится в Рабочем режиме, выходы барабанного командоаппарата доступны всегда. При переходах в Рабочем режиме барабанный командоаппарат переходит к предварительно установленному шагу, и выходы включаются в соответствии с конфигурацией этого шага. Если ваше приложение требует отключения всех выходов при включении питания, возможны два подхода:

- Сделать предварительно установленный шаг барабанного командоаппарата «шагом сброса», с отключением всех выходов.
- Использовать барабанный командоаппарат с маской выходов. Маска при первом сканировании инициализируется значением «0000» с помощью контакта SP0, а также с помощью команд LD K000 и OUT Vxxx, где Vxxxx — это ячейка регистра маски.

Каскадные барабанные командоаппараты, обеспечивающие более 16 шагов

Иногда требуется барабанный командоаппарат, число шагов которого превышает 16. Решение состоит в том, чтобы использовать два или более командоаппарата, логически объединенных в каскад. Когда заканчивает работу первый командоаппарат, начинает работать второй, и т.д. Помните, что команда барабанного командоаппарата записывает выходы на каждом сканировании, даже когда вход Start отключен. Поэтому два командоаппарата, использующие одни и те же точки выхода, будут конфликтовать. Решить эту проблему можно, используя для выходов каждого командоаппарата различные контакты управляющих реле C, а затем выполнить над их выходами логическое ИЛИ, управляя окончательными выходами.





На рисунке справа два барабанных командоаппарата ведут себя как один 32-шаговый командоаппарат. Процедура заключается в следующем:

- Бит завершения работы первого барабанного командоаппарата (в данном примере CT0) используется в качестве входа Start следующего командоаппарата.
- Бит окончания вращения последнего барабанного командоаппарата используется в качестве входа Reset всех командоаппаратов (в данном примере CT4).
- Если для контакта необходим ручной сброс, выполняется операция ИЛИ с описанным контактом входа Reset (X1 в данном примере).
- Если для вашего применения барабанного командоаппарата требуется маска, для маски выходов обоих командоаппаратов используется один и тот же адрес V-памяти.
- Для каждого барабанного командоаппарата используются различные выходные обмотки управляющего реле C, над которыми программа выполняет, как показано, операцию ИЛИ.

Теперь Y0 является окончательным выходом объединенных барабанных командоаппаратов. Обратите внимание, что у каждого командоаппарата должен быть «холостой» шаг (обычно шаг 1), на котором его выходы C выключены во время работы другого командоаппарата (командоаппаратов).

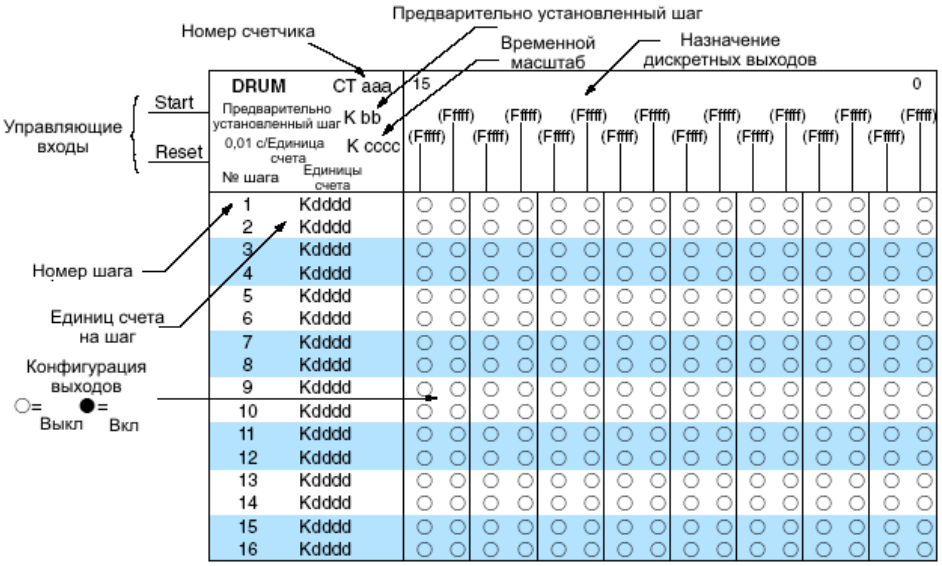
Команды барабанного командоаппарата

Все команды барабанного командоаппарата DL250-1 и DL260 могут быть запрограммированы с помощью *Direct*SOFT, и только для команды EDRUM можно воспользоваться ручным программатором (с версией программного обеспечения 1.8 или более поздней). В данном разделе описано использование *Direct*SOFT для всех команд и мнемоника ручного программатора для команды EDRUM.

Барабанный командоаппарат с дискретными выходами и переходом по времени (DRUM)

230	240	250–1	260
✗	✗	✓	✓

Барабанный командоаппарат с дискретными выходами и переходом по времени — это основная команда барабанного командоаппарата DL250-1 и DL260. Она работает в соответствии с правилами, описанными на предыдущих страницах. Ниже команда показана в виде схемы, выводимой в *Direct*SOFT.



Барабанный командоаппарат с переходом по времени обеспечивает 16 шагов и 16 выходов. Переход с шага на шаг происходит только по времени, которое определяется количеством тактов, заданных для каждого шага. Для неиспользуемых шагов в программе следует указывать нулевое "количество тактов" (значение по умолчанию). Каждой дискретной точке можно назначить тип X, Y или C, либо оставить ее неиспользуемой. Комбинацию выходных состояний для каждого шага можно редактировать в *Direct*SOFT графически.

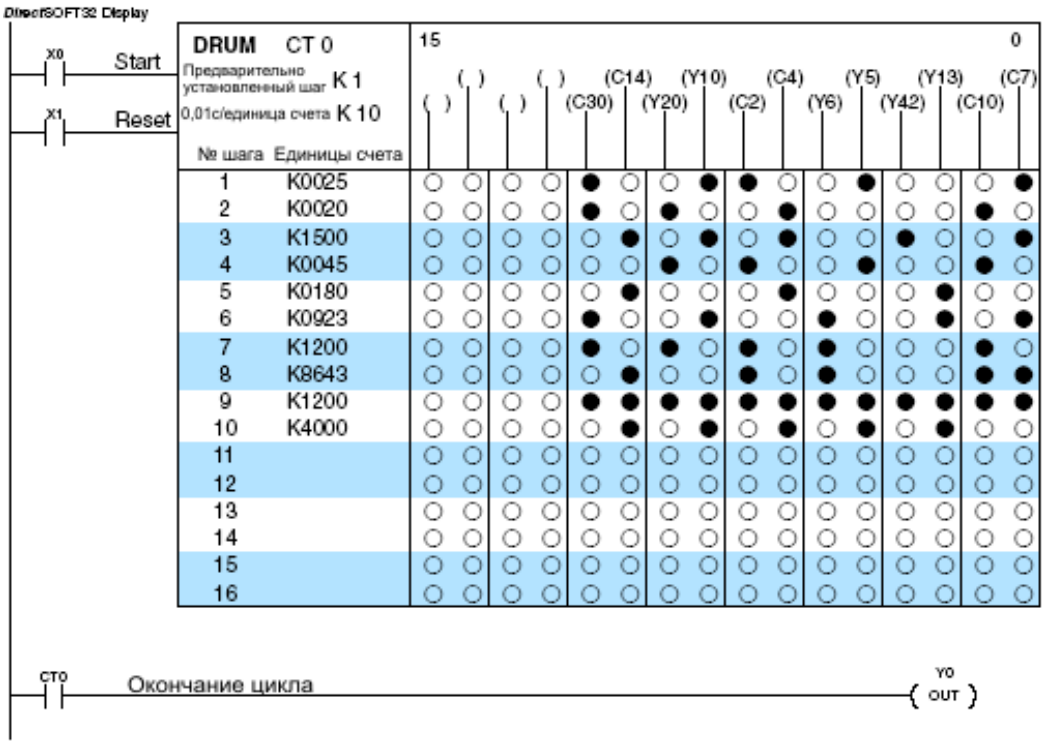
Когда вход Start включен, работает таймер барабанного командоаппарата. Таймер прекращает работу, когда завершается последний шаг или включается вход Reset. При переходе ЦПУ из режима программирования в режим выполнения, а также при включении входа Reset барабан возвращается к выбранному шагу предварительной установки.

Параметры барабанного командоаппарата	Поле	Типы данных	Диапазон
Номер счетчика	aaa	K	0 – 174 (DL250-1) 0 – 374 (DL260)
Предварительно установленный шаг	bb	K	1 – 16
Временной масштаб	cccc	K	0 – 99, 99 с
Единиц счета на шаг	dddd	K	0 - 9999
Дискретные выходы	Fffff	X, Y, C, GX, GY	См. стр 3-52 или стр.3-53

Команды барабанного командоаппарата используют четыре счетчика ЦПУ. Программа может читать значения счетчиков, определяя текущее состояние барабанного командоаппарата. Программа также может в любой момент записать в CTA(n+2) новый номер шага предварительной установки. Что касается других счетчиков, они служат только для целей контроля.

Номер счетчика	Диапазон для DL250-1	Диапазон для DL260	Функция	Функция бита счетчика
CTA(n)	0 -174	0 - 374	Текущее число тактов шага	CTn = Окончание цикла
CTA(n+1)	1 -175	1 - 375	Значение счетчика времени	CT(n+1) = (не используется)
CTA(n+2)	2 -176	2 - 376	Шаг предварительной установки	CT(n+2) = (не используется)
CTA(n+3)	3 -177	3 - 377	Текущий шаг	CT(n+3) = (не используется)

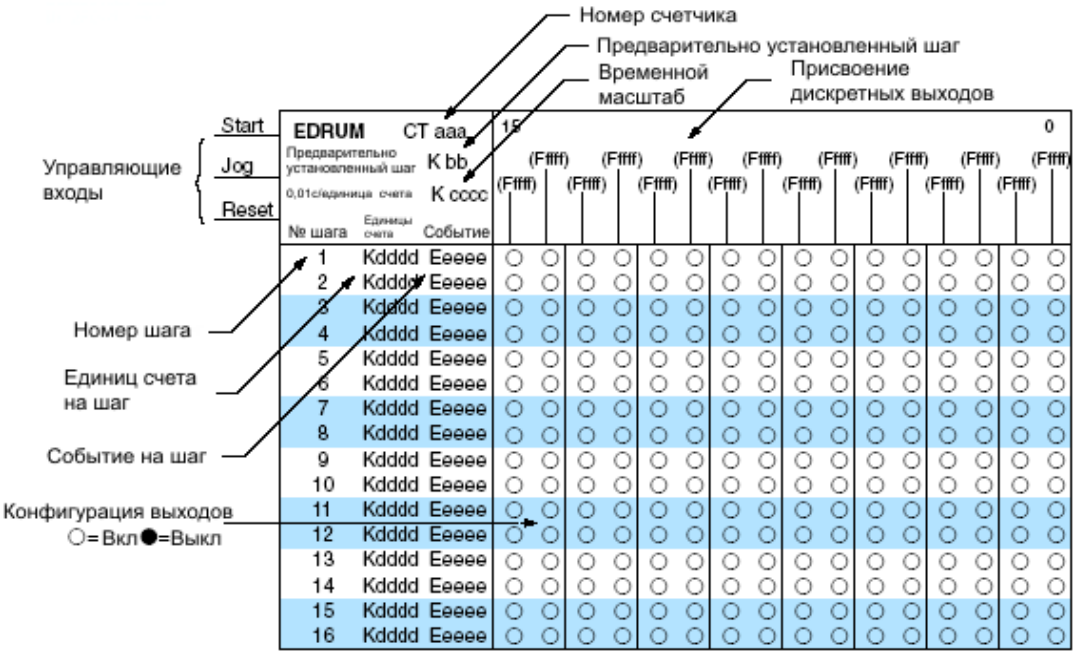
На следующем рисунке показано представление типичной релейной программы, использующей команду DRUM в *Direct*SOFT. Используются шаги 1…10, задействовано 12 выходных точек из 16-ти. Шагом предварительной установки является Шаг 1. Длительность одного такта (временной масштаб) = (K10 x 0.01) = 0.01 секунды. Следовательно, длительность Шага 1 составляет (25 x 0.1) = 2.5 секунды. В последней цепи программы бит завершения цикла (CT10) включает выход Y0 по завершении последнего шага (Шага 10). Барабанный командоаппарат сбрасывается, сбрасывая также CT10.



Барабанный командоаппарат с дискретными выходами и переходом по событию (EDRUM)

230	240	250–1	260
✗	✗	✓	✓

Барабанный командоаппарат с дискретными выходами и переходом по событию обладает всеми свойствами командоаппарата с переходом по времени. Он работает в соответствии с общими правилами работы командоаппаратов, описанными в начале этого раздела. Ниже эта команда показана в виде схемы, выведенной в *Direct*SOFT.



Барабанный командоаппарат с дискретными выходами и переходом по событию обеспечивает 16 шагов и 16 выходов. Переход шага происходит по времени и/или по событию. Положительный фронт на входе Jog (переход из ВЫКЛ во ВКЛ) также переводит барабан на один шаг вперед. Длительность шага задается в импульсах, а в качестве событий указываются дискретные контакты. Неиспользуемые шаги и события на бланке командоаппарата можно оставить пустыми (это их значение по умолчанию). Тип дискретных точек выхода может быть назначен независимо. Конфигурацию выходов можно графически редактировать в *Direct*SOFT.

Как только будет инициализирован вход Start, станет доступным таймер барабанного командоаппарата. Таймер работает, пока имеет место событие для данного шага. Когда номер единицы шага становится равным значению единиц счета на шаг, командоаппарат переходит к следующему шагу. Этот процесс останавливается при завершении последнего шага или при инициализации входа Reset. Командоаппарат переходит к выбранному предварительно установленному шагу при переходе процессора в Рабочий режим программы, а также при каждой инициализации входа Reset.

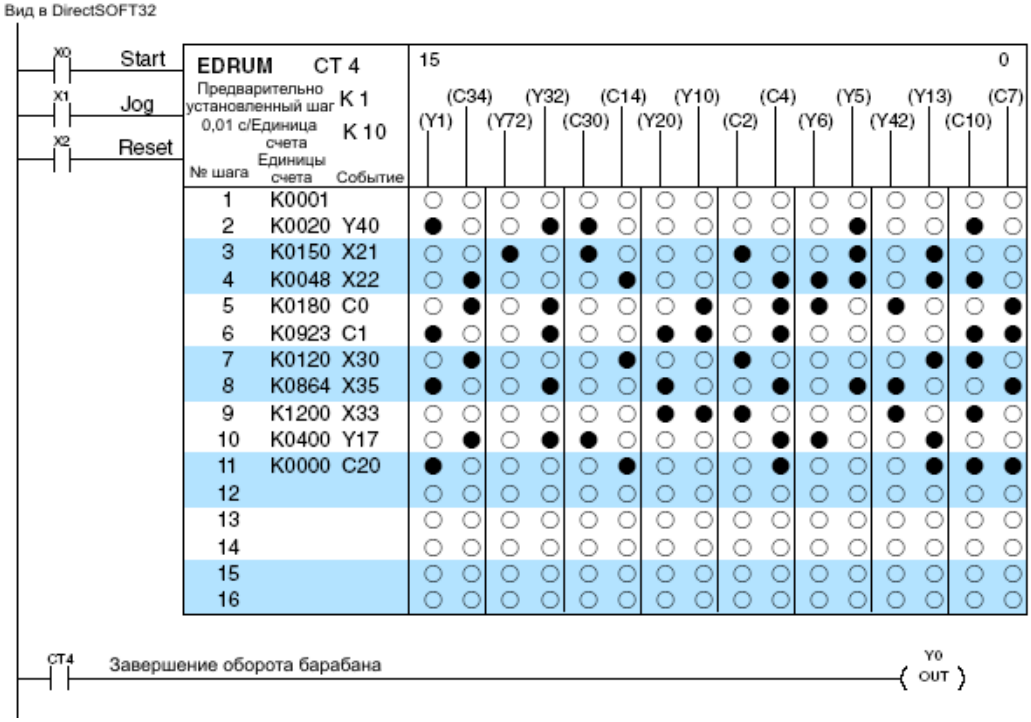
Параметры барабанного командоаппарата	Поле	Типы данных	Диапазон
Номер счетчика	aaa	K	0 – 174 (DL250-1) 0 – 374 (DL260)
Шаг предварительной установки	bb	K	1 – 16
Временной масштаб	cccc	K	0 - 99.99 секунд
Число тактов на шаг	dddd	K	0 - 9999
Событие	Eeeee	X, Y, C, S, T, ST, GX, GY	см. карту памяти
Дискретные выходы	Fffff	X, Y, C, GX, GY	см. карту памяти

Команды барабанного командоаппарата используют четыре счетчика ЦПУ. Программа может читать значения счетчиков, определяя текущее состояние барабанного командоаппарата. Программа также может в любой момент записать в CTA(n+2) новый номер шага предварительной установки. Что касается других счетчиков, они служат только для целей контроля.

Номер счетчика	Диапазон для DL250-1	Диапазон для DL260	Функция	Функция бита счетчика
CTA(n)	0 -174	0 - 374	Текущее число тактов шага	CTn = Окончание цикла
CTA(n+1)	1 -175	1 - 375	Значение счетчика времени	CT(n+1) = (не используется)
CTA(n+2)	2-176	2 - 376	Шаг предварительной установки	CT(n+2) = (не используется)
CTA(n+3)	3 -177	3 - 377	Текущий шаг	CT(n+3) = (не используется)

На следующем рисунке показано представление типичной релейной программы, использующей команду EDRUM, в *Direct*SOFT. Используются шаги 1…11, задействованы все 16 выходных точек. Шагом предварительной установки является Шаг 1. Длительность одного такта (временной масштаб) = (K10 x 0.01) = 0.1 секунды. Следовательно, длительность Шага 1 составляет (1 x 0.1) 0.1 секунды. Обратите внимание, что переход от Шага 1 происходит только по времени (поле "событие" не заполнено). Кроме того, на Стадии 1 все выходы выключены. Чаще всего именно такое состояние должно устанавливаться при включении питания. В последней цепи программы бит завершения цикла (CT4) устанавливает выход Y0 по завершении последнего шага (Шага 11). Барабанный командоаппарат сбрасывается, сбрасывая также CT4.

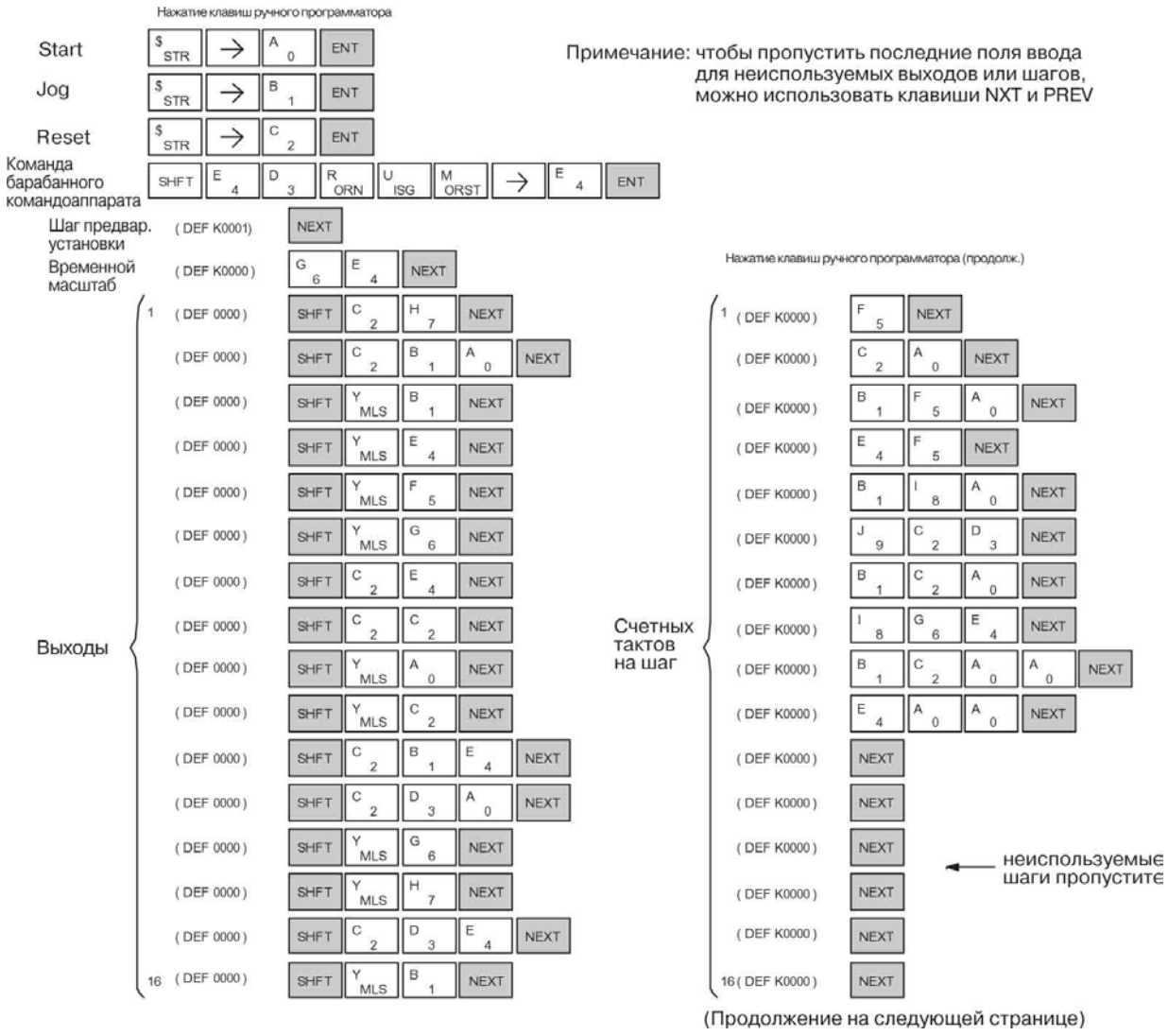
Вид в DirectSOFT32

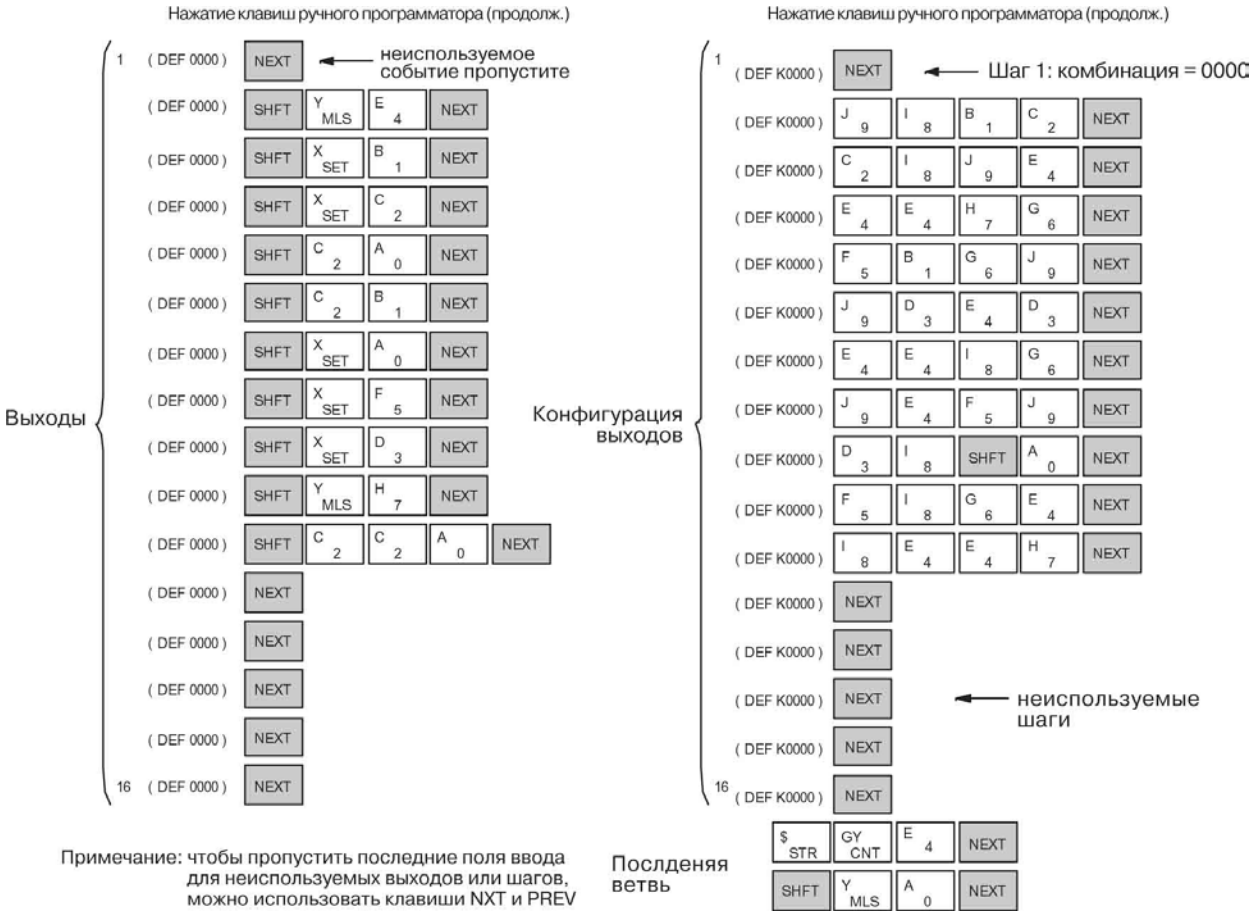


ПРИМЕЧАНИЕ: Если в барабанном командоаппарате только по событиям все события истинны (при числе тактов на всех шагах =0), то процессорный модуль исполняет один шаг за один цикл; следовательно, полный оборот барабана будет завершен за 16 циклов. Однако, поскольку выходы барабана включены все время, пока процессор находится в режиме выполнения программы, дискретные выходы будут включаться как импульсные выходы в каждом цикле.

Для ввода или редактирования команд барабанного командоаппарата можно также использовать Ручной Программатор, но только для EDRUM. На приведенной ниже диаграмме перечислены нажатия клавиш для ввода примера команды, описанного на предыдущей странице.

ПРИМЕЧАНИЕ: Для редактирования с помощью Ручного Программатора требуется, чтобы в его ПЗУ было «зашито» программное обеспечение версии 1.8 или более поздней.



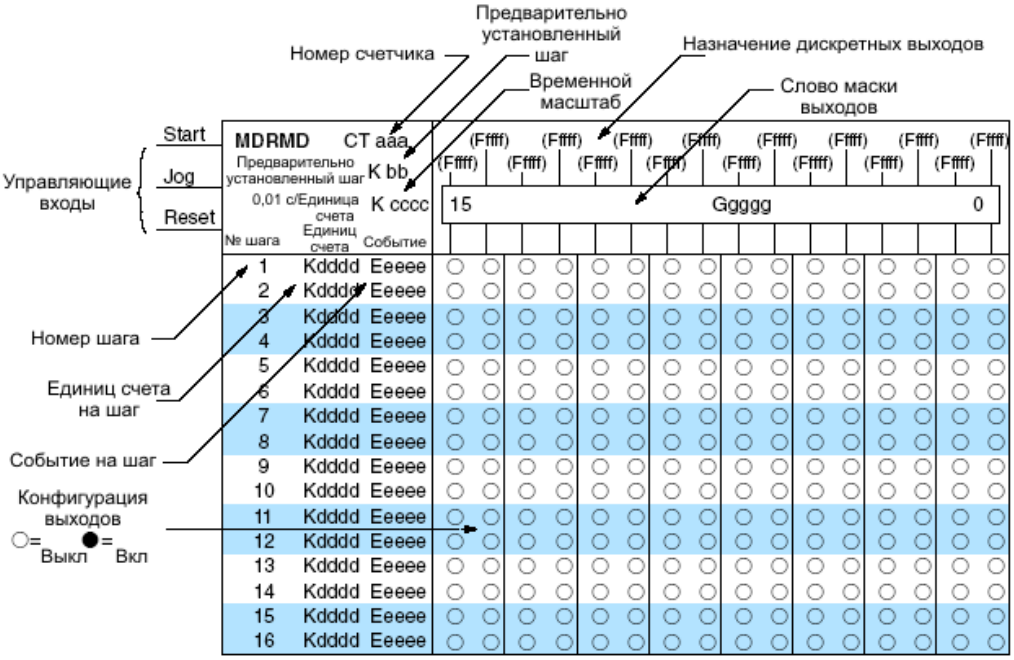

Нажатие клавиш ручного программатора (продолж.)
Выходы
1 (DEF 0000) NEXT ← неиспользуемое событие пропустите
(DEF 0000) SHFT Y MLS E 4 NEXT
(DEF 0000) SHFT X SET B 1 NEXT
(DEF 0000) SHFT X SET C 2 NEXT
(DEF 0000) SHFT C 2 A 0 NEXT
(DEF 0000) SHFT C 2 B 1 NEXT
(DEF 0000) SHFT X SET A 0 NEXT
(DEF 0000) SHFT X SET F 5 NEXT
(DEF 0000) SHFT X SET D 3 NEXT
(DEF 0000) SHFT Y MLS H 7 NEXT
(DEF 0000) SHFT C 2 C 2 A 0 NEXT
(DEF 0000) NEXT
(DEF 0000) NEXT
(DEF 0000) NEXT
(DEF 0000) NEXT
16 (DEF 0000) NEXT
Нажатие клавиш ручного программатора (продолж.)
Конфигурация выходов
1 (DEF K0000) NEXT ← Шаг 1: комбинация = 000C
(DEF K0000) J 9 I 8 B 1 C 2 NEXT
(DEF K0000) C 2 I 8 J 9 E 4 NEXT
(DEF K0000) E 4 E 4 H 7 G 6 NEXT
(DEF K0000) F 5 B 1 G 6 J 9 NEXT
(DEF K0000) J 9 D 3 E 4 D 3 NEXT
(DEF K0000) E 4 E 4 I 8 G 6 NEXT
(DEF K0000) J 9 E 4 F 5 J 9 NEXT
(DEF K0000) D 3 I 8 SHFT A 0 NEXT
(DEF K0000) F 5 I 8 G 6 E 4 NEXT
(DEF K0000) I 8 E 4 E 4 H 7 NEXT
(DEF K0000) NEXT
(DEF K0000) NEXT
(DEF K0000) NEXT ← неиспользуемые шаги
(DEF K0000) NEXT
16 (DEF K0000) NEXT
Послденяя ветвь
$ STR GY CNT E 4 NEXT
SHFT Y MLS A 0 NEXT
Примечание: чтобы пропустить последние поля ввода для неиспользуемых выходов или шагов, можно использовать клавиши NXT и PREV

Маскированный барабанный командоаппарат с дискретными выходами и переходом по событию (MDRMD)

✕	✕	✓	✓
230	240	250–1	260

Маскированный б**арабанный командоаппарат с дискретными выходамим и переходом по событию** обладает всеми свойствами базовой команды барабанного командоаппарата с переходами по событию, а также дополнительным свойством, которое заключается в формировании конечного состояния выходов на каждом шаге с использованием маски. Он работает в соответствии с общими правилами барабанного командоаппарата, изложенными в начале данного раздела. В *Direct*SOFT команда отражается в виде следующей диаграммы.

Барабанный командоаппарат **MDRMD** обеспечивает 16 шагов и 16 выходов. На каждом шаге выполняется побитовое логическое "И" выходов барабана и слова маски выходов. Позиция первого из 16-ти слов маски указывается в поле Ggggg. Переходы происходят по времени и/или по событию. Положительный фронт на входе Jog (переход из ВЫКЛ во ВКЛ) также переводит барабан на один шаг вперед. Длительность шага указывается в тактах, а в качестве событий указываются дискретные контакты. Поля для неиспользуемых шагов и событий можно оставить пустыми (они пустые по умолчанию). неиспользуемые шаги и события можно оставить пустыми (значение по умолчанию).



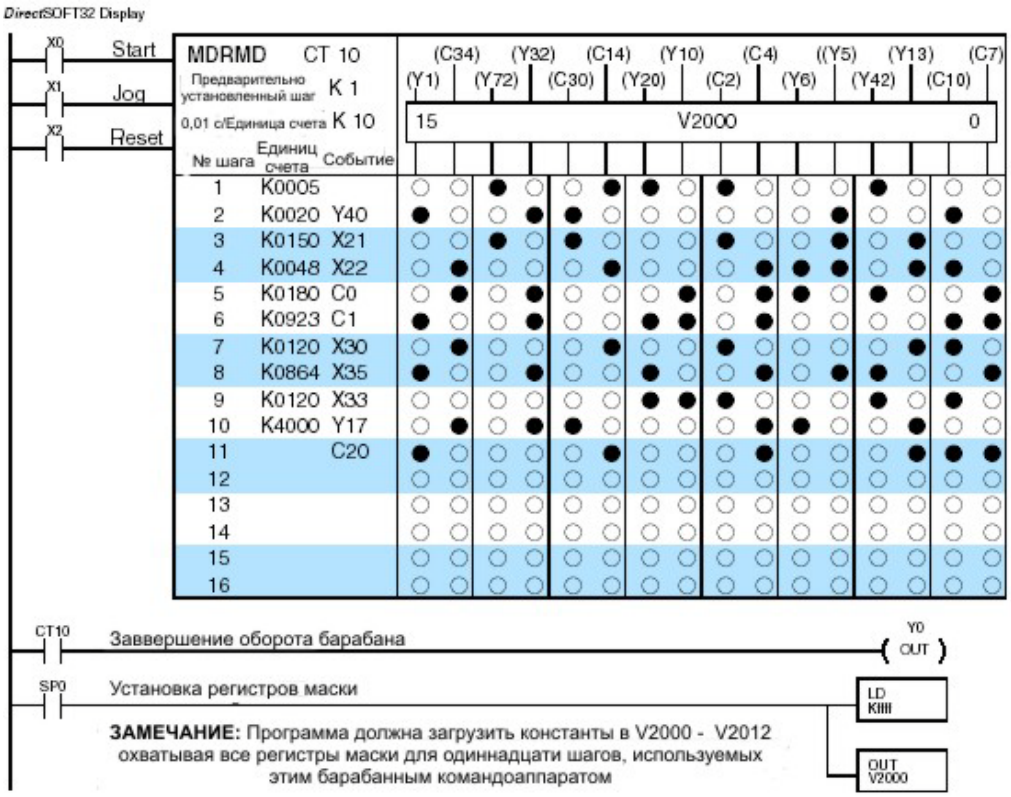
Когда вход Start включен, разрешается работа счетчика барабанного командоаппарата. Пока критерий события для текущего шага выполняется, происходит наращивание таймера. Когда количество тактов достигает порогового значения для данного шага, барабанный командоаппарат переходит к следующему шагу. Процесс прекращается, когда завершается последний шаг, либо когда включается вход Reset. При переходе ЦПУ из режима программирования в режим выполнения либо при включении входа Reset барабанный командоаппарат возвращается к Шагу предварительной установки.

Параметры барабанного командоаппарата		Типы данных	Диапазон
Номер счетчика	aaa	K	0 – 174 (DL250-1) 0 – 374 (DL260)
Шаг предварительной установки	bb	K	1 – 16
Временной масштаб	cccc	K	0 - 99.99 секунд
Число тактов на шаг	dddd	K	0 - 9999
Событие	Eeeee	X, Y, C, S, T, ST, GX, GY	см. карту памяти
Дискретные выходы	Fffff	X, Y, C, GX, GY	
Маска выходов	Ggggg	V	

Команды барабанного командоаппарата используют четыре счетчика ЦПУ. Программа может читать значения счетчиков, определяя текущее состояние барабанного командоаппарата. Программа также может в любой момент записать в CTA(n+2) новый номер шага предварительной установки или новый текущий номер шага в CTA(n+3). Что касается других счетчиков, они служат только для целей контроля.

Номер счетчика	Диапазон для DL250-1	Диапазон для DL260	Функция	Функция бита счетчика
CTA(n)	0 -174	0 - 374	Текущее число тактов шага	CTn = Окончание цикла
CTA(n+1)	1 -175	1 - 375	Значение счетчика времени	CT(n+1) = (не используется)
CTA(n+2)	2 -176	2 - 376	Шаг предварительной установки	CT(n+2) = (не используется)
CTA(n+3)	3 -177	3- 377	Текущий шаг	CT(n+3) = (не используется)

На следующем рисунке показано представление типичной программы релейной логики, использующей команду MDRMD, в *Direct*SOFT. Используются шаги 1…11 и все 16 выходных точек. Словом маски выходов является V2000. Конечное состояние выходов барабанного командоаппарата показано над словом маски. Биты слова V2000 складываются с отдельными выходами текущего шага барабанного командоаппарата по правилам операции "логическое И". Если требуется, чтобы все выходы барабанного командоаппарата были выключены при включении питания, в первом цикле в V2000 следует записать нули. Релейная программа может изменить значение маски выходов в любое время, чтобы активизировать или деактивизировать соответствующие выходы барабанного командоаппарата. Шагом предварительной установки является Шаг 1. Длительность одного такта (временной масштаб) составляет (K10 x 0.01) = 0.1 секунды. Следовательно, длительность Шага 1 = (5 x 0.1) = 0.5 секунды. Заметим, что переход из Шага 1 происходит только по времени (поле события оставлено пустым). В последней цепи бит завершения цикла (CT10) устанавливает выход Y0 по завершении последнего шага (Шага 10). Барабан сбрасывается, одновременно сбрасывая CT10.

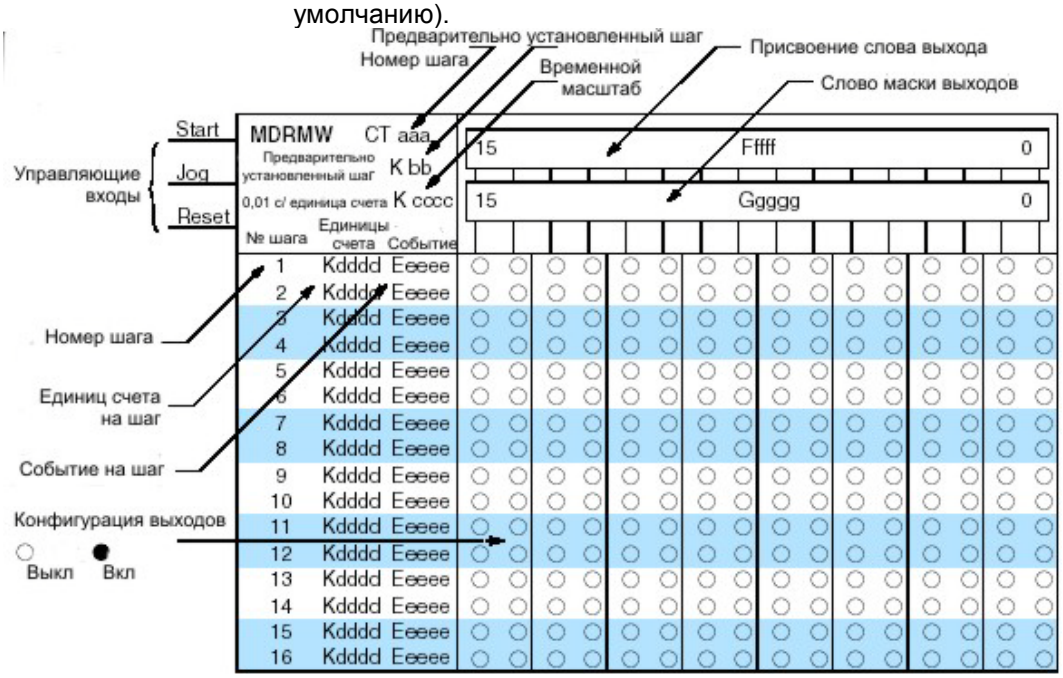


Маскированный барабанный командоаппарат с выходом по словам и переходом по событию (MDRMW)

✕	✕	✓	✓
230	240	250–1	260

В маскированном барабанном командоаппарате с выходом по словам и переходом по событию, выходы организованы не как дискретные точки, а как биты одного слова. Он работает согласно общим правилам барабанного командоаппарата, изложенным в начале данного раздела. В *Direct*SOFT эта команда отображается в виде следующей диаграммы.

Барабанный командоаппарат MDRMW обеспечивает 16 шагов и 16 выходов. На каждом шаге выполняется побитовое "логическое И" выходов барабана и слова маски выходов. Позиция первого из 16-ти слов маски указывается в поле Ggggg. Результирующее состояние выходов определяется словом маски (поле Fffff). Переходы шагов происходят по времени и/или по событию. Положительный фронт на входе Jog (переход из ВЫКЛ во ВКЛ) также переводит барабан на один шаг вперед. Длительность шага указывается в тактах, а в качестве событий указываются дискретные контакты. Поля для неиспользуемых шагов и событий можно оставить пустыми (они пустые по умолчанию).



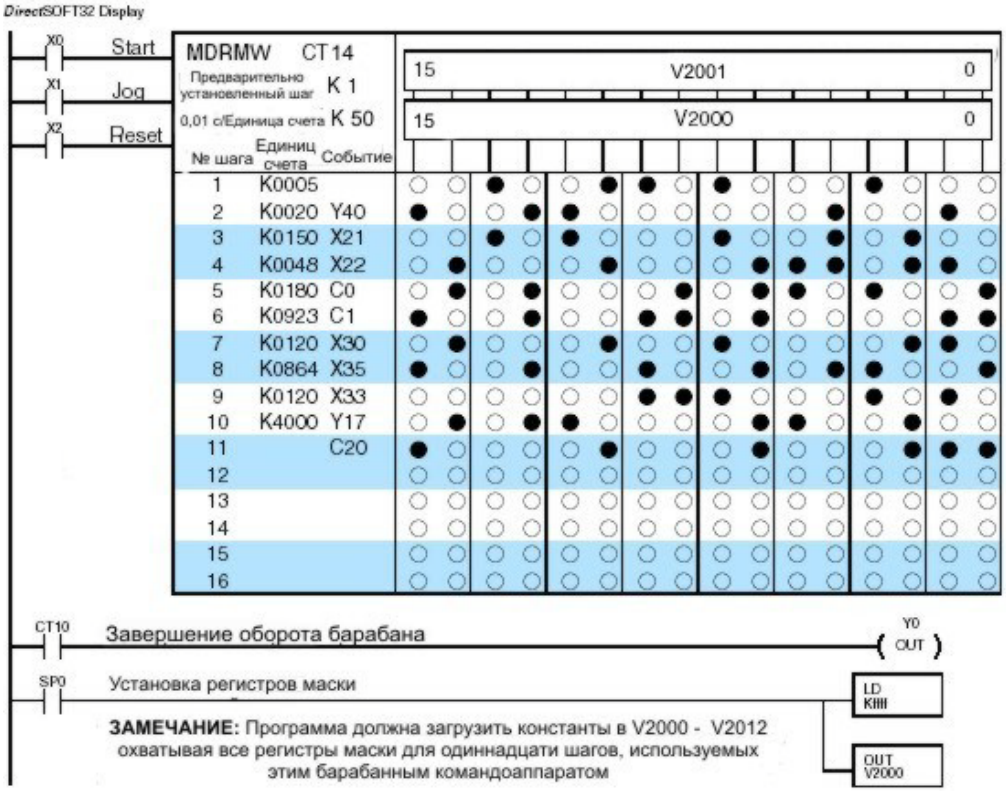
Когда вход Start включен, разрешается работа счетчика барабанного командоаппарата. Пока критерий события для текущего шага выполняется, происходит наращивание таймера. Когда количество тактов достигает порогового значения для данного шага, барабанный командоаппарат переходит к следующему шагу. Процесс прекращается, когда завершается последний шаг, либо когда включается вход Reset. При переходе ЦПУ из режима программирования в режим выполнения либо при включении входа Reset барабанный командоаппарат возвращается к Шагу предварительной установки.

Параметры барабанного командоаппарата		**Типы данных**	**Диапазон**
Номер счетчика	aaa	K	0 – 174 (DL250-1) 0 – 374 (DL260)
Шаг предварительной установки	bb	K	1 – 16
Временной масштаб	cccc	K	0 - 99.99 секунд
Число тактов на шаг	dddd	K	0 - 9999
Событие	Eeeee	X, Y, C, S, T, ST, GX, GY	см. карту памяти
Выходное слово	Fffff	V	
Маска выходов	Ggggg	V	

Команды барабанного командоаппарата используют четыре счетчика ЦПУ. Программа может читать значения счетчиков, определяя текущее состояние барабанного командоаппарата. Программа также может в любой момент записать в CTA(n+2) новый номер шага предварительной установки или новый текущий номер шага в CTA(n+3). Что касается других счетчиков, они служат только для целей контроля.

Номер счетчика	Диапазон для DL250-1	Диапазон для DL260	Функция	Функция бита счетчика
CTA(n)	0 -174	0 - 374	Текущее число тактов шага	CTn = Окончание цикла
CTA(n+1)	1 -175	1 - 375	Значение счетчика времени	CT(n+1) = (не используется)
CTA(n+2)	2 -176	2 - 376	Шаг предварительной установки	CT(n+2) = (не используется)
CTA(n+3)	3 -177	3 - 377	Текущий шаг	CT(n+3) = (не используется)

На следующем рисунке показано представление типичной программы релейной логики, использующей команду MDRMW, в *Direct*SOFT. Используются шаги 1…11 и все 16 выходных точек. Словом маски выходов является V2000. Результирующее состояние выходов барабанного командоаппарата показано над словом маски как слово V2001. Биты слова V2000 складываются с отдельными выходами текущего шага барабанного командоаппарата по правилам операции "логическое И", формируя содержание слова V2001. Если требуется, чтобы все выходы барабанного командоаппарата были выключены при включении питания, в первом цикле в V2000 следует записать нули. Релейная программа может изменить значение маски выходов в любое время, чтобы активизировать или деактивизировать соответствующие выходы барабанного командоаппарата. Шагом предварительной установки является Шаг 1. Длительность одного такта (временной масштаб) составляет (50 x 0.01) 0.5 с. Следовательно, длительность Шага 1 = (5 x 0.5) =2.5 с. Заметим, что переход из Шага 1 происходит только по времени (поле события оставлено пустым). В последней цепи бит завершения цикла (CT14) устанавливает выход Y0 по завершении последнего шага (Шага 10). Барабан сбрасывается, одновременно сбрасывая CT14.



Глава 7. Стадийное программирование RLL^{PLUS}

В этой главе…

- Введение в стадийное программирование
- Как составлять диаграммы перехода состояния
- Использование команды Stage Jump для переходов состояний
- Пример стадийной программы: контроллер включения / выключения лампочки с помощью контроллера
- Четыре действия для создания стадийной программы
- Пример стадийной программы: управление дверью гаража
- Правила создания стадийных программ
- Принципы параллельного выполнения процессов
- Управление большими программами
- Команды RLL^{PLUS}
- Стадийное программирование в вопросах и ответах

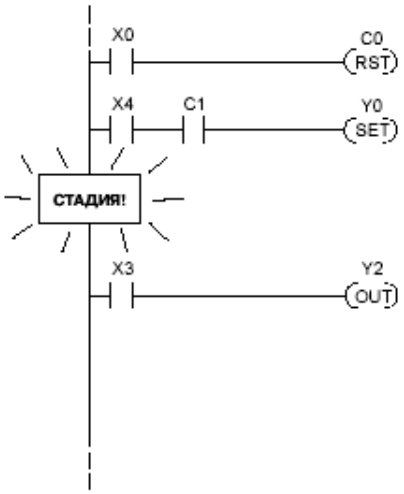
Введение в стадийное программирование

Стадийное программирование по сравнению с решениями, использующими язык релейной логики (RRL) в чистом виде, предоставляет более легкий способ организации и построения программ для решения сложных прикладных задач. Стадийное программирование не заменяет и не отменяет использование традиционных программ на языке релейной логики. Именно поэтому стадийное программирование называют RLL^{PLUS} . Полученные ранее знания, а также предыдущий опыт будут полезны. Стадийное программирование всего лишь позволяет организовать RLL-программу в виде отдельных групп, каждая из которых состоит из команд обычных RLL-программ. Такие группы называются "стадиями". В результате разработка релейных программ становится более быстрой и интуитивно понятной в сравнении с традиционными языками релейной логики.

Стадийное программирование доступно для всех процессорных модулей серии DL205

Преодоление "боязни Стадий"

Программисты, работающие с промышленными контроллерами, привыкли к использованию языка релейной логики, применяя его для всех ПЛК, программы для которых они создают, и зачастую относятся скептически, а иногда и со страхом к изучению такой новой техники, как стадийное программирование. Обладая большими возможностями при решении задач, подчиняющихся правилам булевой алгебры, язык релейной логики обладает одновременно и рядом недостатков:



- Недостаточно структурированные большие программы могут стать практически неуправляемыми.
- В RLL защелки приходится утомительно создавать из реле с самоблокировкой.
- Когда процесс заходит в тупик, бывает очень сложно найти цепь, являющуюся причиной ошибки.
- Программы сложно модифицировать, поскольку между ними и решаемыми прикладными задачами нет интуитивного соответствия.

Легко понять, что данные недостатки отнимают много времени. Преодолеть их позволяет стадийное программирование! Мы убеждены, что несколько минут, потраченные на изучение концепции стадии, являются наилучшим вложением в скорость и эффективность программирования, которое может осуществить программист ПЛК!

Мы призываем вас изучить стадийное программирование, чтобы добавить его в свой «инструментарий» методов программирования.

Чтобы достичь наилучших результатов:

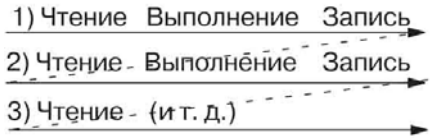
- Начните с самого начала и не пропускайте ни одного раздела.
- Изучайте каждый принцип стадийного программирования, разбирая все примеры. Примеры выстроены последовательно один за другим.
- Прочитайте раздел "Стадийное программирование в вопросах и ответах", содержащий краткий обзор изложенного материала.

Как составлять диаграммы перехода состояния

Знакомство с понятием "состояние процесса"

Тому, кто знаком с принципом исполнения программ, написанных на языке релейной логики, известно, что ЦПУ должен исполнять релейную программу последовательно и непрерывно, цикл за циклом. Любой цикл состоит из трех основных этапов:

- Чтение входов
- Выполнение программы
- Запись выходов



Преимущество состоит в том, что изменение состояния на входе может привести к изменению состояния на выходе всего за несколько миллисекунд.

Большинство производственных процессов состоит из серии операций или состояний, каждое из которых длится несколько секунд, минут или даже часов. Такие состояния можно назвать "состояниями процесса". Каждое "состояние процесса" в любой момент времени является либо активным, либо неактивным. Узким местом при создании RLL-программ является тот факт, что отдельное событие на входе может длиться всего лишь короткое мгновение. Чтобы не потерять входное событие и запомнить состояние процесса на требуемое время, в RLL-программе, как правило, предусматриваются реле с самоблокировкой.
Релейную программу можно разбить на отдельные секции, называемые "Стадиями". Каждая "Стадия" представляет собой состояние процесса. Но прежде чем перейти к подробному описанию программирования Стадий, мы откроем один секрет, который позволит нам понять сущность стадийного программирования, и этот секрет - **диаграммы переходов состояний.**

Зачем нужны диаграммы состояний

Иногда необходимо отвлечься от циклического принципа работы ПЛК и сосредоточиться на состояниях процесса, которые требуется идентифицировать. Ясное понимание и четкий анализ прикладной задачи - вот наилучший способ создания эффективных программ, не содержащих ошибок. **Диаграмма состояний - это всего лишь инструмент, который позволяет нам наглядно представить картину автоматизируемого процесса!** Вы вскоре убедитесь, что если картина процесса была воссоздана правильно, то будет правильной и создаваемая программа.

Процесс с двумя состояниями

На рисунке справа показан простой процесс, заключающийся в управлении промышленным двигателем. Для включения двигателя будем использовать зеленую кнопку без фиксации, а для отключения - красную кнопку. Управляя установкой, оператор нажимает на соответствующую кнопку. Управление длится около секунды. Таким образом, в процессе имеется два состояния: ВКЛ и ВЫКЛ.

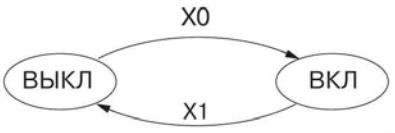
Выражение для выхода: Y0 = ВКЛ

На следующем этапе мы рисуем диаграмму переходов состояний, показанную на рисунке справа внизу. На диаграмме показаны два состояния, ВЫКЛ и ВКЛ, объединенные двумя линиями переходов. Когда верно событие X0, происходит переход от ВЫКЛ к ВКЛ. Когда верно X1, происходит переход от ВКЛ к ВЫКЛ.

Внимательно следуя за ходом мысли, вы, должно быть, уже поняли, в чем состоит идея диаграмм переходов состояний, и в чем заключается их сила. В нашем примере контроллер имеет выход Y0, который верен (находится в состоянии логической "1"), когда наступает состояние ВКЛ. Логическое выражение для выхода: Y0 = ВКЛ.

Теперь попробуем реализовать диаграмму состояний в виде RLL, а затем в виде стадийной программы. Это позволит понять взаимосвязь между рассматриваемыми способами решения задачи.

Показанная справа диаграмма переходов состояний - есть не что иное, как требующаяся нам картина решаемой задачи. Вся прелесть в том, что она иллюстрирует задачу абсолютно независимо от использующегося нами языка программирования. Другими словами, **нарисовав диаграмму, мы в результате решим задачу управления!**

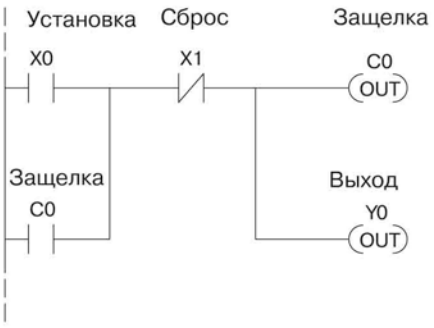


Выражение для выхода: Y0 = ВКЛ

Для начала переведем диаграмму состояний на традиционный язык релейной логики. Затем будет показано, насколько просто можно будет перевести диаграмму на язык стадийного программирования.

Эквивалент на языке RLL

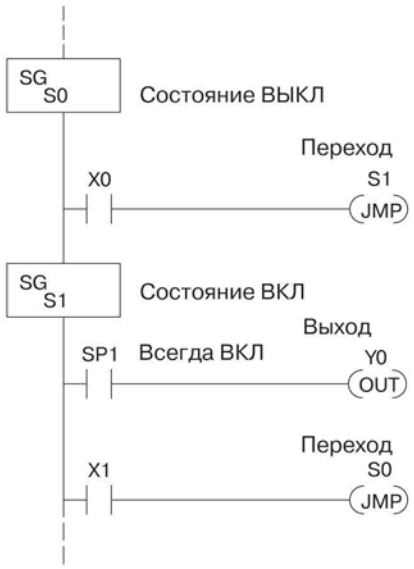
Решение в виде RLL-программы показано справа. В состав программы входит реле управления C0 с самоблокировкой. Когда нажата кнопка включения (X0), возбуждается выходная обмотка C0, и контакт C0 во второй цепи переходит во включенное состояние с самоблокировкой. Итак, X0 **переводит защелку C0 во включенное состояние**, и последняя остается включенной после размыкания контакта X0. Выход управления двигателем Y0 также будет запитан, поэтому двигатель будет включен.



Нажатие кнопки выключения (X1) приводит к размыканию нормально замкнутого контакта X1, который **сбрасывает защелку**. Выход управления двигателем Y0 отключается, когда отключается обмотка защелки C0.

Эквивалент на языке стадийного программирования

Решение в виде стадийной программы показано на рисунке справа. Элементы Стадий S0 и S1, включенные последовательно в левую шину, соответствуют состояниям ВЫКЛ и ВКЛ. Цепи релейной программы относятся к той Стадии (прямоугольному элементу), под которой они расположены. Это означает, что ПЛК должен опрашивать цепи только тогда, когда соответствующая Стадия, под которой они расположены, является активной!



Предположим теперь, что работа начинается в состоянии ВЫКЛ, т.е., активной является Стадия S0. По нажатию кнопки включения (X0) происходит переход к следующей Стадии (переключение состояния). Выполняется команда JMP S1, которая просто сбрасывает бит Стадии S0 и устанавливает бит Стадии S1. Поэтому в следующем цикле ПЛК ЦПУ не будет выполнять Стадию S0, а выполнит Стадию S1.

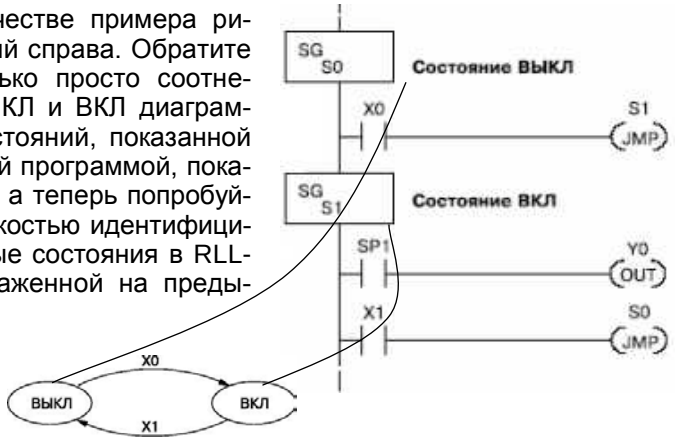
Мы хотим, чтобы в состоянии включения (S1) двигатель работал. Поскольку специальный контакт реле SP1 всегда включен, Y0 включит двигатель.

По нажатию кнопки выключения (X1) происходит возврат к состоянию ВЫКЛ. Выполняется инструкция JMP S0, которая просто сбрасывает бит Стадии S1 и устанавливает бит Стадии S0. В следующем цикле ПЛК ЦПУ не исполняет Стадию S1, поэтому выход управления двигателем Y0 выключается. Состояние выключения (Стадия 0) наступит в следующем цикле.

Давайте сравним

Сейчас вы, возможно, не видите особых преимуществ в стадийном программировании, и стадийная программа кажется длиннее простой RLL. Ну что ж, самое время довериться нам. С возрастанием сложности задач управления стадийное программирование быстро начинает превосходить классическое RLL-программирование с точки зрения простоты, объема программы и т. п.

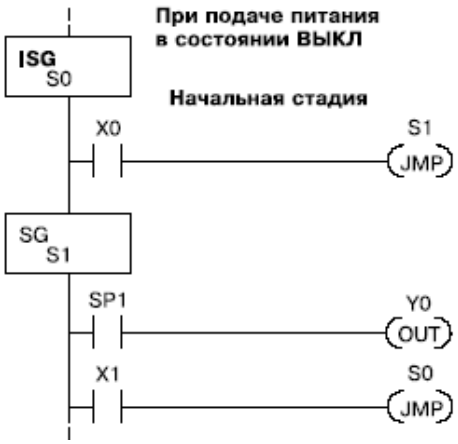
Рассмотрим в качестве примера рисунок, приведенный справа. Обратите внимание, насколько просто соотнести состояния ВЫКЛ и ВКЛ диаграммы переходов состояний, показанной ниже, со стадийной программой, показанной справа. Ну а теперь попробуйте-ка с той же легкостью идентифицировать аналогичные состояния в RLL-программе, изображенной на предыдущей странице!



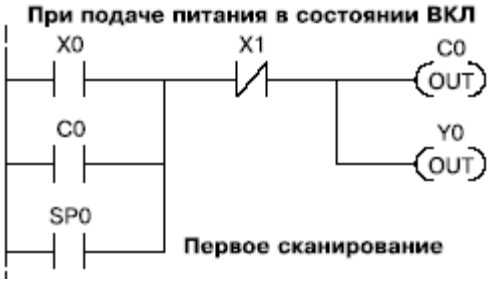
Начальные Стадии

При включении питания и при переходе программы в режим исполнения ПЛК всегда начинает работу с отключенными Стадиями рабочего режима (SG). Поэтому исполнение показанных стадийных программ, на самом деле, начаться не могло (поскольку цепи не опрашиваются, если их Стадии не активны).

Предположим, что нам требуется всегда начинать работу в состоянии ВЫКЛ (двигатель не работает), т.е., именно так, как работают RLL-программы. Указывается, что Начальная Стадия (ISG) является активной после включения питания. Справа показана измененная программа, в которой Стадии S0 присвоен тип ISG (Начальная Стадия). В результате ПЛК гарантированно опросит контакт X0 после включения питания, поскольку Стадия S0 является активной. **После включения питания Начальная Стадия (ISG) работает так же, как и любая другая Стадия!**



Можно изменить обе программы таким образом, чтобы двигатель был включен после включения питания. В приведенной ниже RLL-программе следует добавить реле SP0, опрашиваемое в первом цикле, которое включает самоблокирующуюся защелку C0. В стадийной программе справа мы всего лишь делаем Стадию S1 Начальной Стадией (ISG) вместо Стадии S0.



Состояние, соответствующее включению питания, можно пометить, как показано на рисунке справа, это позволит не забыть, какие Стадии следует делать Начальными при создании стадийной программы. Количество Начальных Стадий может быть любым и определяется требованиями процесса.

Что делают Биты «Включения Стадий»

Вспомним, что Стадия — это всего лишь участок релейной программы, который либо активен, либо неактивен в данный момент времени. Все Биты Включения Стадий (S0 — Sxxx) располагаются в регистре отображения ПЛК, как отдельные биты состояний. В любой момент времени каждый бит состояния находится в состоянии логического "0" или "1".

При исполнении программы цепи релейной программы читаются сверху вниз, справа налево. На рисунке ниже показано, к какому результату приводит состояние Бита «Включения Стадии». Цепи релейной программы, расположенные под обозначением Стадии, выполняются либо до следующего обозначения Стадии, либо до конца программы, принадлежащего Стадии 0. Эквивалентное представление показано справа. Когда S0 верно, активны две цепи.

- Если бит стадии S0=0, ее программные цепи не опрашиваются (не выполняются).
- Если бит стадии S0=1, ее программные цепи опрашиваются (выполняются).

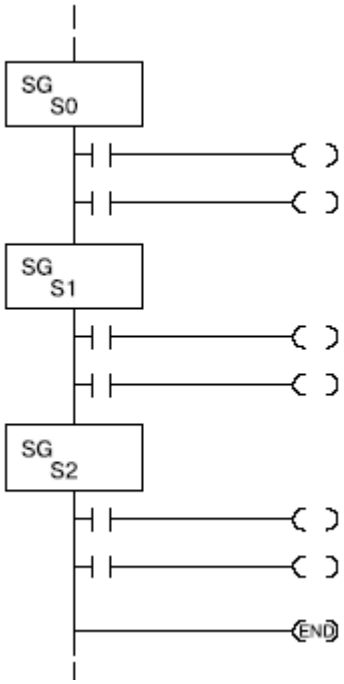
Свойства команды Стадия» (Stage)

Элементы "Стадия", включаемые последовательно в левую шину, группируют цепи релейной программы в отдельные Стадии. Вот несколько правил, которые применяются для Стадий.

Исполнение - В каждом цикле выполняются только цепи, относящиеся к активным Стадиям.

Переходы - Действие команды перехода к Стадии вступает в силу после того, как соответствующая Стадия встретится в следующий раз.

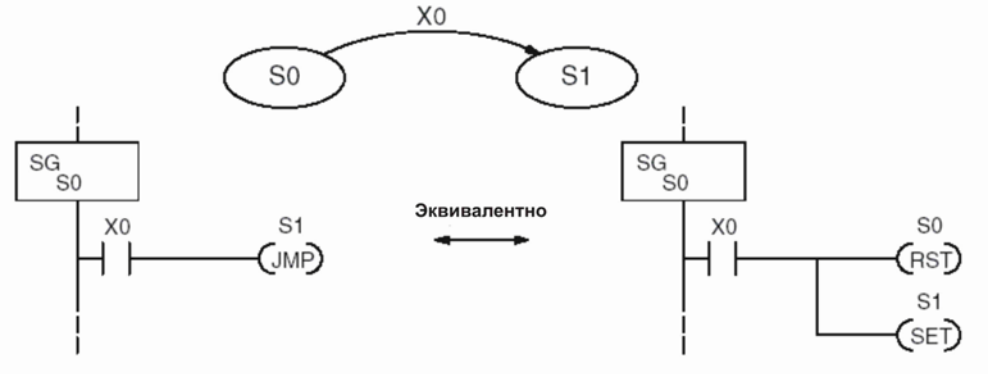
- **Восьмеричная нумерация** - Для нумерации Стадий, как и для точек ввода/вывода, используется восьмеричная система. Поэтому обозначение "S8" не допускается.
- **Общее количество Стадий** — максимально допустимое количество стадий зависит от процессора.
- **Дублирование не допускается** - Каждая Стадия имеет уникальный номер, который не может использоваться дважды.
- **Произвольный порядок** - Номера Стадий можно пропускать и располагать их в любом порядке.
- **Последняя Стадия** - Последняя Стадия релейной программы охватывает все цепи, расположенные под ее обозначением, до завершающей обмотки (END).



Использование команды Stage Jump для переходов

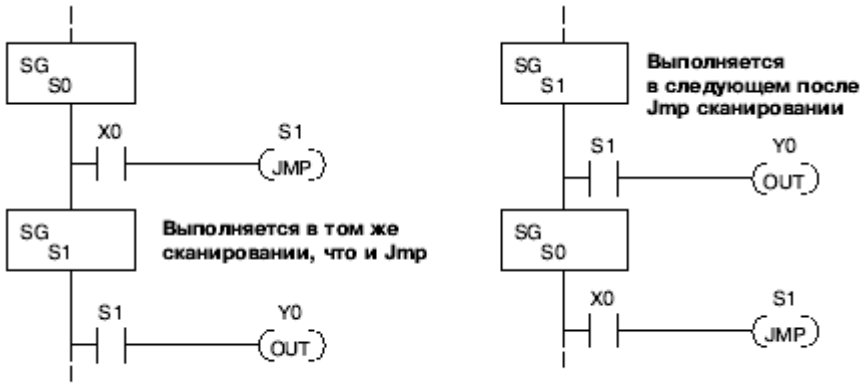
Команды перехода, установки и сброса стадий (Stage Jump, Set и Reset)

Использованная нами команда **Stage JMP** (Переход к Стадии) делает неактивной Стадию, в которой происходит выполнение команды, и активизирует Стадию, указанную в команде JMP. Обратимся к переходу состояния, показанному на рисунке ниже. Когда через контакт X0 начинает протекать ток, происходит переход из состояния S0 в S1. Две схемы смены состояний, показанные ниже, эквивалентны между собой. Таким образом, команда перехода к Стадии (Stage JMP) равносильна команде Stage Reset (Сброс Стадии), выполняемой для текущей Стадии одновременно с командой Stage Set (Активизация Стадии) для Стадии, к которой требуется перейти.



ПОДСКАЗКА: команду перехода можно легко понять неправильно. «Переход» не возникает сразу же после выполнения, как это происходит при использовании таких команд управления программой, как GOTO или GOSUB. Он выполняется следующим образом:

- Команда перехода сбрасывает Бит Включения Стадии для Стадии, в которой она выполняется. Все остальные цепи данной Стадии будут исполнены в текущем цикле, **даже если они находятся под цепью, в которой расположена команда перехода!**
- Сброс вступит в силу в следующем цикле, поэтому Стадия, в которой в предыдущем цикле была выполнена команда перехода, будет неактивна и пропущена.
- Бит Включения Стадии для Стадии, указанной в команде перехода, будет установлен сразу же, поэтому Стадия будет выполнена, когда встретится в программе в следующий раз. В программе слева Стадия S1 выполняется в том же цикле, в котором выполнилась команда JMP S1 в Стадии S0. На примере справа Стадия S1 происходит в следующем цикле после исполнения JMP S1, поскольку Стадия S1 расположена над Стадией S0.



Замечание: в обоих примерах предполагается, что в начале стадия 0 активна, а стадия 1 неактивна.

Пример стадийной программы: контроллер включения / выключения лампочки

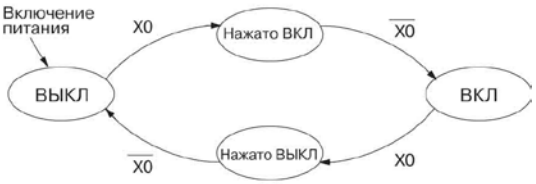
Процесс с четырьмя состояниями

На рисунке справа показан процесс, в котором для управления лампочкой используется одиночная кнопка без фиксации. В релейной программе на входе переключения будет предусмотрен триггер (защелка). Чтобы включить лампу, следует нажать и отпустить кнопку, чтобы выключить - вновь нажать и отпустить кнопку (что иногда называют функцией переключения). Разумеется, можно попросту купить механический переключатель, в котором предусмотрена функция переменного включения/ отключения. С другой стороны, мы хотим поучиться, а заодно и получить удовольствие! Затем мы нарисуем диаграмму переходов состояний.

Обычно первое желание — использовать X0 для обоих переходов (подобно примеру, показанному справа). Однако это некорректно (см. дальше).

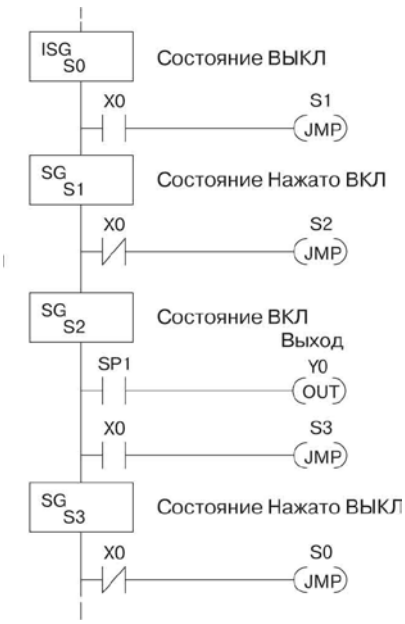
Обратите внимание, этот пример отличается от предыдущего примера с двигателем, поскольку в данном случае у нас всего одна кнопка. Когда мы нажимаем на кнопку, удовлетворяются оба условия смены состояний. Мы будем лишь циркулировать по диаграмме состояний на большой скорости. Стадийная программа, реализованная по такому принципу, в каждом цикле будет или включать, или выключать лампочку (что явно не желательно!).

Решение состоит в том, чтобы сделать нажатие и отпускание кнопки отдельными событиями. Обратимся к новой диаграмме переходов состояний, показанной ниже. После включения питания мы переходим к состоянию ВЫКЛ. Когда переключатель X0 будет нажат, происходит переход к состоянию Нажато ВКЛ. Когда переключатель будет отпущен, происходит переход к состоянию ВКЛ. Заметим, что символ X0 с расположенной над ним линией означает инверсию X0 (НЕ X0).



Следующие нажатие и отпускание кнопки, осуществленные в состоянии ВКЛ, аналогичным образом приведут нас назад к состоянию ВЫКЛ. В результате, после отпускания кнопки будут использованы два различных состояния (ВЫКЛ и ВКЛ), что, собственно, и требовалось для решения задачи управления.

Справа показана эквивалентная стадийная программа. После включения питания требуется установление состояния ВЫКЛ, поэтому Начальной Стадией (ISG) была сделана Стадия S0. В Стадию для состояния ВКЛ был добавлен контакт специального реле SP1, который всегда замкнут. Заметим, что даже тогда, когда программы становятся гораздо более сложными, сопоставить диаграмму переходов состояний стадийной программе по-прежнему легко.



Четыре действия для создания стадийной программы

К этому моменту Вы, должно быть, заметили, что для решения задачи в каждом примере мы выполняли одни и те же действия. Возможно, Вам удалось выделить эти действия самостоятельно, если Вы проработали все примеры данной главы. Будет полезно перечислить последовательность действий, которой можно будет руководствоваться при решении определенной задачи. Последовательность действий по созданию стадийной программы приводится ниже:

1. Напишите словесное описание решаемой задачи.

Опишите все функции процесса своими словами. Перечислите, что происходит сначала, что - потом и т. д. Если окажется, что слишком много событий происходит одновременно, попробуйте разделить задачу на несколько процессов. Помните, что можно по-прежнему реализовать взаимодействие между процессами, чтобы координировать весь процесс в целом.

2. Нарисуйте структурную схему

Входы соответствуют данным, которые требуются процессу для принятия решений, а выходы подключаются ко всем устройствам, управляемым процессом.

- Создайте списки входов и выходов, используемых в процессе.
- Пронумеруйте физические входы и выходы как точки ввода/вывода (X и Y).
-

3. Нарисуйте диаграмму перехода состояний.

Диаграмма переходов состояний описывает центральную функцию структурной схемы - чтение входов и управление выходами.

- Идентифицируйте и дайте названия состояниям процесса.
- Идентифицируйте событие (-я), необходимые для каждого перехода между состояниями.
- Обеспечьте возможность самоперезапуска процесса или сделайте процесс циклическим.
- Выберите для своего процесса состояние, устанавливающееся после включения питания.
- Напишите выражения для выходов.

4. Создайте стадийную программу.

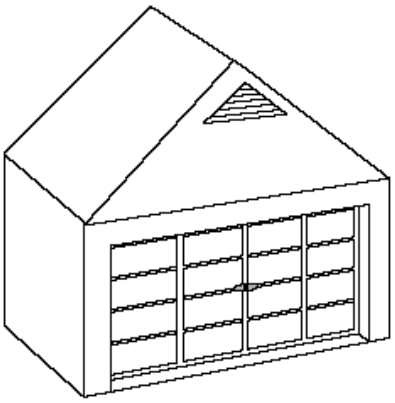
- Переведите диаграмму переходов состояний на язык стадийной программы.
- Каждое состояние сделайте стадией. Не забудьте, что нумерация стадий восьмеричная. В процессорах DL230 и DL240 доступно вплоть до 384 Стадий одновременно. В процессорах DL250-1 и DL-260 их общее количество может достигать 1024.
- В каждой Стадии должны быть запрограммированы условия, по которым происходит каждый переход (для каждой стрелочки, отходящей от Стадии (состояния)).
- Любое состояние, которое должно стать активным после включения питания, сделайте Начальной Стадией (ISG).
- В соответствующих Стадиях разместите выходы или действия.

Не сложно заметить, что действие 1…3 является всего лишь подготовкой к написанию стадийной программы (действие 4). С другой стороны, после таких подготовительных действий программу можно считать мысленно написанной. Очень скоро Вы сами сможете с легкостью создавать стадийные программы, начиная со словесного описания задачи.

Пример стадийной программы: управление дверью гаража

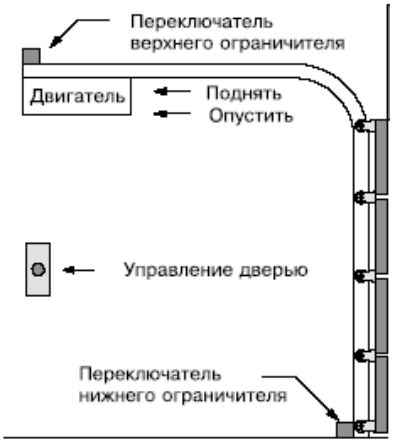
Опишем работу устройства

В нашем следующем примере мы создадим программу контроллера устройства открывания гаражной двери. Скорее всего, большинству читателей знакомо подобное устройство, и чтение помимо пользы принесет им удовольствие. Первым делом мы должны описать, как работает устройство открывания двери. Начнем с того, что опишем работу устройства в общем, а дополнительные функции добавим позже. Стадийные программы модифицируются очень легко.



В контроллере гаражной двери предусмотрен двигатель, который поднимает или опускает дверь по команде. Чтобы дверь поднялась, владелец гаража нажимает и отпускает кнопку. После того, как дверь поднята, последующее нажатие/отпускание приведет к закрытию двери.

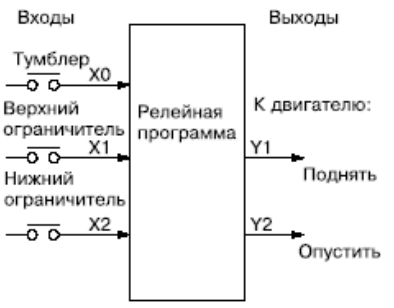
Чтобы определиться с входами и выходами системы, иногда бывает полезно набросать эскиз с основными элементами, как на рисунке справа, на котором показан вид со стороны двери. Сверху и снизу двери предусмотрены концевые выключатели. Такой выключатель замыкается только тогда, когда дверь достигла своего конечного положения в соответствующем направлении. В среднем положении двери ни один из переключателей не замкнут.



Двигатель имеет два входа управления: подъем и опускание. Когда ни один из входов не активен, двигатель остановлен. Командой на перемещение двери является простое нажатие на кнопку. Вмонтирована ли кнопка в стену, как показано на рисунке, или это кнопка пульта дистанционного управления, в любом случае все команды управления дверью объединяются по логическому ИЛИ, как если бы это была одна пара переключающих контактов.

Составим структурную схему

Структурная схема контроллера показана на рисунке справа. Кнопка управления дверью подключена ко входу X0. Вход X1 устанавливается, когда дверь полностью поднята. Вход X2 устанавливается тогда, когда дверь достигает крайней нижней позиции. Когда дверь находится где-то между нижним и верхним положениями, оба концевых выключателя разомкнуты.



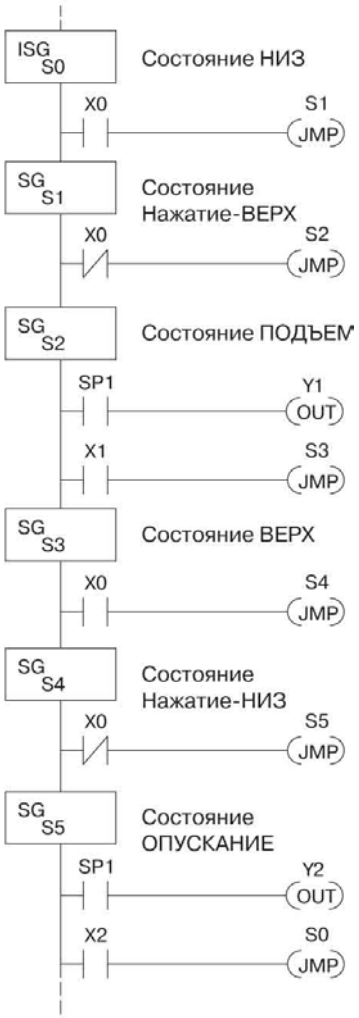
В контроллере предусмотрено два выхода для управления двигателем. Y1 служит командой подъема, а Y2 - командой опускания двери.

Нарисуем диаграмму состояния

Теперь мы можем приступить к созданию диаграммы переходов состояний. Как и в предыдущем примере с контроллером лампочки, в данном случае на входе управления также предусмотрен всего один переключатель. Обратимся к рисунку, показанному ниже.

- Когда дверь опущена (состояние НИЗ), ничего не происходит, пока не активизируется X0. Нажатие и отпускание переключателя на входе X0 приводит к переходу к состоянию ПОДЪЕМ, в котором устанавливается выход Y1, в результате чего двигатель поднимает дверь.
- Переход к состоянию ВЕРХ происходит, когда срабатывает концевой переключатель (X1), после чего двигатель выключается.
- После этого ничего не происходит, пока вновь не повторяется нажатие/отпускание X0. Это приводит к состоянию ОПУСКАНИЕ, в котором устанавливается выход Y2, заставляющий двигатель опускать дверь. Когда срабатывает конечный переключатель X2, вновь происходит переход к состоянию НИЗ.

Выражение для выходов: Y1 = ПОДЪЕМ Y2 = ОПУСКАНИЕ



Справа показана эквивалентная стадийная программа. Будем считать, что после включения питания дверь закрыта (опущена), поэтому включению питания должно соответствовать состояние НИЗ. Начальной Стадией (ISG) делаем Стадию S0. Стадия S0 остается активной до тех пор, пока не будет нажата кнопка управления дверью. После этого происходит переход (JMP) к состоянию НАЖАТИЕ-ВЕРХ (Стадия S1). Повторное нажатие/отпускание кнопки приводит к переходу от Стадии S1 к Стадии ПОДЪЕМ (S2). Для активизации команды подъема двигателя (Y1) используется постоянно замкнутый контакт SP1. Когда дверь полностью поднята, срабатывает концевой переключатель X1. Это приводит нас к Стадии ВЕРХ (S3), в которой мы ожидаем поступления новой команды управления дверью.
В состоянии ВЕРХ (S3) нажатие/отпускание кнопки приведет к Стадии ОПУСКАНИЕ (S5), в которой активизируется выход Y2, являющийся командой для двигателя на опускание двери. Опускание происходит до тех пор, пока дверь не достигает крайнего нижнего положения (срабатывает нижний переключатель X2). Когда X2 замыкается, происходит переход от Стадии X5 к Стадии НИЗ (S0), с которой мы начали.

ПРИМЕЧАНИЕ: *Единственным отличием Начальной Стадии (ISG) является то, что она автоматически становится активной по включению питания. Во всем остальном она ничем не отличается от других Стадий.*

Добавим сигнальную лампу

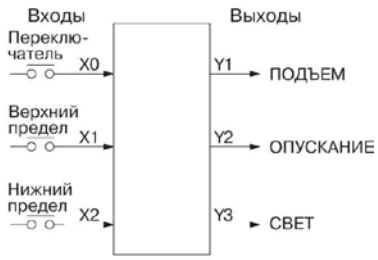
Теперь добавим к устройству открытия двери сигнальную лампу. Мы уже добились выполнения устройством главной функции, и теперь можем снабжать его дополнительными функциями. Такая последовательность действий наиболее разумна.

Сигнальная лампа предусмотрена во многих системах открытия гаражной двери, имеющихся в продаже. Сигнальная лампа крепится к кожуху двигателя, как показано на рисунке справа. Лампа включается после перемещения двери в любом направлении, и остается включенной после этого, приблизительно, три минуты.

На этом примере будет показано использование параллельных состояний в нашей диаграмме состояний. Вместо команды JMP мы будем использовать команды установки и сброса (Set и Reset).

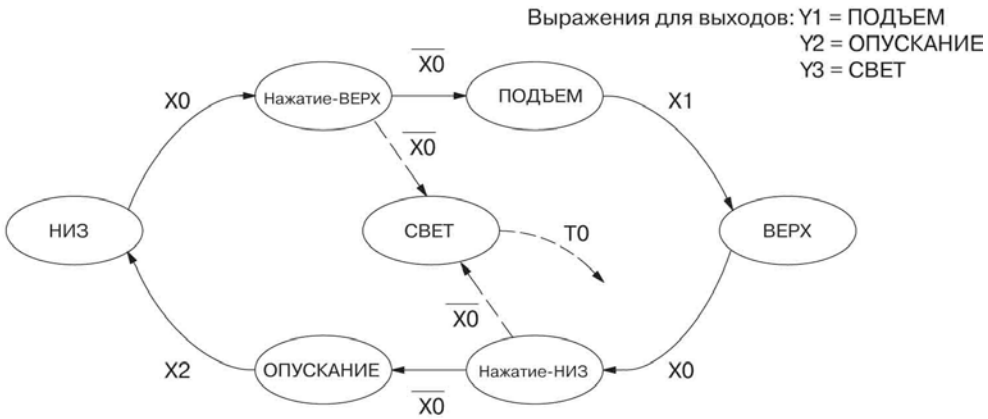
Изменим структурную схему и диаграмму состояний

Для управления лампочкой мы добавим выход в блок-схему нашего контроллера, на рисунке справа Y3 — это выход управления светом.



На диаграмме ниже добавлено еще одно состояние, «СВЕТ». Всякий раз, когда владелец гаража нажимает и отпускает кнопку управления дверью, активизируется состояние ПОДНЯТЬ либо ОПУСТИТЬ и одновременно активизируется состояние СВЕТ. Линия, ведущая к состоянию Свет, изображена штрихами, поскольку это второстепенный переход.

Состояние СВЕТ будем считать параллельным процессом по отношению к состояниям ПОДЪЕМ и ОПУСКАНИЕ. Переход к состоянию СВЕТ не является сменой состояния (Stage Jmp), а соответствует команде установки состояния (State Set). В релейной программе Стадии СВЕТ предусмотрен трехминутный таймер. Когда истекает время, устанавливается бит таймера T0, а Стадия СВЕТ сбрасывается. Стрелка, отходящая от Стадии СВЕТ, никуда не ведет, указывая на то, что Стадия СВЕТ просто становится неактивной, и свет выключается!



Используем таймер внутри Стадии

Законченная модифицированная программа показана на рисунке справа. Затененные фрагменты соответствуют дополнениям, внесенным в программу.

В Стадии S1 (Нажатие-ВЕРХ) была добавлена команда установки Бита Стадии S6. Когда контакт X0 размыкается, мы переходим от S1 к двум новым активным состояниям: S2 и S6. В состоянии S4 (Нажатие-НИЗ) мы произвели те же изменения, поэтому, всякий раз, когда кто-то нажимает кнопку управления дверью, включается лампочка.

Большинство тех, кто впервые создает стадийные программы, зададутся вопросом, куда разместить Стадию СВЕТ и какой номер ей присвоить. Это не имеет никакого значения!

Просто присвойте новой Стадии еще неиспользуемый номер.

Место размещения в программе не имеет значения, поэтому разместим команду в конце.

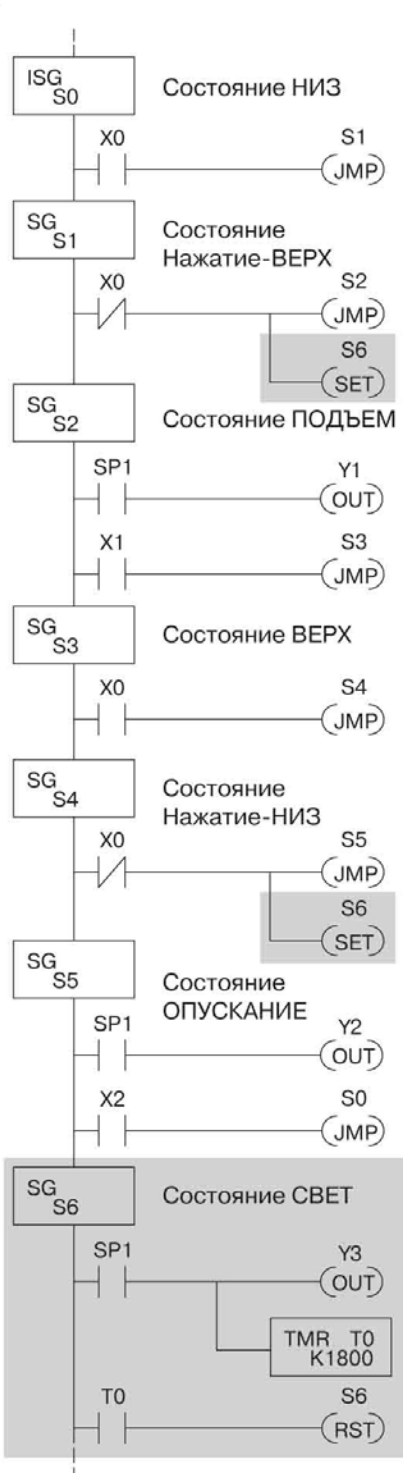
Возможно, вы считаете, что каждая Стадия должна обязательно находиться под Стадией, из которой происходит переход. Хотя это и распространено на практике, это совсем не обязательно (и это хорошо, поскольку в нашем случае это было бы невозможно из-за двух размещенных команд Set S6). Номера Стадий и то, как они используются, определяется линиями переходов.

В Стадии S6 мы включаем сигнальную лампу, активизируя выход Y3. Специальный релейный контакт SP1 всегда замкнут. Таймер T0 ведет отсчет с интервалом 0.1 с. Рассчитаем количество тактов, необходимое для достижения трех минут:

К = (3 мин * 60 сек/мин)/(0.1 сек/такт) = 1800 тактов.

Пока Стадия S6 остается активной, таймер продолжает отсчет времени. По завершении счета устанавливается соответствующий бит таймера T0. Таким образом, спустя три минуты устанавливается T0, и команда сброса S6 (Reset S6) делает Стадию неактивной.

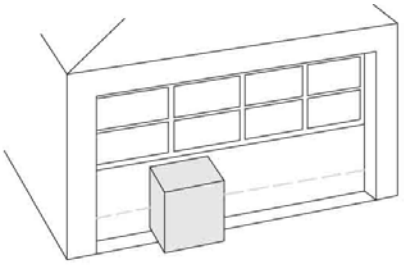
Пока Стадия S6 активна и лампочка включена, переключения состояний по основной линии переключений продолжаются обычным образом и независимо от Стадии 6. Другими словами, дверь может подниматься и опускаться, но лампочка будет включена ровно три минуты.



Добавим функцию аварийного останова

Некоторые современные устройства открывания гаражной двери способны обнаружить объект, находящийся под дверью. Опускающаяся дверь, снабженная, как правило, фотодатчиком ("электронным глазом"), остановится и начнет подниматься. Запрограммируем работу функции защиты именно таким образом, добавив в структурную схему вход для фотоэлемента, как показано на рисунке справа. Вход X3 будет установлен, если на пути к двери расположен какой-либо предмет.

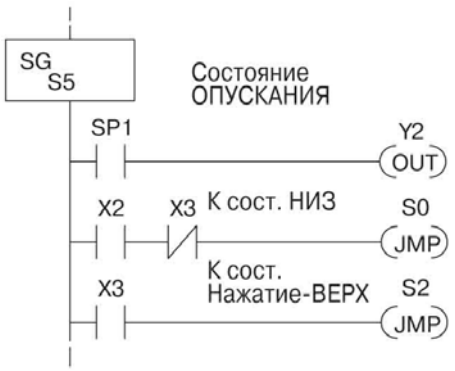
После этого сделаем простое дополнение в диаграмму переходов состояний, показанное затененными областями на рисунке ниже. Обратите внимание на новый путь перехода над состоянием ОПУСКАНИЕ. Если в процессе опускания двери будет обнаружено препятствие (X3), мы перейдем к состоянию Нажатие-ВЕРХ. Это происходит вместо перехода напрямую к состоянию ПОДЪЕМ, чтобы для сброса выхода Y2 был в запасе один цикл, прежде чем активизируется выход подъема Y1.



Введем взаимоисключающие переходы

Теоретически возможно, что нижний переключатель (X2) и вход обнаружения препятствия (X3) активизируются одновременно. В этом случае произойдет переход одновременно к состояниям Нажатие-ВЕРХ и НИЗ, что не имеет никакого смысла.

Чтобы избежать этого, наделим сигнал препятствия более высоким приоритетом, изменив условие перехода к состоянию НИЗ на [X2 И НЕ X3]. Благодаря этому обнаружение препятствия будет иметь более высокий приоритет. Изменения, которые мы должны сделать в Стадии ОПУСКАНИЕ (S5), показаны на рисунке справа. Первая цепь остается без изменений. Во второй и третей цепях мы реализуем требуемые переходы. Обратите внимание на инверсное использование контакта реле, которое обеспечит выполнение Стадией лишь одной из команд JMP.



Правила создания стадийных программ

Организация стадийной программы

До сих пор в данной главе приводились примеры, в которых использовалась автономная диаграмма состояний, представляющая основной процесс. Тем не менее, стадийное программирование позволяет реализовать несколько процессов одновременно в одной и той же программе. Программисты, только начавшие работать со стадийным программированием, ошибочно полагают, что только одна Стадия (состояние) может быть активной в определенный момент времени, и поэтому часто пытаются включать и отключать Стадию в каждом цикле. Если требуется, чтобы цепи релейной программы выполнялись в каждом цикле, их следует разместить в Стадии, которая всегда активна.
На следующем рисунке показана типичная прикладная задача. И основной производственный процесс "Главный процесс", и "Инициализация по включению питания", и "Система аварийного останова", и "Контроль аварийных состояний", и "Операторский интерфейс" в режиме работы активны одновременно. После включения питания начинается работа трех Начальных Стадий, показанных на рисунке.

В типичном случае работа отдельных последовательностей Стадий, показанных выше, протекает следующим образом:

- **Инициализация по включению питания** - данная Стадия содержит те ветви релейной программы, которые выполняются только один раз после включения питания. Ее последняя цепь приводит к сбросу Стадии, поэтому данная Стадия является активной только в течение одного цикла (либо только требуемое количество циклов).

- **Главный процесс** - эта последовательность Стадий управляет самой центральной частью процесса или установки. Будучи исполненной от начала до конца, она реализует один цикл работы установки, либо один цикл процесса.

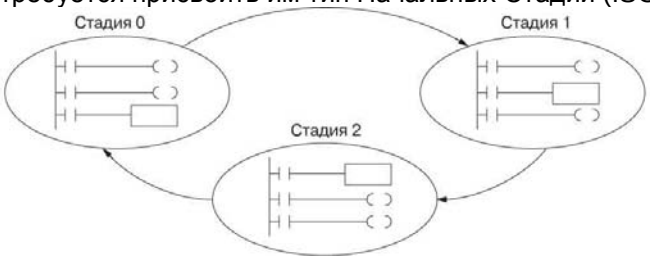
- **Аварийный останов и контроль аварийных состояний** - данная Стадия всегда активна, поскольку она следит за ошибками, которые являются признаками аварийных состояний или требуют аварийного останова. Как правило, данная Стадия приводит к сбросу Стадий главного процесса или где-либо еще с целью их инициализации по возникновению состояния ошибки.

- **Операторский интерфейс** - это еще одна задача, которая всегда должна оставаться активной и способной реагировать на действия оператора. Она позволяет оператору изменять режимы работы и выполнять другие задачи независимо от текущей Стадии, выполняемой в рамках главного процесса.

Хотя мы и имеем дело с отдельными процессами, между ними можно достичь взаимодействия. Например, в случае возникновения ошибки, в Стадии "Состояние" может потребоваться автоматическое переключение операторского интерфейса в режим "Состояние", в котором будет отображена информация о возникшей ошибке (см. рисунок справа). Стадия "Контроль" могла бы активизировать Стадию "Состояние" и сбросить Стадии "Управление" и "Рецепты".

Работа команд внутри Стадий

Можно считать, что выделение состояний или Стадий сводится к простому разбиению релейной программы, как показано на рисунке ниже. Каждая Стадия содержит только те цепи релейной программы, которые требуются соответствующему состоянию процесса. Условия перехода от определенной Стадии к другой содержатся в самой этой Стадии. Не сложно выбрать, какие цепи релейной программы будут активными после включения питания. Для этого требуется присвоить им тип Начальных Стадий (ISG).



Большинство команд работает так же, как и в стандартной RLL-программе. Стадию можно считать всего лишь миниатюрной RLL-программой, которая в любой момент времени либо активна, либо не активна.

Выходные обмотки - Как и раньше, выходные обмотки (Coil) в активных Стадиях приводят к включению или отключению выходов в зависимости от протекания тока по обмотке. В то же время, отметим следующее:

- Выходы работают как обычно при том условии, что имя каждого выхода (например, "Y3") используется лишь в одной Стадии.
- Выход может управляться из нескольких Стадий при условии, что только одна из этих Стадий является активной в каждый момент времени.
- Если выходная обмотка реле управляется несколькими Стадиями одновременно, конечное состояние выхода в пределах каждого цикла определяется активной Стадией, расположенной ближе всего к концу программы. Поэтому в том случае, когда для управления выходом требуется формировать логическое ИЛИ из нескольких Стадий, следует использовать команду OROUT.

Обмотки одиночного импульса (PD) - Будьте внимательны при использовании в Стадиях обмотки реле одиночного импульса (PD). Помните, что на входе обмотки должен произойти переход из 0 в 1. Если ток через катушку уже протекает в первом цикле, когда Стадия становится активной, обмотка одиночного импульса не сработает, поскольку переход из 0 в 1 уже произошел.
Альтернатива обмотке одиночного импульса: если имеется задача, которую требуется выполнить лишь один раз (в одном цикле), ее можно разместить в той Стадии, переход от которой к следующей Стадии осуществляется в том же цикле.

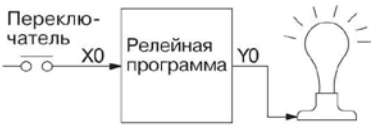
Счетчик - При использовании в пределах Стадии счетчика Стадия должна быть активной в течение одного цикла до того, как на входе счетчика произойдет переход из 0 в 1. В противном случае фактического перехода не произойдет, и счет осуществляться не будет.
На обычный счетчик, использующийся внутри Стадии, накладываются ограничения: он может не сброситься из других Стадий с помощью команды RST для выходного бита счетчика. Эта проблема, впрочем, устраняется за счет применения специального счетчика.

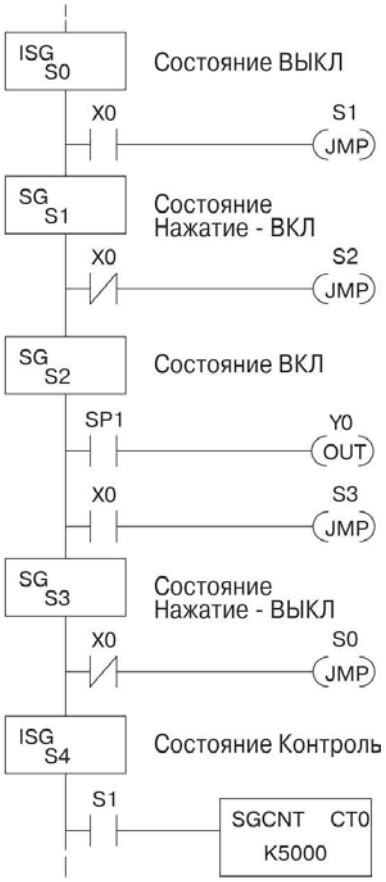
Специальный счетчик - Преимуществом специального счетчика является то, что он может быть сброшен из других Стадий с помощью команды RST. Он имеет счетный вход, но не имеет входа сброса. Это его единственное отличие от стандартного счетчика.

Барабанный командоаппарат (Drum) - Следует понимать, что барабанный командоаппарат является самостоятельным процессом и его программирование отличается от стадийного программирования. Если требуется использовать стадийную программу с барабанным командоаппаратом, следует разместить команду барабанного командоаппарата в Стадии ISG, которая всегда активна.

Использование Стадии в качестве контролирующего процесса

Вспомните пример контроллера включения/выключения лампочки из предыдущей главы. Предположим, к примеру, что нам требуется контролировать "Производительность" процесса управления лампой, подсчитывая количество циклов включения/выключения. Эта задача потребует внесения простого счетчика, но главное - принять решение, где разместить счетчик.





Те, кто только начал изучать стадийное программирование, как правило, пытаются разместить счетчик внутри одной из Стадий процесса, которую они предполагают контролировать. Проблема состоит в том, что Стадия является активной только некоторую часть времени. Чтобы счетчик мог осуществить счет, сигнал на счетном входе должен перейти из "0" в "1" хотя бы один цикл спустя после активизации Стадии, в которой находится счетчик. Чтобы это произошло, требуется дополнительное программирование, которое может быть достаточно сложным.

В этом случае нам всего лишь требуется добавить дополнительную Стадию Контроль, как показано на рисунке выше, чтобы "присматривать" за главным процессом. Счетчик в пределах Стадии Контроль использует в качестве входа бит состояния S1 главного процесса. **Биты состояния, используемые в качестве контакта, позволяют нам контролировать процесс!**

Обратите внимание, что и Стадия Контроль, и Стадия ВЫКЛ являются Начальными Стадиями. Стадия Контроль остается активной все время.

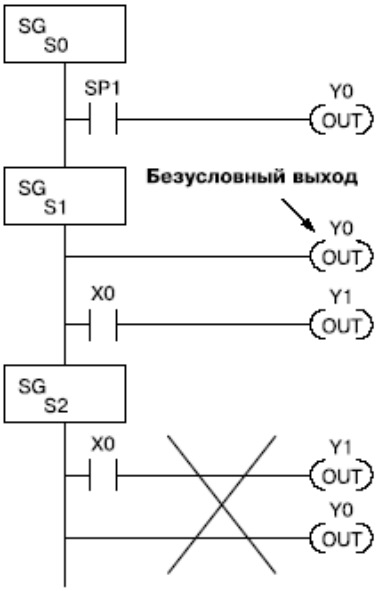
Счетчик для стадийной программы

Примененный в примере выше счетчик является специальным Счетчиком для стадийной программы. Обратите внимание, что в нем не предусмотрен вход сброса. Счетчик сбрасывается по команде сброса (Reset), в которой указывается бит счетчика (в нашем случае CT0). Преимуществом счетчика для стадийной программы является то, что он может быть сброшен глобально по всей программе из других Стадий. Стандартный счетчик не обладает возможностью глобального сброса. Ну и конечно, вы можете по-прежнему использовать обычный счетчик в пределах Стадии, но единственным средством его сброса является вход сброса.

Безусловные выходы

Вашему приложению может потребоваться, чтобы конкретный выход находился в состоянии ВКЛ независимо от активности конкретной Стадии. До сих пор во всех примерах последовательно с выходными обмотками использовался специальный контакт реле SP1 (постоянно замкнутый).

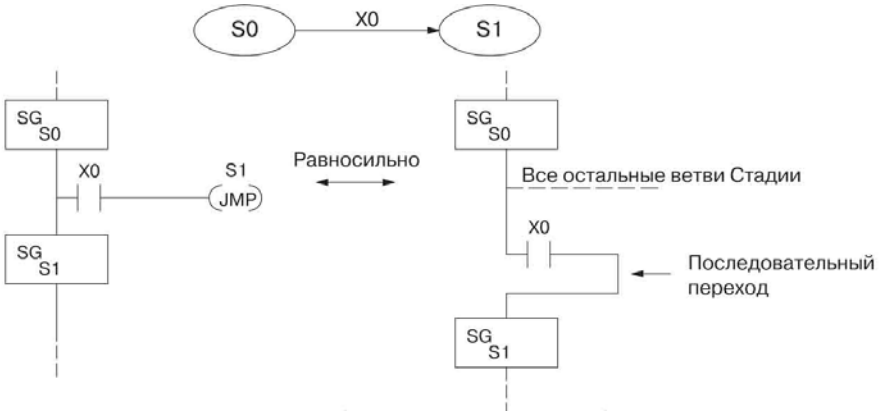
Можно обойтись без использования такого контакта, если все безусловные выходы поместить в начале (наверху) раздела Стадии программы (см. на первую цепь Стадии 1).



ПРЕДУПРЕЖДЕНИЕ: *Безусловные выходы, помещенные в каком-либо другом месте Стадии, необязательно будут включены во время активности стадии. Справа Y0 в Стадии 2 изображен как безусловный выход, однако, он запитывается от вышерасположенной цепи. Таким образом, статус Y0 совпадет со статусом Y1 (что неправильно).*

Переключение состояний методом последовательного перехода

Рассматривая переходы состояний, мы видели, как команда Stage JMP (Переход к Стадии) сбрасывает текущую Стадию и делает активной следующую Стадию (указанную в команде JMP). В пакете *Direct*SOFT для переключения состояний также можно использовать метод последовательного перехода. Главное требование состоит в том, чтобы текущая Стадия располагалась в релейной программе непосредственно над следующей Стадией (к которой происходит переход). Такое расположение Стадий показано на рисунке снизу (Стадии S0 и S1 соответственно).



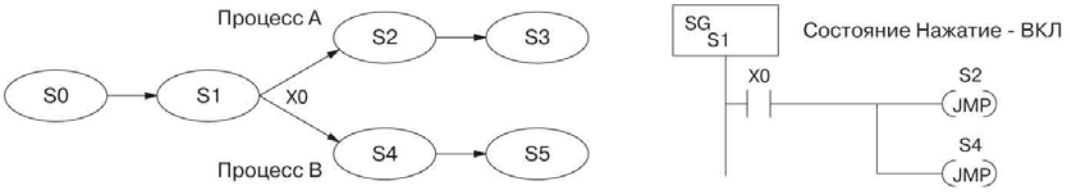
Вспомним, что команда Stage JMP может быть расположена в любом месте текущей Стадии, но результат будет один и тот же. В то же время, переход состояния методом последовательного перехода (показанный выше), должен быть реализован в последней цепи Стадии. Все остальные цепи Стадии будут ему предшествовать. Переход состояния методом последовательного перехода можно также реализовать с помощью Ручного Программатора, разместив после условия перехода команду Stage (Стадия) для следующей Стадии.

При использовании метода последовательного перехода остается только одна команда Stage JMP, и в этом преимущество данного метода. В то же время, вносить в программу изменения сложнее, чем при использовании Stage JMP, поэтому большинству программистов для реализации переходов состояний рекомендуется использовать команду Stage JMP.

Принципы параллельного выполнения процессов

Параллельные процессы

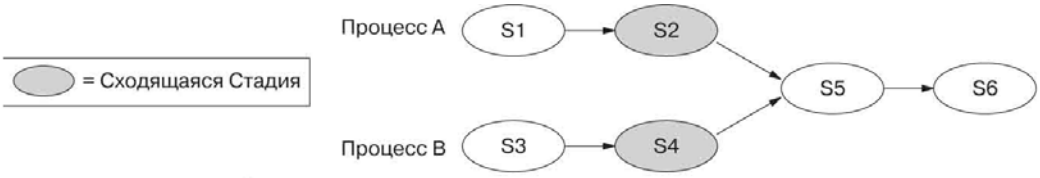
В предыдущих разделах данной главы мы рассматривали ситуацию, когда переход из текущего состояния осуществлялся либо к одному, либо к другому состоянию, называя такие переходы **взаимоисключающими.** В ряде случаев может потребоваться разветвление на два или более параллельных процесса, которые должны выполняться одновременно, как показано на рисунке ниже. Для каждого ответвления можно использовать команду JMP, как показано на рисунке, либо можно использовать одну команду JMP и команду (команды) установки Бита Включения Стадии (Set Stage bit) (должна быть хотя бы одна команда JMP, чтобы выйти из S1). Следует помнить, что при переходе от Стадии все команды этой Стадии будут выполнены (команда JMP не эквивалентна GOTO).



Заметьте, что если нам требуется, чтобы Стадии S2 и S4 активизировались в одном и том же цикле, обе этих Стадии должны располагаться под или над Стадией 1 в релейной программе (смотрите пояснение внизу страницы). Как бы там ни было, разветвление процессов осуществляется очень просто!

Сходящиеся процессы

Теперь рассмотрим противоположный случай, когда параллельно протекающие процессы сходятся в одну точку. Такая ситуация имеет место, когда мы прекращаем выполнять несколько действий одновременно и переходим к выполнению чего-то одного. На рисунке, показанном ниже, процессы A и B сходятся, когда Стадии S2 и S4 обе переходят к Стадии S5 в некоторый момент времени. Другими словами, S2 и S4 - это **Сходящиеся Стадии**.

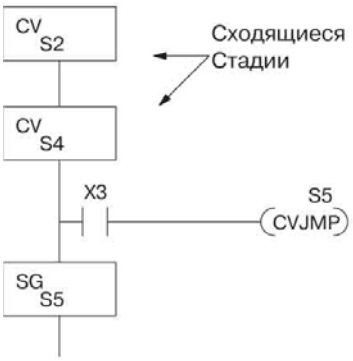


Сходящиеся Стадии (CV)

Хотя сам по себе принцип схождения процессов достаточно прост, с ним связано одно усложнение. Процессы, выполнявшиеся одновременно, практически никогда не завершаются в одно и то же время. Другими словами, мы не можем точно знать, какой из процессов завершился первым, Стадия S2 или S4. Это очень важно, поскольку нам требуется принять решение, каким образом перейти к Стадии S5.

Решение состоит в синхронизации условий перехода, предусмотренных в Сходящихся Стадиях. Это осуществляется с помощью Стадии специального типа, предусмотренной для этих целей: Сходящейся Стадии (тип CV).

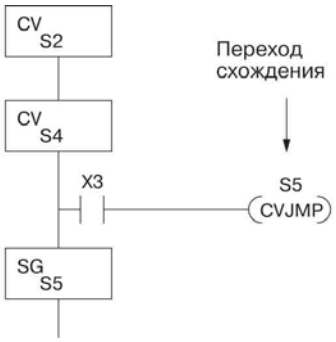
Справа показан пример, в котором S2 и S4 должны быть сгруппированы вместе, как показано на рисунке. **Размещение логических элементов между Стадиями CV не допускается!** Условие перехода (в нашем случае X3) должно размещаться в последней Сходящейся Стадии. Условия перехода активизируются только тогда, когда становятся активными все Сходящиеся Стадии в группе.



Переход схождения (CVJMP)

✗	✓	✓	✓
230	240	250–1	260

Вспомним, что последняя Сходящаяся Стадия становится активной только тогда, когда активны все Стадии CV в группе. Чтобы завершить Сходящуюся Стадию, нам требуется новая команда перехода. Показанная на рисунке справа команда Перехода схождения (Convergence Jump = CVJMP) приведет к переходу к Стадии S5, когда установится условие X3 (как Вы, должно быть, и предполагали), **но она также автоматически сбрасывает все Сходящиеся Стадии в группе.** Таким образом, команда перехода CVJMP является очень эффективной командой. Заметим, что данная команда может использоваться только для Сходящихся Стадий.



Указания по реализации Сходящихся Стадий

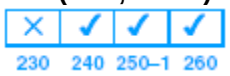
Ниже сведены требования к использованию Сходящихся Стадий, включая некоторые советы по их более эффективному применению:

- Сходящаяся Стадия должна использоваться как последняя Стадия процесса, которая протекает параллельно с другим процессом или процессами. Переход к Сходящейся Стадии означает, что определенный процесс пройден, и является точкой ожидания до тех пор, пока все остальные процессы также не будут завершены.
- Максимальное число сходящихся стадий, образующих группу, равно 17. Другими словами, в одну стадию может сойтись не более 17 стадий.
- Сходящиеся Стадии одной и той же группы следует размещать в программе вместе, подключенными к шине без каких-либо других логических элементов между ними.
- В пределах группы Сходящиеся Стадии могут вводиться в любом порядке, сверху вниз. Не имеет значения, какая именно Стадия является в группе последней, поскольку до того, как последняя Стадия станет активной, все Сходящиеся Стадии также должны стать активными.
- Последняя Сходящаяся Стадия в группе может содержать элементы релейной программы. Эта релейная программа, впрочем, не будет выполняться до тех пор, пока все Сходящиеся Стадии в группе не станут активными.
- Переход схождения (CVJMP) - это специальная команда, которая используется для перехода от группы Сходящихся Стадий к следующей Стадии. CVJMP приводит к сбросу всех Сходящихся Стадий группы и активизирует Стадию, указанную в команде перехода.
- Команду CVJMP можно использовать только в Сходящейся Стадии, и она не работает в обычных или начальных Стадиях.
- Сходящиеся Стадии или команды CVJMP не должны применяться в подпрограммах или подпрограммах обработки прерываний.

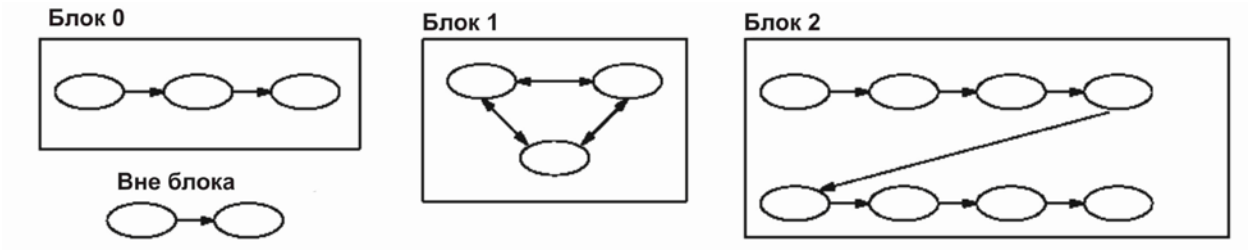
Управление большими программами

Стадия может содержать множество программных цепей, либо всего лишь одну или две цепи. Для большинства приложений хорошее проектирование программы гарантирует, что среднее число цепей на Стадию будет небольшим. Однако большие прикладные программы по-прежнему используют огромное число Стадий. Мы вводим новый конструктивный элемент, который поможет нам организовать взаимосвязанные стадии в группы, называемые **блоками.** Итак, основным преимуществом использования блоков стадий является организация программы.

Блок Стадий (BLK, BEND)



Блок — это раздел релейной программы, который содержит Стадии. На следующем рисунке каждый блок имеет собственный номер. Как и Стадия, Блок Стадий может быть активным или неактивным. Ограничения на способы перехода внутри блока от Стадии к Стадии отсутствуют. Заметим, что использование Блоков Стадий не требует, чтобы каждая Стадия размещалась внутри какого-либо Блока, что показано на рисунке с помощью «Стадий вне блоков».



Программа с 20 и более Стадиями может рассматриваться как достаточно большая, чтобы использовать группирование в блоки (однако их использование не является обязательным). Рекомендуется по возможности использовать два и более Блоков Стадий, поскольку использование одного Блока дает слишком незначительное преимущество.

Блок стадий отделяется от остальной программы специальными начальной и завершающей командами. На рисунке справа команда BLK вверху обозначает начало блока стадий. Внизу команда BEND обозначает конец блока. Стадии между этими граничными метками (SO и S1 в данном случае) и все связанные с ними цепи образуют блок.

Заметим, что команда блока включает поле ссылки (в примере установлено в C0). Команда Блока заимствует или использует номер контакта управляющего реле, так что другие части программы могут управлять Блоком. **Любой номер управляющего реле (такой как C0), используемый в какой-либо команде BLK, недоступен для использования в качестве управляющего реле.**

Обратите внимание на то, что внутри блока могут использоваться только обычные стадии (SG) или сходящиеся (CV). Они не могут быть начальными стадиями. Нумерация Стадий внутри блока может производиться в любом порядке и совершенно не зависит от нумерации Блоков.

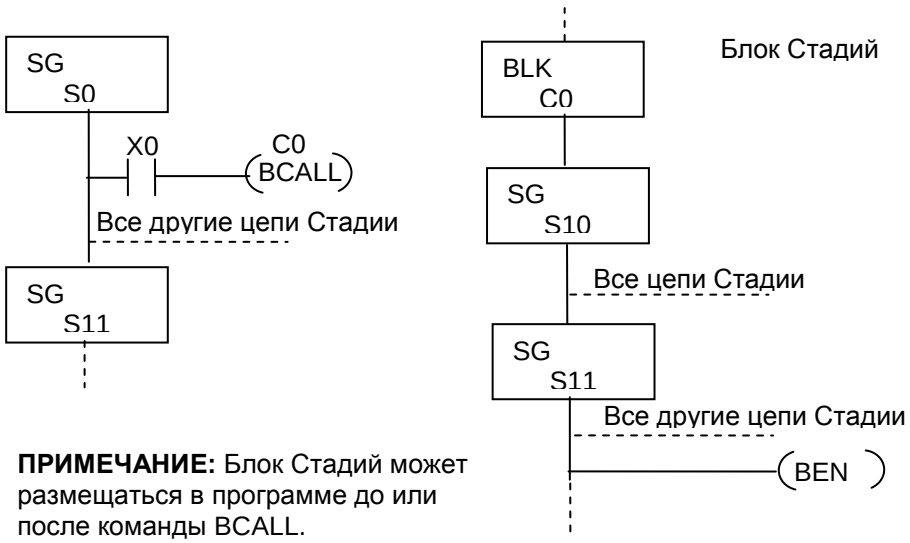
Вызов Блока (BCALL)

Команды Блока Стадий позволяют активизировать одновременно группу Стадий. При включении питания или после переходов из режима Программирования в Рабочий режим все Блоки Стадий и их Стадии неактивны. Как показано на следующем рисунке, команда Вызов Блока имеет тип выходной обмотки. Когда контакт X0 замыкается, BCALL активизирует Блок Стадий, на который ссылается команда (C0). Когда BCALL отключается, соответствующий Блок Стадий и стадии внутри него становятся неактивными.

Необходимо избежать путаницы между работой Вызова Блока и выполнением «Вызова Подпрограммы». После срабатывания обмотки BCALL выполнение программы продолжается со следующей цепи программы. Когда же выполнение программы достигает места, где располагается упомянутый в BCALL Блок Стадий, то выполняется логика внутри Блока, поскольку он становится активным. Так же нельзя классифицировать BCALL как тип перехода состояния (это не JMP).

Когда Блок Стадий становится активным, автоматически на том же сканировании становится активной первая Стадия блока. «Первая» Стадия блока — это Стадия, расположенная в программе непосредственно за командой блока (BLK). Так что роль этой Стадии аналогична роли начальной Стадии, рассмотренной ранее.

Команда Вызов Блока может использоваться в нескольких контекстах. Очевидно, что первое выполнение BCALL должно происходить снаружи Блока Стадий, поскольку Блоки Стадий изначально неактивны. Кроме того, как показано ниже, BCALL может вызываться в обычной цепи программы или внутри активной Стадии. Заметим, что выключение BCALL либо Стадии, содержащей BCALL, дезактивирует соответствующий блок Стадий. С помощью BCALL можно также управлять Блоком Стадий из другого Блока Стадий.



ПРИМЕЧАНИЕ: Блок Стадий может размещаться в программе до или после команды BCALL.

Команда BCALL может быть использована многими способами и в различных контекстах, так что найти для нее наилучшее применение может оказаться не просто. Запомните, что цель Блоков Стадий состоит в том, чтобы помочь организовать задачу приложения, группируя взаимосвязанные Стадии. Не забывайте, что Начальные Стадии должны располагаться вне Блоков Стадий.

Команды языка RLL^{PLUS}

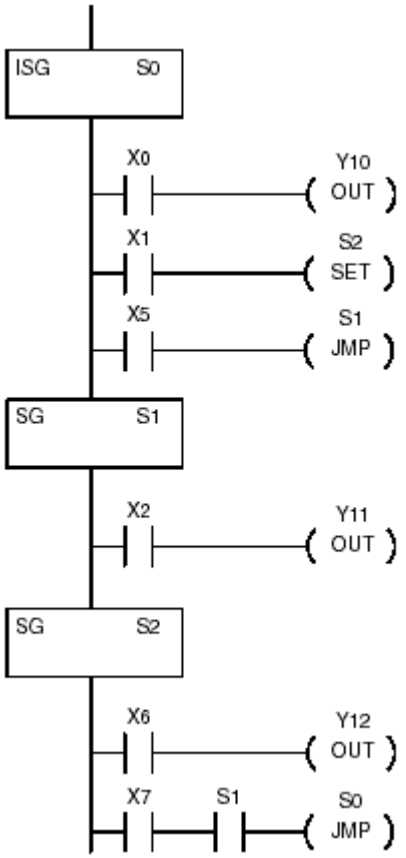
Стадия (SG)

Команды **Stage** (Стадия) используется для создания структурированных программ на языке RLL^{PLUS}. Стадии являются сегментами программы, которые становятся активными согласно запрограммированной логике переходов состояний, а также по командам перехода или установки Стадии, которые исполняются в текущей активной Стадии. Стадии перестают быть активными один цикл спустя после выполнения логического условия перехода, после выполнения команды перехода или команды сброса Стадии.

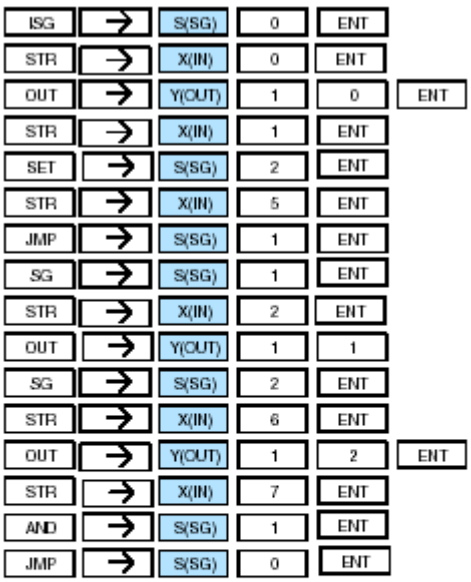
Тип данных операнда	Диапазон для DL230	Диапазон для DL240	Диапазон для DL250-1	Диапазон для DL260
	aaa	aaa	aaa	aaa
Стадия S	0-377	0-777	0-1777	0-1777

Далее приводится пример простой программы на языке RLL^{PLUS}. Для создания структурированной программы применены следующие команды (элементы): Начальная Стадия, Стадия и Переход.

Окно *Direct*SOFT Программаторе



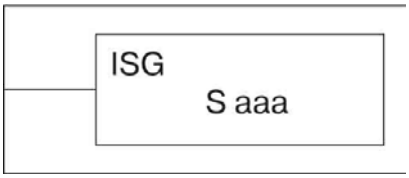
Набор клавиш на Ручном



Начальная Стадия (ISG)

✓ 230 ✓ 240 ✓ 250–1 ✓ 260

Команда **Initial Stage** (Начальная Стадия) обычно используется в качестве первого сегмента программы RLL^{PLUS}. Допускается включать в программу несколько Начальных Стадий. Они становятся активными, когда ЦПУ переходит в режим исполнения (RUN), и определяют точку начала исполнения программы.

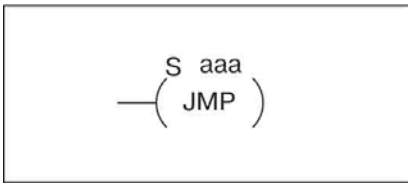


Тип данных операнда	Диапазон для DL230	Диапазон для DL240	Диапазон для DL250-1	Диапазон для DL260
	aaa	aaa	aaa	aaa
Стадия S	0-377	0-777	0-1777	0-1777

Переход (JMP)

✓ 230 ✓ 240 ✓ 250–1 ✓ 260

Команда Jump (Переход) позволяет программе перейти из текущей активной Стадии, в которой находится команда перехода, к другой Стадии (указанной в команде). Переход осуществляется, когда выполняется входное логическое условие. Активная Стадия, содержащая команду перехода, перестанет быть активной спустя один цикл.

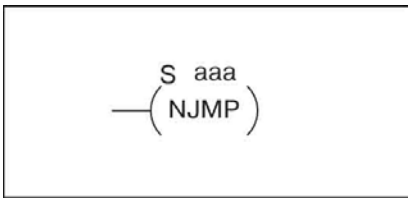


Тип данных операнда	Диапазон для DL230	Диапазон для DL240	Диапазон для DL250-1	Диапазон для DL260
	aaa	aaa	aaa	aaa
Стадия S	0-377	0-777	0-1777	0-1777

Переход по НЕ (NJMP)

✓ 230 ✓ 240 ✓ 250–1 ✓ 260

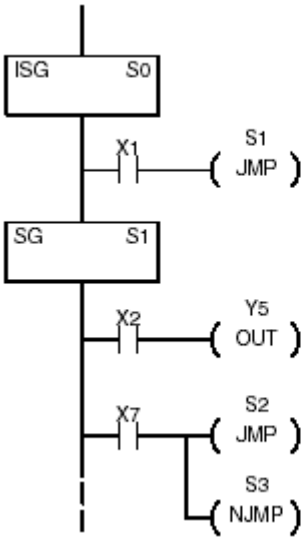
Команда **Not Jump** (Переход по НЕ) позволяет программе перейти из текущей активной Стадии, в которой находится команда перехода, к другой Стадии (указанной в команде). Переход происходит тогда, когда входное логическое условие не выполняется (находится в состоянии "0"). Активная Стадия, содержащая команду Not Jump, становится неактивной один цикл спустя после исполнения команды Not Jump.



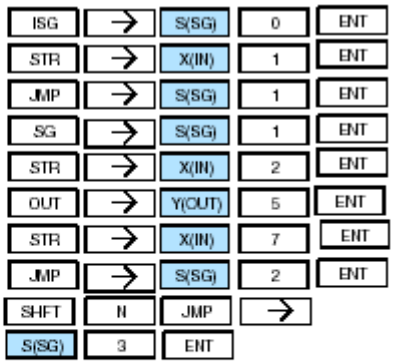
Тип данных операнда	Диапазон для DL230	Диапазон для DL240	Диапазон для DL250-1	Диапазон для DL260
	aaa	aaa	aaa	aaa
Стадия S	0-377	0-777	0-1777	0-1777

В следующем примере вначале выполнения программы будет активной только Стадия ISG0. Когда выполнится условие X1, программа перейдет от Начальной Стадии "0" к Стадии "1". В «Стадии 1», если X2 включен, будет включен выход Y5. Если включен X7, программа переходит из «Стадии 1» в «Стадию 2». Если же X7 выключен, программа переходит из «Стадии 1» в «Стадию 3».

Окно *Direct*SOFT Программаторе



Набор клавиш на Ручном

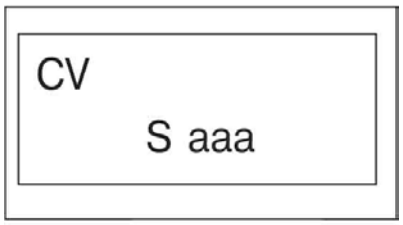


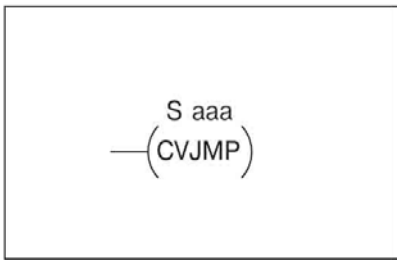
Сходящаяся Стадия (CV) и Переход схождения (CVJMP)

Команда **Converge Stage** (Сходящаяся Стадия) используется с целью сведения определенных Стадий к одной Стадии.

Когда все Сходящиеся Стадии в пределах группы становятся активными, происходит исполнение команды CVJMP (и всех дополнительных логических инструкций, предусмотренных в конечной Стадии CV). Прежде чем логическая программа конечной Стадии CV будет исполнена, должны стать активными все предшествующие Стадии CV. Все Сходящиеся Стадии становятся неактивными один цикл спустя после исполнения команды CVJMP.

Дополнительные логические команды допускается вставлять только между последней Сходящейся Стадией и командой CVJMP. Включать несколько команд CVJMP не допускается.



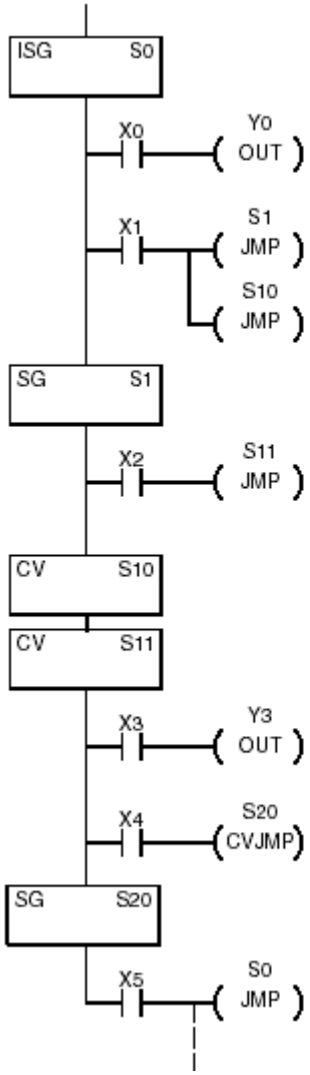


Сходящиеся Стадии должны программироваться в основной части прикладной программы. Другими словами, Сходящиеся Стадии нельзя создавать в пределах подпрограмм или подпрограмм обработки прерываний.

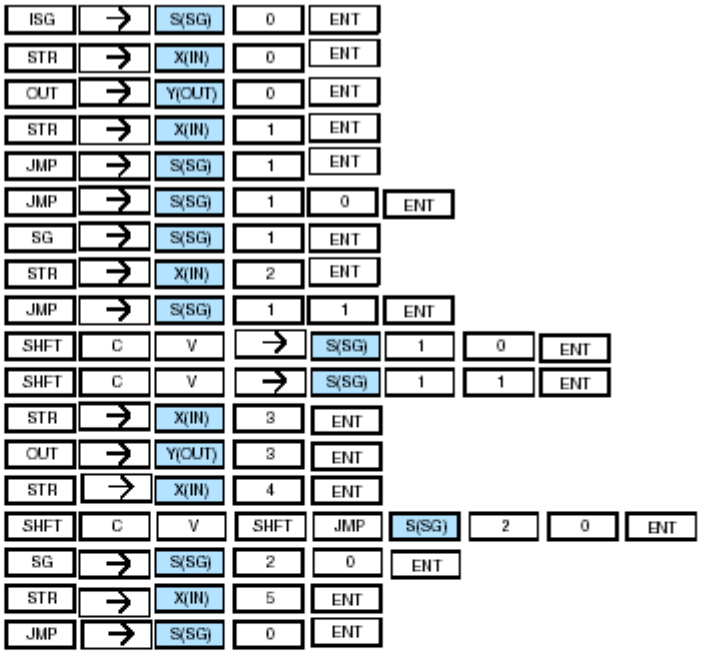
Тип данных операнда		Диапазон для DL240	Диапазон для DL250-1	Диапазон для DL260
		aaa	aaa	aaa
Стадия	S	0-777	0-1777	0-1777

Ниже приводится пример, в котором команда CVJMP будет исполнена по исполнению условия X4 после того, как обе Сходящиеся Стадии S10 и S11 станут активными. CVJMP отключит Стадии S10 и S11 и активизирует S20. Далее, когда исполнится условие X5, программа перейдет вновь к Начальной Стадии S0.

Окно *Direct*SOFT



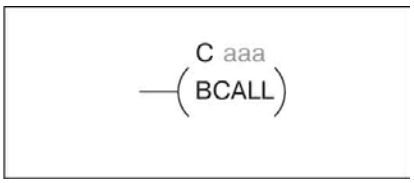
Набор клавиш на Ручном Программаторе



Вызов Блока (BCALL)

230	240	250–1	260
✗	✓	✓	✓

Команды блока стадий позволяют активизировать одновременно группу Стадий. Команды **Block Call** (Вызов Блока), Block (Блок) и Block End (Завершение Блока) должны применяться вместе. Чтобы активизировать блок Стадий, используется команда BCALL.



О команде BCALL необходимо знать следующее:

В ней используются номера реле C - BCALL вводится в программу как выходная обмотка, но на номер Стадии она не ссылается, как можно было бы предположить. Вместо этого блок идентифицируется как управляющее реле (Caaa). Это управляющее реле нельзя использовать в качестве выхода в любом другом месте программы.

Она должна оставаться активной - Команда BCALL фактически управляет всеми Стадиями, расположенными между командами BLK и BEND, даже после того, как Стадии, расположенные внутри блока, начали выполняться. BCALL должна оставаться активной, или все Стадии в блоке автоматически выключатся. Если либо команда BCALL, либо Стадия, содержащая команду BCALL, выключаются, Стадии в указанном блоке также автоматически выключаются.

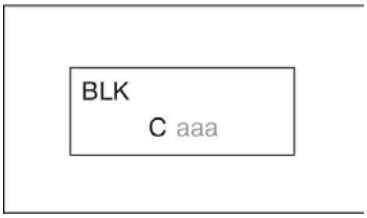
Она активизирует первую Стадию Блока - При исполнении команды BCALL автоматически активизируется первая Стадия, следующая за командой BLK.

Тип данных операнда	Диапазон для DL240	Диапазон для DL250-1	Диапазон для DL260
	aaa	aaa	aaa
Управляющее реле C	0-777	0-1777	0-3777

Блок (BLK)

230	240	250–1	260
✗	✓	✓	✓

Команда **Block** (Блок) является признаком начала Блока Стадий, которые могут активизироваться одновременно, как единая группа. Команда Stage (Стадия) должна размещаться сразу же за командой Start Block (Начало Блока). Внутри блока не допускается использовать Начальные Стадии. Реле управления (Caaa), указанное в команде Block (Блок), не должно использоваться в качестве выхода где-либо еще в программе.

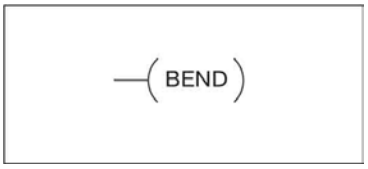


Тип данных операнда	Диапазон для DL240	Диапазон для DL250-1	Диапазон для DL260
	aaa	aaa	aaa
Управляющее реле C	0-777	0-1777	0-3777

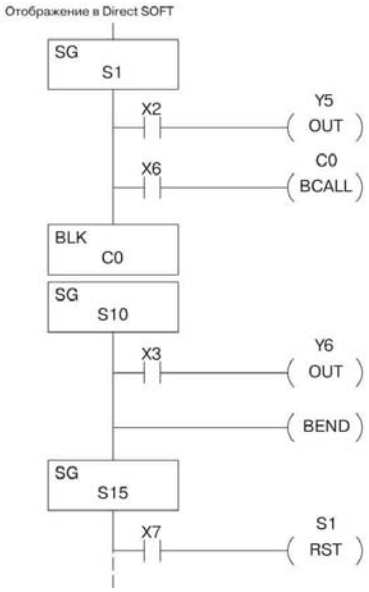
Завершение блока (BEND)

230	240	250–1	260
✗	✓	✓	✓

Команда **Block End** (Завершение Блока) является признаком завершения Блока Стадий. Эта команда не содержит операнда. Совместно с одной командой Block Call (Вызов Блока) используется только одна команда Block End (Завершение Блока).



В нашем примере команда Block Call (Вызов Блока) выполняется, когда активна Стадия 1 и выполнено условие X6. После этого команда Block Call (Вызов Блока) автоматически активизирует Стадию S10, которая следует сразу же за командой Block (Блок).
Благодаря этому Стадии, расположенные между S10 и командой Block End (Завершение Блока), могут быть выполнены в соответствии с программой. Если команда BCALL сбрасывается, либо деактивизируется Стадия, содержащая команду BCALL, все Стадии между командами BLK и BEND также автоматически сбрасываются.
Изучив S15, можно увидеть, что X7 мог бы сбросить Стадию S1, которая приведет к отключению BCALL и, таким образом, сбросу всех Стадий внутри блока.

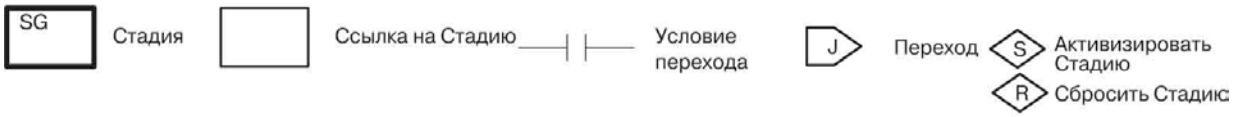


Последовательность нажатия клавиш ручного программатора

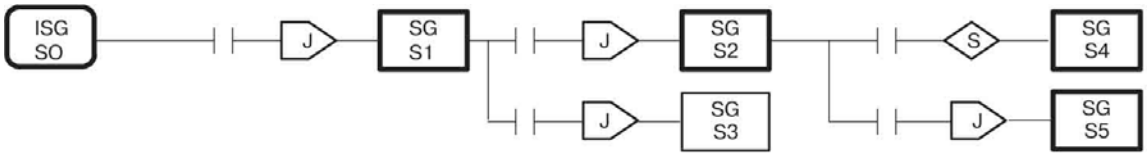
SG	→	S(SG)	1	ENT					
STR	→	X(IN)	2	ENT					
OUT	→	Y(OUT)	5	ENT					
STR	→	X(IN)	6	ENT					
SHFT	B	C	A	L	L	→	C(CR)	0	ENT
SHFT	B	L	K	→	C(CR)	0	ENT		
SG	→	S(SG)	1	0	ENT				
STR	→	X(IN)	3	ENT					
OUT	→	Y(OUT)	6	ENT					
SHFT	B	E	N	D	ENT				
SG	→	S(SG)	1	5	ENT				
STR	→	X(IN)	7	ENT					
RST	→	S(SG)	1	ENT					

Вид стадийной программы в пакете *Direct*SOFT

Функция **Stage View** (Вид стадийной программы) в *Direct*SOFT позволяет отобразить релейную программу в виде последовательной блок-схемы. На рисунке ниже показаны символьные обозначения, используемые в блок-схемах. Представление стадийной программы в виде последовательной блок-схемы может оказаться полезным, когда требуется убедиться в том, что стадийная программа в точности соответствует логике диаграммы переходов состояний, которую требуется реализовать.



На следующем рисунке показано типичное представление релейной программы, содержащей Стадии, в виде последовательной блок-схемы. Обратите внимание, что чтение схемы осуществляется слева на право.



Стадийное программирование в вопросах и ответах

Ниже приводятся вопросы по стадийному программированию, задаваемые наиболее часто, а также ответы на них. Это может быть полезным для тех, кто только начал изучать стадийное программирование. Темы, затрагиваемые в вопросах, были раскрыты подробно в данной главе.

Что позволяет делать стадийное программирование, чего нельзя достичь с помощью обычных программ на языках RLL?

Введение Стадий позволяет идентифицировать все состояния процесса еще до начала программирования. Это более организованный подход, поскольку вся релейной программе разделяется на отдельные участки. Являясь Стадиями процесса, эти фрагменты программы активны только тогда, когда они действительно требуются для управления процессом. Большинство процессов можно представить в виде последовательности Стадий (состояний), объединенных линиями переходов, происходящих по определенным событиям.

Разве Стадия на самом деле не аналогична подпрограмме?

Нет, имеются существенные различия. Подпрограмма вызывается основной программой при необходимости и выполняется только один раз прежде, чем вернется в точку, откуда она была вызвана. Стадия — это часть основной программы. Она представляет состояние процесса, и активная Стадия выполняется в каждом цикле процессора, пока не станет неактивной.

Что такое Биты Включения Стадий?

Бит Стадии - это отдельный бит в регистре памяти ЦПУ, соответствующий состоянию Стадии в реальном времени (активен/неактивен). Например, Бит Стадии "0" обозначается "S0". Если S0 = "0", все цепи релейной программы Стадии 0 пропускаются (не выполняются) в каждом цикле ЦПУ. Если S0 = "1", цепи релейной программы Стадии 0 выполняются в каждом цикле ЦПУ. При использовании Битов Включения Стадий в качестве контактов отдельные части программы могут контролировать друг друга, определяя активные/неактивные состояния Стадий.

Каким образом Стадия становится активной?

Существует три способа:
Если Стадия является Начальной (ISG), она становится активной автоматически после включения питания.
Другая Стадия может выполнить команду Stage JMP (Переход к Стадии), в которой указана активизируемая Стадия, в результате чего Стадия становится активной следующий раз, когда она встречается в программе.

- Цепь программы может выполнить команду Set Stage Bit (Установить Бит Включения).

Каким образом Стадия становится неактивной?

Существует три способа:
Стадии Рабочего режима (SG) после включения питания автоматически неактивны.
Стадия может выполнить команду Stage JMP (Переход к Стадии), сбросив Бит Стадии в "0".

- Любая цепь программы может выполнить команду Reset Stage Bit (Сбросить Бит Включения)

Что представляет собой метод последовательного перехода?

Метод последовательного перехода, используемый для объединения смежных Стадий (расположенных непосредственно друг над другом в программе) фактически ничем не отличается от команды Stage Jump (Переход к Стадии), исполняемой в верхней Стадии для Стадии, расположенной ниже. Переходы, реализованные методом последовательного перехода, в пакете *Direct*SOFT редактировать гораздо сложнее, и мы отделили их от двух предыдущих вопросов.

Можно ли сделать так, чтобы Стадия был активной только в течение одного цикла?

Да, но определяется это не в самом обозначении Стадии. Вместо этого в последней цепи такой Стадии следует разместить команду Stage JMP (Переход к Стадии), в результате чего цепь релейной программы будет активной только один цикл. Далее последует единственный цикл исполнения этого участка релейной программы, после чего произойдет переход к новой Стадии.

Не эквивалентна ли команда Stage JMP (Переход к Стадии) обычной команде GOTO, используемой в программировании?

Нет, эти команды совершенно различны. Команда GOTO перенаправляет выполнение программы сразу же в точку, указанную в команде GOTO. Команда Stage JMP всего лишь сбрасывает Бит Включения Стадии для текущей Стадии и одновременно устанавливает Бит Включения Стадии для Стадии, указанной в команде JMP. Биты Включения Стадий находятся в состоянии либо "0", либо "1", определяя активность/неактивность соответствующих Стадий. Выполнение команды Stage JMP (Переход к Стадии) приводит к следующим результатам:
После выполнения JMP оставшиеся цепи Стадии продолжают выполняться, даже если находятся после команды JMP. В следующем цикле эта Стадия не выполняется, поскольку является неактивной.

- Стадия, указанная в команде Stage JMP (Переход к Стадии), будет выполнена в следующий раз, когда она встретится в программе. Если она размещена за текущей Стадией, она будет выполнена в этом же цикле. Если она размещена до него (выше), она будет выполнена в следующем цикле.

Как определить, какую команду использовать, Переход к Стадии (Stage JMP), Установка Бита Стадии (Set Stage Bit) или Сброс Бита Стадии (Reset Stage Bit)?

Эти команды используются, в зависимости от топологии диаграммы переходов состояний, которая была нарисована:
Для смены состояний, следующих одно за другим, используйте команду Переход к Стадии (Stage JMP).
Когда текущая Стадия расщепляется на несколько новых состояний или Стадий, протекающих одновременно, либо когда контролирующая Стадия активизирует последовательность Стадий своей командой, используйте команду Установить Бит Стадии (Set Stage Bit).

- Если текущее состояние является последним в последовательности состояний и его задача завершена, либо когда контролирующее состояние завершает последовательность состояний своей командой, используйте команду Сброс Бита Стадии (Reset Stage Bit).

Что такое Начальная Стадия, и где ее использовать?

Начальная Стадия (ISG) автоматически активизируется после включения питания. После этого она работает, как любая другая Стадия. Если требуется, можно ввести несколько Начальных Стадий. Начальные Стадии используются в релейной программе, если они должны быть всегда активны, либо в качестве начальной точки программы.

Можно ли размещать цепи релейной программы за пределами Стадий, чтобы они были всегда активны?

Это возможно, но на практике этого делать не рекомендуется. Стадии, которые должны быть всегда активны, лучше размещать в Начальной Стадии, и не сбрасывать эту Стадию, либо использовать внутри этой Стадии команду Stage JMP. Она может запустить другие последовательности Стадий в нужное время, установив соответствующие биты включения Стадий.

Можно ли сделать так, чтобы несколько Стадий были активны одновременно?

Да, и это очень часто встречается во многих программах. При этом важно разбить свое приложение на отдельные процессы, составляющие отдельные Стадии. Грамотно спроектированный процесс будет, большей частью, состоять из последовательности Стадий, и только одна из них будет активна в определенный момент времени. Конечно, одновременно могут быть активны все процессы программы.

Глава 8. Работа контуров ПИД-регулирования

(только для DL250-1/DL260)

В этой главе…

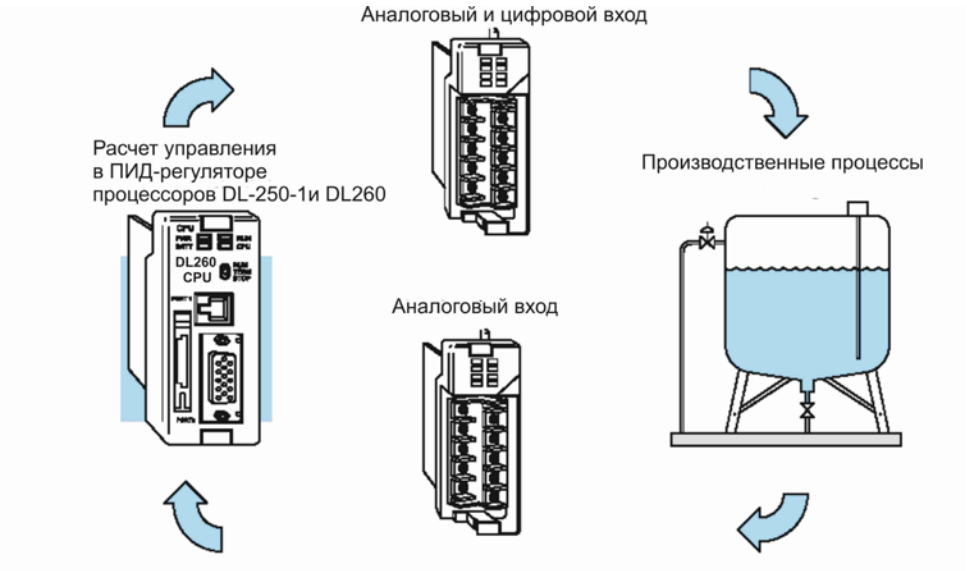
Контуры ПИД-регулирования DL250-1 и DL260

Основные возможности

Контуры регулирования, встроенные в DL250-1 и DL260, обеспечивают богатый набор функций, позволяющий решать множество различных задач управления процессами. К основным свойствам и функциям относятся:

- DL260 - до 16 контуров с независимым программированием частоты опроса
- DL250-1 - до 4 контуров с независимым программированием частоты опроса
- Возможность ручного/автоматического/каскадного управления контуром
- Наличие двух типов безударных переходов
- Полнофункциональные аварийные сигналы
- Программный задатчик с профилем до 16 участков
- Автонастройка

Помимо выполнения релейной программы процессорные модули DL250-1 и DL260 поддерживают работу контуров регулирования. Все датчики и привода, как показано ниже, подключаются к стандартным модулям ввода/вывода контроллеров DL205. Измеряемые переменные процесса, величины коэффициентов передачи, аварийные уровни сигналов и т.п., связанные с каждым контуром, хранятся в Таблице переменных контура регулирования в ЦПУ. При каждом цикле процессор считывает переменные процесса (PV), присутствующие на его входах. После этого в течение определенного временного интервала, выделяемого в каждом цикле ПЛК, ЦПУ выполняет расчет реакции контура и обновляет выходное значение на управляющем аналоговом выходе. Контуры управления используют для расчета управляющего сигнала алгоритм ПИД-регулирования (пропорционально-дифференциально-интегральный алгоритм регулирования). В этой главе описывается работа контуров регулирования и действия, необходимые для их конфигурирования и настройки.



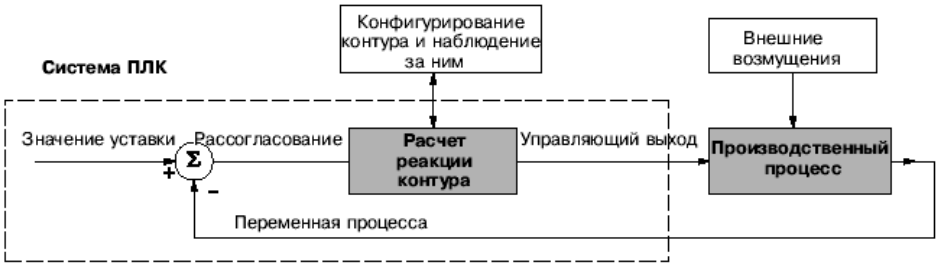
Лучшим средством настройки контуров процессорного модуля является пакет *Direct*SOFT, версия 2.1 или более поздняя. Предусмотренные в *Direct*SOFT диалоговые окна позволяют выполнить конфигурирование каждого контура в отдельности. Завершив конфигурирование, можно приступить к настройке каждого контура, используя для этого инструмент PID Trend View (Просмотр тренда ПИД-регулятора), имеющийся в *Direct*SOFT. Выбранные при конфигурировании и настройке параметры хранятся в энергонезависимой флэш-памяти процессоров. Параметры контура также можно сохранить на диск, чтобы в дальнейшем считать их с диска и использовать повторно.

Свойства контура ПИД-регулятора	Характеристики
Количество контуров	Настраивается, для DL260 максимум 16, для DL250-1 максимум 4
Требуемая V-память процессора	32 слова (V ячейки) на выбранный контур, 64 слова при использовании программного задатчика
Алгоритм ПИД-регулятора	ПИД-регулирование по положению или по скорости (Position / Velocity)
Направление управляющего выхода	Прямое или обратное действие по выбору
Зависимость рассогласования	Выбирается зависимость от рассогласования: линейная, квадратичная или пропорциональная корню квадратному
Скорость обновления контура (время между расчетами управляющего воздействия)	0.05 - 99.99 секунды, программируется пользователем
Минимальная скорость обновления контура	0.05 секунды для 1 - 4 контуров (DL250-1/260) 0.1 секунды для 5 - 8 контуров (DL260) 0.2 секунды для 9 - 16 контуров (DL260)
Режимы контуров	Автоматический, ручной (управление оператором) или каскадный
Генератор профиля программного задатчика	До 8 шагов «наклон/выдержка» (16 участков) на контур с индикацией номера шага
Обработка переменной процесса (PV)	Выбирается стандартная линейная или с извлечением квадратного корня (для входа измерения расхода)
Границы задания (Setpoint Limits)	Указываются минимальное и максимальное значения задания
Границы переменной процесса	Указываются минимальное и максимальное значения переменной процесса
Коэффициент усиления пропорционального звена (Gain)	Указывается коэффициент усиления в диапазоне 0.01…99.99
Интегрирующее звено (Reset)	Указывается время интегрирования в пределах 0.1…999.8, в секундах или минутах
Дифференцирующее звено (Rate)	Указывается время дифференцирования в пределах 0.01…99.99 секунды
Границы производной	Указывается предельное значение коэффициента передачи дифференцирующего звена от 1 до 20
Безударный переход I	Автоматическая инициализация смещения и уставки при переключении из ручного управления в автоматическое
Безударный переход II	При переключении из ручного управления в автоматическое смещение автоматически устанавливается равным значению на управляющем выходе
Пошаговое (ступенчатое) смещение	При больших изменениях задания обеспечивает постепенное смещение рабочей точки
Анти-выбег (Anti-windup)	В случае ПИД-регулирования по положению останавливает работу интегратора, когда управляющий выход достигает 0% или 100% (чем ускоряет возвращение контура к исходному режиму, когда выход выходит из насыщения)
Зона нечувствительности рассогласования (Error Deadband)	Задается допуск на составляющую рассогласования (SP - PV), в пределах которого значение управляющего выхода не изменяется

Аварийный сигнал	Назначение и характеристики
Зона нечувствительности (Deadband)	Зона нечувствительности указывается в пределах 0.1%…5% для всех аварийных сигналов
Аварийные уровни переменной процесса (PV Alarm Points)	Выбираются аварийные уровни переменной процесса: Самый Низкий, Низкий, Высокий и Самый Высокий
Отклонение переменной процесса (PV Deviation)	Указываются два аварийных уровня отклонения переменной процесса от величины задания
Скорость изменения (Rate of Change)	Обнаружение превышения скорости изменения переменной процесса от указанной предельной скорости

Основы построения контуров ПИД-регулирования

На следующем рисунке представлены ключевые элементы контура ПИД-регулирования. Цепь от ПЛК к производственному процессу и обратная цепь к ПЛК образуют "петлю" замкнутого контура управления.



Производственный процесс – это набор операций по переработке сырья в готовую продукцию. В ходе процесса могут изменяться физические и/или химические свойства материала. В результате изменений материал становится пригодным для использования в определенных целях, являясь, в итоге, конечным продуктом переработки.

Переменная процесса – значение некоторого измеряемого физического свойства сырья. Для измерений используются датчики различных типов. Например, если в производственном процессе используется печь, скорее всего, потребуется регулировать температуру. В этом случае переменной процесса является температура.

Значение задания (уставки) – теоретически идеальное значение переменной процесса или заданное значение, обеспечивающее продукт наилучшего качества. Зная это значение, оператор установки либо задает его вручную, либо указывает в программе ПЛК для дальнейшего использования в ходе автоматического управления установкой.

Внешнее возмущение – непредсказуемые источники ошибок, последствия которых система управления стремится устранить, компенсируя их. Например, если интенсивность подачи топлива не изменяется, печь будет нагреваться сильнее в теплую погоду, чем в холодную. Система управления печью должна противодействовать этому явлению, чтобы поддерживать температуру в печи постоянной в любое время года. Таким образом, погода, которая не слишком предсказуема, является одним из источников возмущений для данного процесса.

Рассогласование – алгебраическая разность между переменной процесса и заданием. Рассогласование является ошибкой контура управления и равно нулю, когда переменная процесса равна уставке (требуемому значению). В правильно работающем контуре управления поддерживается минимальное значение рассогласования.

Расчет реакции контура – применение в реальном времени к сигналу рассогласования математического алгоритма, на основании которого вырабатывается такой управляющий сигнал, который минимизирует величину рассогласования. Существуют различные типы алгоритмов управления. В DL250-1 и DL260 используется алгоритм ПИД – регулирования (Пропорционально-Интегрально-Дифференциальный), о котором более подробно будет рассказано позже.

Управляющий выход – результат расчета математических выражений, описывающих контур, являющийся командой управления для процесса (например, уровень нагрева в печи).

Конфигурирование контура – ввод оператором параметров, задающих и оптимизирующих работу контура регулирования. Заданные при конфигурировании параметры используются функцией расчета реакции контура в реальном времени для подстройки коэффициентов передач, уровней смещения и т.п.

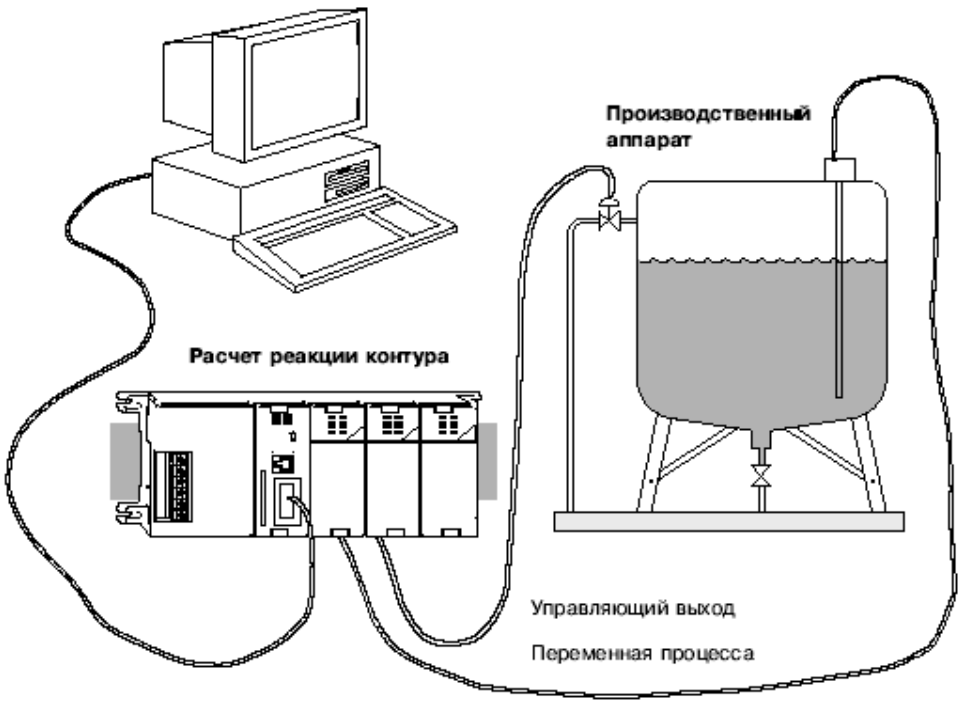
Наблюдение за контуром – функция, позволяющая оператору следить за состоянием и работой контура управления. Она используется совместно с конфигурированием контура для оптимизации работы контура (минимизации рассогласования).

Для разработки и понимания контуров регулирования важно связать каждый элемент контура с его реальным физическим компонентом. Взгляните на нижний рисунок. В приведенном примере показан производственный аппарат реактора. Датчик измеряет переменную процесса, которая может быть давлением, температурой или другим параметром. Сигнал датчика усиливается преобразователем и передается по проводам в аналоговой форме на входной модуль ПЛК.

ПЛК читает переменную на аналоговом входе. Процессор выполняет расчет контура и осуществляет запись управляющего сигнала в ячейку модуля аналогового выхода.

Выходной управляющий сигнал может быть не только аналоговым (пропорциональным), но и цифровым (включить/выключить) в зависимости от установок контура. Этот сигнал передается исполнительному механизму для управления нагревателем, клапаном, насосом и т.п. для выполнения физических или химических изменений жидкости в реакторе. Спустя некоторое время изменение регулируемого параметра жидкости становится достаточным, чтобы быть обнаруженным датчиком. Соответственно изменяется и переменная процесса. Выполняется следующий расчет контура, и таким образом контур работает непрерывно.

Конфигурирование контура и наблюдение за ним



На компьютере, изображенном на рисунке, работает программный пакет *Direct*SOFT. В состав программного пакета входит редактор форм, служащий для конфигурирования параметров контура. Кроме того, в нем предусмотрен экран просмотра тренда контура ПИД-регулирования, который крайне полезен при настройке контура. Подробное использование данного программного пакета приводится в руководстве по *Direct*SOFT.

Параметры настройки контура

Таблица контуров и количество контуров

Процессоры DL250-1 и DL260 получают команды для контура ПИД-регулирования только из таблиц в V-памяти. Это означает, что в языке релейной логики RLL нет команды "ПИД-регулирование". Процессор читает параметры контура из зарезервированных ячеек V-памяти. Как показано в таблице ниже, в ячейке V7640 необходимо задать значение, указывающее на начальный адрес основной таблицы контура. Затем в V7641 необходимо указать количество контуров, для которых ЦПУ будет выполнять расчеты. В V7642 содержатся флаги ошибок, которые устанавливаются, если значения в V7640 или в V7641 заданы неправильно.

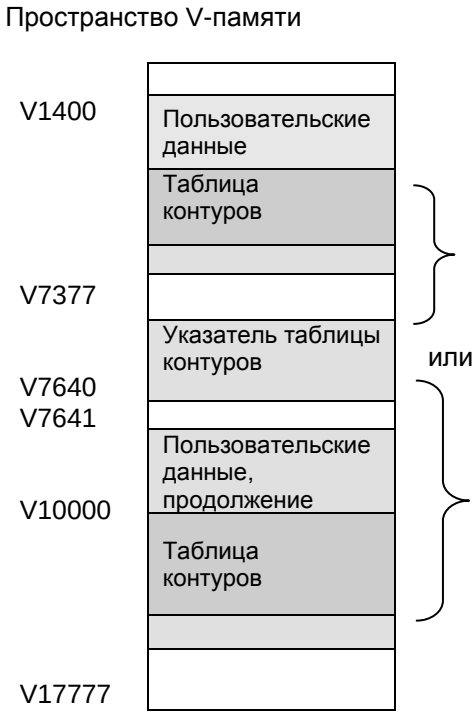
Адрес	Параметр настройки	Тип данных	Диапазон	Чтение/ запись
V7640	Указатель таблицы параметров контура	Восьмерич-ные	V1400 -V7340, V10000 -V17740 (DL250-1) V10000 –V35740 (DL260)	Запись
V7641	Количество контуров	BCD	0-4 (DL250-1) 0 –16 (DL260)	Запись
V7642	Флаги ошибок конту-ра	Двоичные	0 или 1	Чтение

Если количество контуров равно «0», то во время выполнения релейной программы задача расчета контуров отключена.

Контроллер контуров позволяет использовать контуры в порядке возрастания, начиная с 1. Например, нельзя использовать контуры 1 и 4, пропустив 2 и 3. Контроллер контура пытается управлять всеми контурами, заданными в V7641.

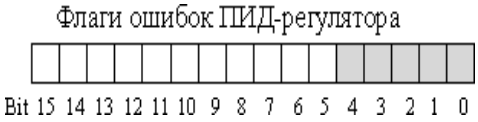
Таблица параметров контуров может занимать блок памяти в нижней области данных пользователя (V1400 — V7377) или в верхней области данных пользователя (V10000 — V17777 для DL250-1 и V10000 – V35740 для DL260), как показано на рисунке справа. Убедитесь, что выбранного вами адресного пространства достаточно для приложения. Значение в V7641 сообщает процессору размер таблицы, учитывая, что для каждого контура используется 32 ячейки.

Диалоговое окно *Direct*SOFT Настройка ПИД-регулятора (PID Setup) предлагает вам способ задания параметров. Можно также использовать команды программы RLL, например, LDA или LD, и команды OUT. Однако, эти ячейки памяти являются частью системной памяти для реализации контуров регулирования, поэтому запись их из программы RLL не обязательна.



Флаги ошибок контуров ПИД-регулирования

ЦПУ сообщает о любых ошибках программирования параметров настройки в V7640 и в V7641, устанавливая соответствующие биты в V7642, при переходе из режима программирования в режим выполнения.



При использовании диалогового окна настройки контуров в *Direct*SOFT встроенная автоматическая проверка диапазона препятствует возможным ошибкам настройки. Однако параметры настройки могут быть записаны другими способами, например, из программы релейной логики, поэтому в подобных случаях регистр флагов ошибок может оказаться полезным.

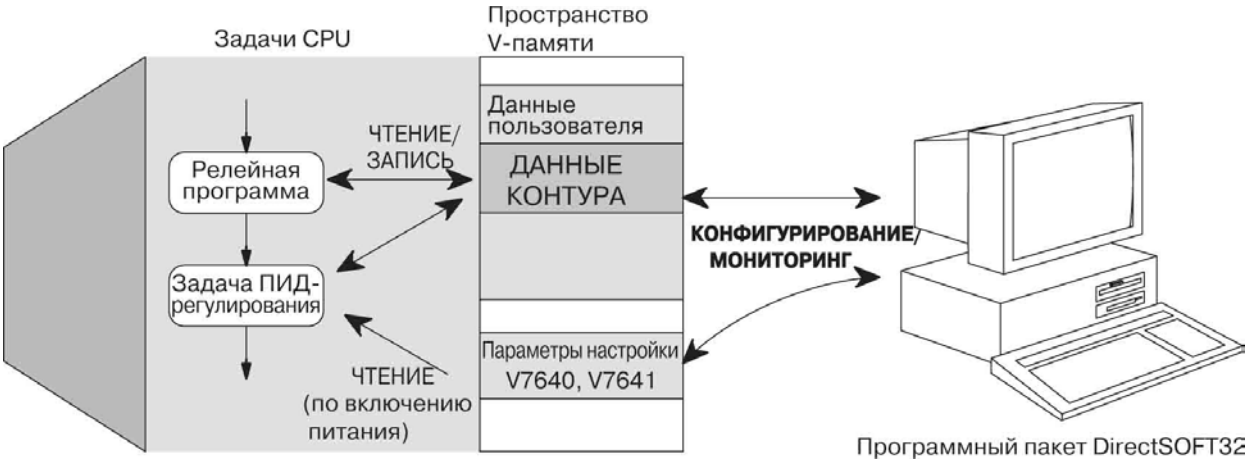
В следующей таблице перечислены ошибки, о которых сообщается в V7642.

Бит	Описание ошибки (0 = нет ошибки, 1 = ошибка)
0	Начальный адрес (в V7640) выходит за пределы нижней области V - памяти.
1	Начальный адрес (в V7640) выходит за пределы верхней области V -
2	Выбранное количество контуров (в V7641) превышает 4 (DL250-1) или 16 (DL260).
3	Таблица контуров вышла за пределы V7377. Используйте адрес ближе к V1400
4	Таблица контуров вышла за пределы V17777 (DL250-1) или V35777 (DL260). Используйте адрес ближе к V10000.

В режиме выполнения отсутствие ошибок программирования можно быстро проверить по значению регистра V7642. Оно должно быть равно 0000.

Задание размера и местонахождения таблицы контуров

Процесс считывания параметров настройки контуров процессором при переходе из режима программирования в режим выполнения представлен на рисунке ниже. В этот момент ЦПУ получает сведения о размещении таблицы контуров и о количестве сконфигурированных в ней контуров. Затем при циклическом выполнении релейной программы параметры контура используются задачей расчета реакции контура ПИД-регулирования для выполнения расчетов, генерации аварийных сообщений (сигналов) и т.п. Некоторые параметры таблицы контуров ЦПУ считывает или записывает в каждой операции расчета контура.



В таблице параметров контуров содержатся данные только для того количества контуров, которое было задано в V7641. Каждый сконфигурированный контур занимает 32 слова (0…37 в восьмеричной системе) в таблице контуров. Например, предположим, что мы используем четыре контура и в качестве начальной ячейки выбрана V2000. Параметры контура 1 будут находиться в словах V2000 - V2037, контура 2 – в V2040 - V2077 и т. д. Для контура 4 будут введены слова V2140 и V2177.

Описание слов таблицы контуров

Параметры, которыми характеризуется каждый контур, перечислены в следующей таблице. Поиск определенного параметра в таблице контуров облегчается благодаря смещению, приведенному в восьмеричном формате. Например, если таблица начинается со слова V2000, то параметр интегрирующего звена (интеграла) размещается по адресу Addr+11, то есть, V2011. Не пользуйтесь для вычисления адреса номером слова.

№ слова	Адрес + смещение	Описание	Формат	Оперативное чтение
1	Addr + 0	Настройки контура ПИД. Слово 1	биты	Да
2	Addr + 1	Настройки контура ПИД. Слово 2	биты	Да
3	Addr + 2	Значение уставки (задания) (SP)	слово /двоичный	Да
4	Addr + 3	Переменная процесса (PV)	слово /двоичный	Да
5	Addr + 4	Значение смещения (интегратора)	слово /двоичный	Да
6	Addr + 5	Значение управляющего выхода	слово /двоичный	Да
7	Addr + 6	Режим работы контура и состояние аварийных сигналов	биты	-
8	Addr + 7	Частота опроса (Sample Rate)	слово /BCD	Да
9	Addr + 10	Коэффициент усиления пропорционального звена(Gain)	слово /BCD	Да
10	Addr + 11	Время интегрирующего звена (Reset)	слово /BCD	Да
11	Addr + 12	Время дифференцирующего звена (Rate)	слово /BCD	Да
12	Addr + 13	Значение PV, аварийный уровень "самый низкий"	слово /двоичный	Нет*
13	Addr + 14	Значение PV, аварийный уровень "низкий"	слово /двоичный	Нет*
14	Addr + 15	Значение PV, аварийный уровень "высокий"	слово /двоичный	Нет*
15	Addr + 16	Значение PV, аварийный уровень "самый высокий"	слово /двоичный	Нет*
16	Addr + 17	Значение PV, аварийный уровень отклонения ("ЖЕЛТЫЙ")	слово /двоичный	Нет*
17	Addr + 20	Значение PV, аварийный уровень отклонения ("КРАСНЫЙ")	слово /двоичный	Нет*
18	Addr + 21	Значение PV, аварийный сигнал скорости изменения	слово /двоичный	Нет*
19	Addr + 22	Значение PV, гистерезис сигнала аварии	слово /двоичный	Нет*
20	Addr + 23	Значение PV, зона нечувствительности рассогласования	слово /двоичный	Да
21	Addr + 24	Постоянная фильтра нижних частот PV	слово /BCD	-
22	Addr + 25	Признак ограничения коэффициента передачи дифференцирующего звена	слово /BCD	Нет**
23	Addr + 26	Значение SP, нижнее предельное значение	слово /двоичный	Да
24	Addr + 27	Значение SP, верхнее предельное значение	слово /двоичный	Да
25	Addr + 30	Нижнее предельное значение управляющего выхода	слово /двоичный	Нет**
26	Addr + 31	Верхнее предельное значение управляющего выхода	слово /двоичный	Нет**
27	Addr + 32	Указатель адреса V-памяти значения удаленного задания (Remote SP)	слово/ шестнадцатеричный	Да
28	Addr + 33	Флаг программного задатчика (Ramp/Soak)	биты	Да
29	Addr + 34	Начальный адрес таблицы программного задатчика	слово/ шестнадцатеричный	Нет**
30	Addr + 35	Флаги ошибок таблицы программного задатчика	биты	Нет**
31	Addr + 36	Прямой доступ к PV, номер /указатель база/слот /канал	слово/ шестнадцатеричный	Да
32	Addr + 37	Прямой доступ к управляющему выходу, номер канала	слово/ шестнадцатеричный	Да

* Данные считываются только тогда, когда бит активации аварийных сигналов переходит из 0 в 1

** Данные считываются только при изменении режима работы ПЛК

Назначение битов Слова 1 параметров настройки ПИД-регулятора (Addr+00)

В следующей таблице описано назначение отдельных битов Слова 1 "Параметры настройки ПИД-регулятора" (Addr+00). Более подробно каждый из них описан в соответствующем разделе данной главы.

Бит	Назначение битов Слова 1	Чтение/ Запись	Бит=0	Бит= 1
0	Запрос на работу контура в Ручном режиме	запись	–	0⇒1 запрос
1	Запрос на работу контура в Автоматическом режиме	запись	–	0 ⇒1 запрос
2	Запрос на работу контура в Каскадном режиме	запись	–	0⇒1 запрос
3	Выбор режима безударного перехода	запись	Режим I	Режим II
4	Прямое или обратное действие выхода	запись	Прямой	Обратный
5	Тип алгоритма ПИД: положение/ скорость	запись	Положение	Скорость
6	Преобразование PV: линейное / квадратный корень	запись	Линейное	Кв. корень
7	Выбор линейной/ квадратичной зависимости рассогласования	запись	Линейная	Квадратичная
8	Введение зоны нечувствительности рассогласования	запись	Нет	Да
9	Выбор предела коэффициента усиления дифференцирующего звена	запись	Выкл	Вкл
10	Выбор "замораживания" смещения (интегратора)	запись	Выкл	Вкл
11	Режим работы программного задатчика	запись	Выкл	Вкл
12	Контроль аварийного уровня PV	запись	Выкл	Вкл
13	Контроль аварийного уровня отклонения PV	запись	Выкл	Вкл
14	Контроль аварийного уровня скорости изменения PV	запись	Выкл	Вкл
15	Режим работы контура не зависит от режима ЦПУ при установке бита в 1.	запись	Контур зависит от режима ЦПУ	Контур не зависит от режима ЦПУ

Назначение битов Слова 2 параметров настройки ПИД-регулятора (Addr+01)

В следующей таблице описано назначение отдельных битов Слова 2 "Параметры режима ПИД-регулятора" (Addr+01). Более подробно каждый из них описан в соответствующем разделе данной главы.

Бит	Назначение битов Слова 2	Чтение/ Запись	Бит = 0	Бит = 1
0	Тип сигнала входной переменной (PV) и выходного диапазона (см. Прим. 1 и 2)	запись	Униполярный	Биполярный
1	Формат данных ввода/вывода контура (см. Прим.1 и 2)	запись	12 бит	15 бит
2	Входной аналоговый фильтр/ Автоматическая передача	запись	выкл	вкл
3	Ограничение на величину вводимого задания (SP)	запись	Невозможно	Возможно
4	Выбор единиц коэффициента передачи интегратора (Reset)	запись	секунды	минуты
5	Выбор алгоритма ПИД для автонастройки	запись	Замкнутый контур	Разомкнутый контур
6	Выбор варианта автонастройки	запись	ПИД	Только PI (дифф. = 0)
7	Запуск автонастройки	чтение /запись	Автонастройка выполнена	Принудит. запуск
8	Тактовый сигнал цикла ПИД-регулятора (для внутреннего использования)	чтение	–	–
9	Выбор 16-ти битового формата данных (см. Прим.1 и 2)	запись	Не 16 бит	16 бит
10	Установить разные форматы данных для ввода и вывода (см. Прим. 2 и 3)	запись	Одинаковые	Разные
11	Диапазон управляющего выхода: одно/биполярный (см. Прим. 2 и 3)	запись	Однополярный	Биполярный
12	Выбор формата управляющего выхода (см. Прим. 2 и 3)	запись	12 бит	15 бит
13	Выбор 16-ти битового формата данных выхода (см. Прим. 2 и 3)	запись	Не 16 бит	16 бит
14-15	Зарезервирован для будущего использования	–	–	–

Примечание 1. Если значение 9-го бита равно 0, тогда биты 0 и 1 читаются. Если значение 9-го бита равно 1, тогда биты 0 и 1 не читаются, а бит 9 определяет формат данных (диапазон автоматически однополярный).

Примечание 2. Если значение 10-го бита равно 0, тогда значения в битах 0, 1 и 9 определяют диапазон и формат данных ввода/вывода (биты 11, 12 и 13 не читаются). Если значение 10-го бита равно 1, тогда значения в битах 0, 1 и 9 определяют только диапазон и формат данных ввода, а биты 11, 12 и 13 определяют формат данных вывода.

Примечание 3. Если значение 10-го бита равно 1 и значение 13-го бита равно 0, биты 11 и 12 определяют диапазон и формат данных вывода. Если значения 10-го бита и 13-го бита оба равны 1, биты 11 и 12 не читаются и бит 13 определяет формат данных вывода (диапазон автоматически однополярный).

Режим работы контура и состояние аварийных сигналов (Addr+06)

В таблице приводится описание отдельных битов слова "Режим работы контура и состояние аварийных сигналов" (Addr+06). Подробности можно найти в разделах "Режим контура ПИД-регулирования" и "Аварийные сигналы".

Бит	Описание битов режима/аварии	Чтение/ Запись	Бит=0	Бит=1
0	Индикация Ручного режима	чтение	-	Ручной
1	Индикация Автоматического режима	чтение	-	Авто
2	Индикация Каскадного режима	чтение	-	Каскад
3	Аварийный сигнал входа PV Самый НИЗКИЙ	чтение	Выкл	Вкл
4	Аварийный сигнал входа PV НИЗКИЙ	чтение	Выкл	Вкл
5	Аварийный сигнал входа PV ВЫСОКИЙ	чтение	Выкл	Вкл
6	Аварийный сигнал входа PV Самый ВЫСОКИЙ	чтение	Выкл	Вкл
7	Аварийный сигнал рассогласования PV Желтый (YELLOW)	чтение	Выкл	Вкл
8	Аварийный сигнал рассогласования PV Красный (RED)	чтение	Выкл	Вкл
9	Аварийный сигнал скорости изменения входа PV	чтение	Выкл	Вкл
10	Ошибка программирования значения аварийного сигнала	чтение	-	Ошибка
11	Переполнение/потеря значимости при расчете контура	чтение	-	Ошибка
12	Индикация нахождения контура в автонастройке	чтение	Выкл	Вкл
13	Индикация ошибки автонастройки	чтение	Выкл	Вкл
14-15	Зарезервированы для будущего использования	-	-	-

Флаги таблицы программного задатчика (Addr+33)

В следующей таблице описано назначение отдельных битов флагов таблицы программного задатчика (Ramp/Soak Profile) (Addr+33). Дополнительные сведения можно найти в разделе "Работа программного задатчика".

Бит	Описание битов флагов программного задатчика	Чтение/ Запись	Бит=0	Бит=1
0	Запуск программного задатчика	запись	-	0⇒1 Старт
1	Остановка программного задатчика	запись	-	0⇒1 Пауза
2	Возобновление работы программного задатчика	запись		0⇒1 Возобн.
3	Перевод программного задатчика на один шаг	запись	-	0⇒1 Перевод
4	Завершение работы программного задатчика	чтение	-	Завершен
5	Отклонение входа PV от сигнала программного задатчика	чтение	Выкл	Вкл
6	Программный задатчик приостановлен	чтение	Выкл	Вкл
7	Зарезервирован	чтение	-	-
8-15	Текущий шаг программного задатчика	чтение	Декодируется как байт (шестнадцатеричный)	

Биты 8-15 должны считываться как байт, показывающий номер текущего шага в профиле программного задатчика. Этот байт может принимать значения 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F и 10, представляя шаги с 1 по 16, соответственно. Если байт = 0, то таблица программного задатчика неактивна.

Местонахождение таблицы программного задатчика (Addr+34)

У каждого конфигурируемого контура есть возможность использовать встроенный программный задатчик, предназначенный для данного контура. При этом создается непрерывный массив значений сигналов задания SP, называемый профилем. Для использования программного задатчика нужно создать отдельную таблицу из 32 слов с соответствующими значениями. Диалоговое окно *Direct*SOFT упрощает этот процесс.

В основной таблице контура указатель на таблицу программного задатчика (Addr+34) должен указывать на начало данных программного задатчика для этого контура. Они могут располагаться в любом месте пользовательских данных, необязательно рядом с таблицей параметров контура, что и показано слева. Для каждой таблицы программного задатчика независимо от количества заданных участков требуется 32 слова.

Параметры таблицы программного задатчика определяются в приведенной ниже таблице. Дальнейшие подробности описаны в разделе «Работа программного задатчика» этой главы.

Смещение адреса	Шаг	Описание	Смещение адреса	Шаг	Описание
+ 00	1	Значение SP окончания участка наклона	+ 20	9	Значение SP окончания участка наклона
+ 01	1	Крутизна наклона	+ 21	9	Крутизна наклона
+ 02	2	Длительность участка выдержки	+ 22	10	Длительность участка выдержки
+ 03	2	Отклонение PV на участке выдержки	+ 23	10	Отклонение PV на участке выдержки
+ 04	3	Значение SP окончания участка наклона	+ 24	11	Значение SP окончания участка наклона
+ 05	3	Крутизна наклона	+ 25	11	Крутизна наклона
+ 06	4	Длительность участка выдержки	+ 26	12	Длительность участка выдержки
+ 07	4	Отклонение PV на участке выдержки	+ 27	12	Отклонение PV на участке выдержки
+ 10	5	Значение SP окончания участка наклона	+ 30	13	Значение SP окончания участка наклона
+ 11	5	Крутизна наклона	+ 31	13	Крутизна наклона
+ 12	6	Длительность участка выдержки	+ 32	14	Длительность участка выдержки
+ 13	6	Отклонение PV на участке выдержки	+ 33	14	Отклонение PV на участке выдержки
+ 14	7	Значение SP окончания участка наклона	+ 34	15	Значение SP окончания участка наклона
+ 15	7	Крутизна наклона	+ 35	15	Крутизна наклона
+ 16	8	Длительность участка выдержки	+ 36	16	Длительность участка выдержки
+ 17	8	Отклонение PV на участке выдержки	+ 37	16	Отклонение PV на участке выдержки

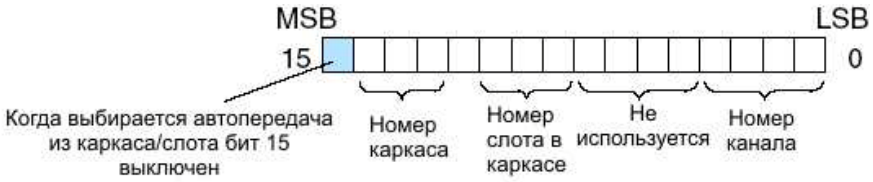
Флаги ошибок программирования таблицы программного задатчика (Addr+35)

В следующей таблице приводится описание отдельных битов флагов ошибок программирования таблицы программного задатчика (Addr+35). Дополнительные сведения можно найти в разделах "Режимы контура ПИД-реглирования" и "Аварийные сигналы".

Бит	Описание бита флага ошибки программного задатчика	Чтение/ Запись	Бит=0	Бит=1
0	Начальный адрес нижнего диапазона V-памяти	чтение	-	Ошибка
1	Начальный адрес верхнего диапазона V-памяти	чтение	-	Ошибка
2-3	Зарезервирован для будущего использования	-	-	-
4	Начальный адрес диапазона системных параметров V- памяти	чтение	-	Ошибка
5-15	Зарезервированы для будущего использования	-	-	-

Прямой доступ PV (Addr + 36) с выбором каркаса/ модуля/канала

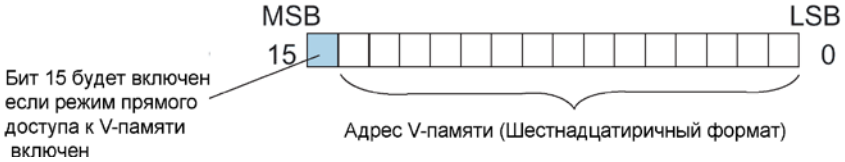
В следующей таблице перечислены определения отдельных слогов слова (Addr+36) автоматической передачи значения переменной процесса PV из модуля ввода/вывода (каркас/слот/канал). **Если этот вариант передачи используется с любым каналом аналогового выходного модуля, то к этому модулю при программировании релейной логики нельзя применять метод указателя.** (Обратитесь к руководству пользователя аналоговыми модулями D2-ANLG-M для получения информации о методе указателя.)



Процессорный модуль	Номер каркаса	Номер слота в каркасе	Номер канала
DL250-1	Процессорный каркас = 0 Каркас расширения = 1- 2	0 - 7	1 - 8
DL260	Процессорный каркас = 0 Каркас расширения = 1- 4		

Прямой доступ PV (Addr+36) с выбором ячейки V-памяти

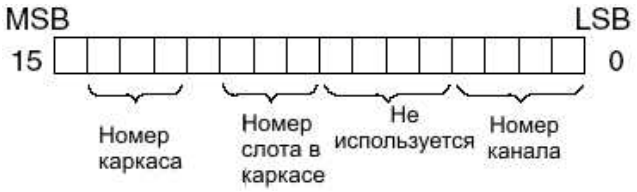
В следующей таблице приведено определение слова (Addr+36) автоматической передачи значения переменной процесса PV из V-памяти. Метод указателя при программировании релейной логики может использоваться в этом варианте для того, чтобы передать данные в V-память из аналогового модуля. Если этот вариант передачи используется с любым каналом аналогового входного модуля, то к этому модулю нельзя применять метод указателя.



Тип памяти	Диапазон для DL250-1	Диапазон DL260
V-память V	V1400 – V7377 V10000 – V17777	V400 – V677 V1400 – V7377 V10000 – V35777

Прямой доступ к управляющему выходу (Addr+37)

В следующей таблице перечислены определения отдельных слогов слова (Addr+37) автоматической передачи значения управляющего выхода. **Если этот вариант передачи используется с любым каналом аналогового выходного модуля, то к этому модулю при программировании релейной логики нельзя применять метод указателя.** (Обратитесь к руководству пользователя аналоговыми модулями (D2-ANLG-M) для получения информации о методе указателя.)



Процессорный модуль	Номер каркаса	Номер слота в каркасе	Номер канала
DL250-1	Процессорный каркас = 0 Каркас расширения = 1-2	0 - 7	1 - 8
DL260	Процессорный каркас = 0 Каркас расширения = 1-4		

Частота опроса и планирование контура

Частота опроса контура

На рисунке справа показаны основные задачи ЦПУ. Эти задачи выполняются в каждом цикле ПЛК, когда ЦПУ находится в режиме выполнения (Run). Обратите внимание, что расчеты реакций ПИД-контуров выполняются после реализации задачи релейной программы.

Работа ПИД-контуров может происходить даже тогда, когда программа не работает. Для этого следует выбрать прямой доступ в Addr+36 и установить бит 15 в Addr+00.

Частота опроса контура управления — это просто частота расчета реакции контура ПИД-регулирования. Результатом каждого расчета является новое значение управляющего выхода. Для процессоров DL250-1 и DL260 частоту опроса контура можно задавать в диапазоне от 50 мс до 99.99 секунд. Для большинства контуров расчет реакции ПИД-регулятора выполняется не для каждого цикла сканирования. Реально для некоторых контуров расчеты могут понадобиться только раз в 1000 циклов.

Вы выбираете нужную частоту для каждого контура, и процессор автоматически планирует и выполняет расчеты реакции ПИД-регулятора в соответствующих циклах.

Выбор наилучшей частоты опроса

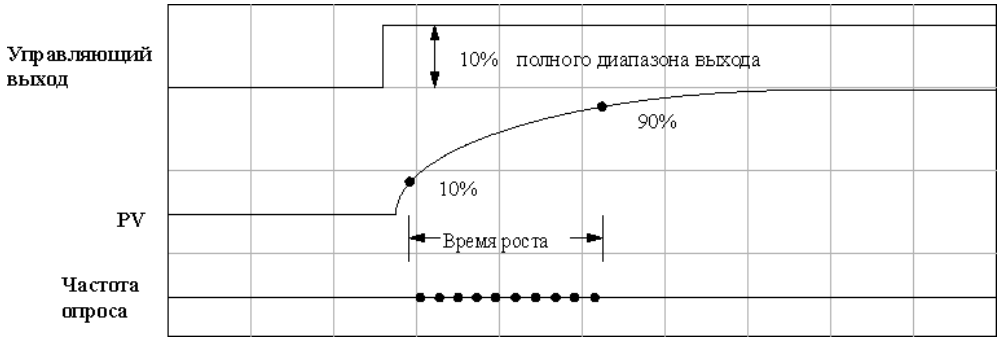
Для любого конкретного контура управления невозможно подобрать единственную идеальную частоту. Оптимальная частота представляет собой компромисс, одновременно удовлетворяющий различным условиям:

- Требуемая частота опроса пропорциональна времени отклика переменной процесса (PV) на изменение управляющего выхода. Обычно, инерционным процессам соответствует малая частота опроса, а для быстрых (малоинерционных) процессов требуется высокая частота опроса.
- Большая частота опроса обеспечивает более гладкое поведение управляющего выхода и точное значение PV, но использует больше процессорного времени. При излишне высокой частоте вычислительные ресурсы процессора расходуются вхолостую.
- Низкая частота опроса обеспечивает более грубое управление и меньшую точность PV, но использует меньше процессорного времени.
- Слишком низкая частота опроса приводит к нестабильности системы, особенно при изменении уставки и при возникновении возмущений.

Для начала частоту опроса можно выбрать с таким расчетом, чтобы избежать нестабильности управления (что чрезвычайно важно). Чтобы определить начальную частоту опроса, выполните действия, перечисленные на следующей странице:

Определение оптимальной частоты опроса (Addr+07)

Выполните управление процессом с разомкнутым контуром (к этому моменту контур может быть даже еще не сконфигурирован). Переведите ЦПУ в режим выполнения (а контур в режим ручного управления, если он уже был сконфигурирован). Вручную установите на управляющем выходе такое значение, чтобы переменная PV устойчиво находилась в середине безопасного диапазона. В определенный момент, когда процесс испытывает незначительное внешнее возмущение, измените значение на управляющем выходе сразу на 10%. Зафиксируйте время роста или спада PV (интервал между уровнем 10% и 90%). Полученный интервал роста или спада поделите на 10. Это и будет начальное значение периода опроса, с которого можно начать настройку контура.

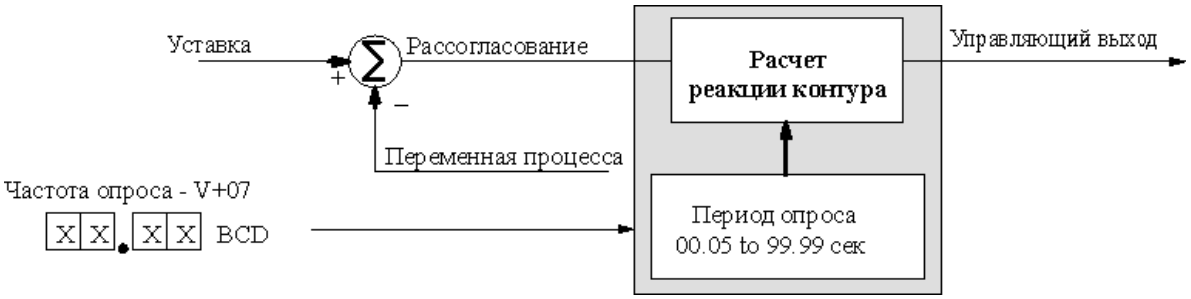


Предположим, что измеренное время роста PV вследствие изменения управляющего выхода составило 25 секунд (см. рис. выше). Рекомендуемая частота опроса для этого случая составит 2.5 секунды. Для наглядности процесс опроса показан в виде десяти точек на линии частоты опроса в пределах интервала роста PV. Эти точки показывают частоту выполнения расчетов реакции ПИД-контура, когда PV претерпевает изменения. Разумеется, в ходе работы опрос контура и расчет реакции производятся постоянно.

ПРИМЕЧАНИЕ: *Слишком большая частота опросов уменьшит доступное разрешение аварийного сигнала скорости изменения переменной PV, так как аварийное значение скорости определяется как изменение PV за период опроса. Например, период опроса 50 мс означает, что наименьшая обнаруживаемая скорость изменения PV составит 20 единиц PV (изменений младшего значащего бита) в секунду, или 1200 изменений младшего значащего бита в минуту.*

Задание частоты опроса

В таблице параметров контуров для каждого контура отведено поле частоты опроса. Как показано на рисунке ниже, в ячейке V+07 содержится число (в формате BCD) от 00.05 до 99.99 (с подразумевающейся десятичной точкой). Оно соответствует значениям от 50 мс до 99.99 с. Это число можно запрограммировать с помощью экрана PID Setup (Настройка ПИД-контура) в *Direct*SOFT, или записав это число в V-память любым другим способом. Для правильной работы контура это число должно быть обязательно задано.



Влияние контура ПИД-регулирования на длительность цикла ЦПУ

Поскольку расчет реакции ПИД-контура является одной из задач, выполняемых ЦПУ в пределах цикла, использование ПИД-контуров приведет к увеличению средней длительности цикла. Длительность цикла увеличивается пропорционально количеству используемых контуров и частоте опроса каждого контура.

Время выполнения расчета реакции одного контура зависит от количества выбранных опций (аварийные сигналы, квадратичная ошибка и т.п.). В таблице справа приведены ориентировочные значения времени расчета.

Время расчета ПИД-регулятора

Минимум	150 мкс
Типично	250 мкс
Максимум	350 мкс

Чтобы рассчитать, насколько вырастет длительность цикла, нам также требуется знать (или оценить) длительность цикла (выполнения) релейной программы (без контуров). При вводе одинаковых затрат на расчет реакции ПИД-регулятора в программу с малой длительностью цикла и в программу с большой длительностью цикла, в первом случае длительность цикла вырастет в процентном отношении меньше, чем во втором. Для расчета средней длительности цикла используется следующая формула:

$$\text{Средняя длительность цикла с контуром ПИД-регулирования} = \left[\frac{\text{Длительность цикла без контура}}{\text{Период опроса контура}} \times \text{Длительность расчета контура} \right] + \text{Длительность цикла без контура}$$

Например, пусть оценочная длительность цикла сканирования без расчета контуров составляет 50 мс, а период опроса контура — 3 с. Вычислим новую длительность цикла:

$$\text{Средняя длительность цикла с контуром ПИД-регулирования} = \left[\frac{50\ \text{мс}}{3\ \text{с}} \times 250\ \text{мкс} \right] + 50\ \text{мс} = 50{,}004\ \text{мс}$$

Как показывают расчеты, добавление только одного контура с малой частотой опроса влияет на длительность цикла незначительно. Теперь расширим приведенное выше выражение, чтобы показать влияние произвольного количества контуров:

$$\text{Средняя длительность цикла с контуром ПИД-регулирования} = \left[\sum_{n=1}^{n=L} \frac{\text{Длительность цикла без контура}}{\text{Период опроса контура n-ого контура}} \times \text{Длительность расчета контура} \right] + \text{Длительность цикла без контура}$$

В новом выражении увеличение длительности для каждого контура от 1 до L (последний контур) суммируется (внутри квадратных скобок) и полученная сумма добавляется к правому слагаемому "длительность цикла без контуров" только один раз. Допустим, в ПЛК используются четыре контура регулирования. В следующей таблице приводятся параметры и суммарные значения для каждого контура.

Номер контура	Описание	Период опроса контура	Суммируемый член
1	Поток пара, впускной клапан	0.25 с	50 мкс
2	Температура водяной бани	30 с	0.42 мкс
3	Уровень красителя, основной резервуар	10 с	1.25 мкс
4	Давление пара, автоклав	1.5 с	8.3 мкс

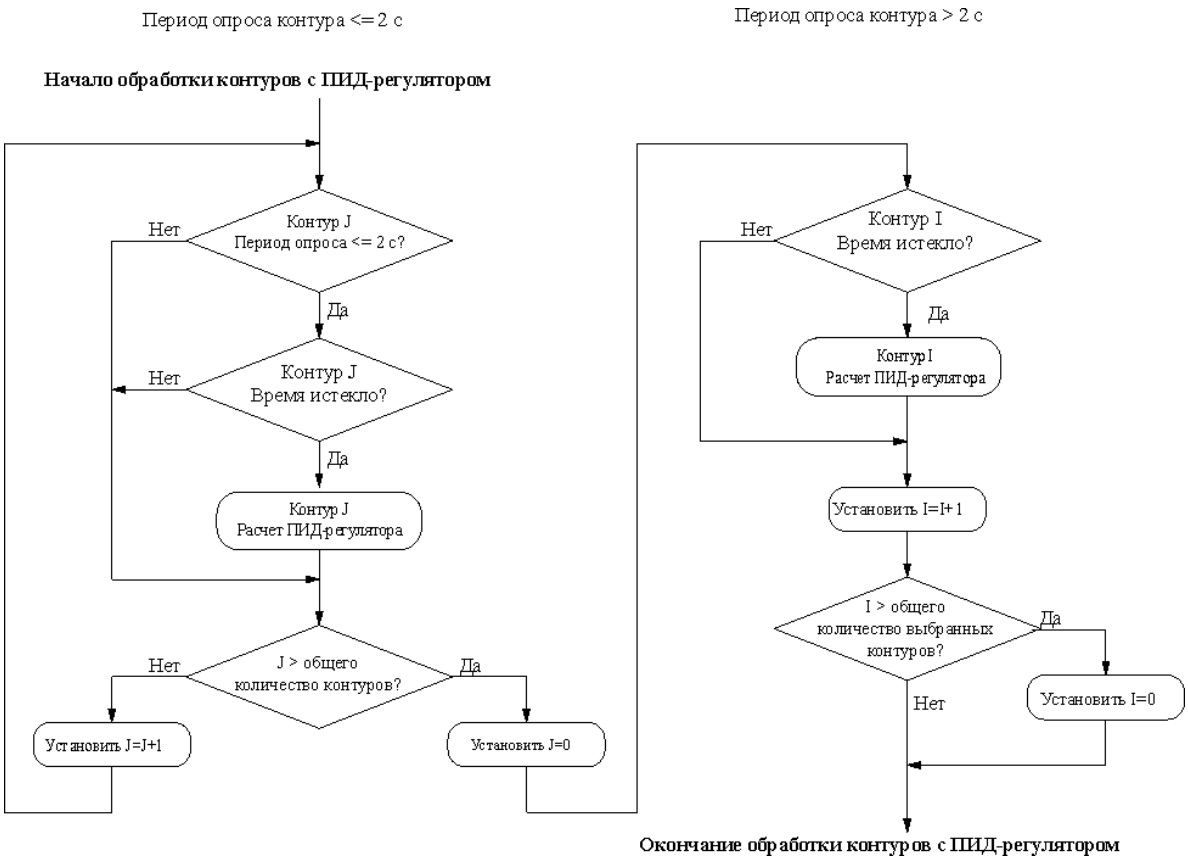
Теперь, складывая суммируемые члены и первоначальное значение длительности цикла сканирования, получаем:

$$\text{Средняя длительность цикла с контуром ПИД-регулирования} = \left[50\ \text{мкс} + 0{,}42\ \text{мкс} + 1{,}25\ \text{мкс} + 8{,}3\ \text{мкс} \right] + 50\ \text{мс} = 50{,}06\ \text{мс}$$

Процессорные модули DL250-1 и DL260 выполняют расчет реакции ПИД-регулятора в назначенных циклах и только для контура(-ов), для которых должно быть произведено обновление (расчет) согласно установленному для них периоду опроса. Встроенный планировщик работы контуров применяет следующие правила:

- За один цикл сканирования обрабатывается столько контуров с периодами опроса ≤2 секунд, сколько нужно для обеспечения периода опроса каждого контура. Задание контуров с большими частотами опроса увеличит время цикла сканирования ПЛК.
- Используйте эту возможность только при необходимости!
- Контуры с периодами опроса > 2 секунд обрабатываются по одному или менее контуров за цикл, (с минимальной скоростью, требуемой для поддержания периода опроса каждого контура).

Процедура планирования расчета контура показана на приведенной ниже блок-схеме. Это более подробное представление содержимого задачи "Расчет реакций ПИД-контуров", содержащейся на диаграмме цикла ЦПУ. Указатель "I" соответствует медленному контуру (период опроса > 2с), а указатель "J" - быстрому контуру (период ≤ 2с). Согласно блок-схеме указатель "J" "перебирает" все указанные "быстрые" контуры от первого до последнего в пределах одного цикла. Указатель "I" увеличивается на один раз в каждом цикле и только в том случае, когда наступает время обновления следующего "медленного" контура. Таким образом, оба указателя, "I" и "J", принимают последовательно значения от 1 до наивысшего номера используемого контура, но с разной скоростью. Их совместная работа обеспечивает надлежащее обновление всех контуров.



Десять шагов к успешному управлению процессом

Современные электронные контроллеры, подобные процессору DL250-1 и DL260, обеспечивают функции управления сложными процессами. Автоматические системы управления могут с трудом поддаваться отладке, так как у конкретного симптома может быть множество различных причин. При оперативном введении новых контуров управления рекомендуется использовать осторожный, пошаговый подход

Шаг 1: знание технологии

Самое важное знание — это знать, как сделать конечное изделие. Эта информация является основой проектирования эффективной системы управления. Для составления хорошего «рецепта» нужно выполнить следующее:

- Определить все существенные переменные процесса, например, температуру, давление, расход и т.п., требующие точного управления.
- Спланировать значения заданий для всех переменных процесса в течение длительности одного производственного цикла.

Шаг 2: выбор стратегии управления контуром

Этот шаг подразумевает выбор метода, который будет использоваться установкой, чтобы обеспечить соответствие значений переменных процесса уставкам. Подобный выбор состоит из разрешения множества вопросов и компромиссов, среди которых эффективное использование электроэнергии, стоимость оборудования, возможность обслуживания в процессе производства и многое другое. Также необходимо определить, как в ходе процесса будут генерироваться значения уставок, и сможет ли оператор управления изменять эти значения вручную.

Шаг 3: определение размеров и чувствительности компонентов контура

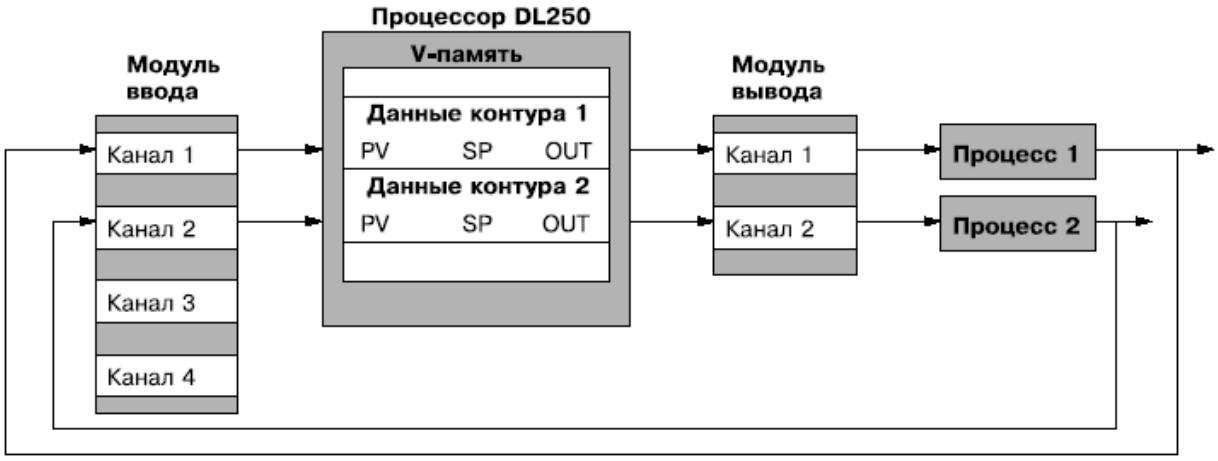
После выбора стратегии управления другим принципиальным вопросом является выбор надлежащих размеров исполнительных механизмов и шкалы датчиков.

Выберите исполнительный механизм (нагреватель, насос и т.п.), соответствующий параметрам нагрузки. Чересчур мощный исполнительный механизм после изменения задания (SP) будет оказывать на процесс слишком сильное воздействие. С другой стороны, использование исполнительного механизма с недостаточными характеристиками приведет к отставанию или возможному отклонению переменной (PV) от задания (SP) при изменении последнего или при возмущении процесса.

- Выберите датчик переменной процесса, подходящий для интересующего вас диапазона (и управления). Определите требуемое разрешение регулирования переменной процесса (например, в пределах 2°C) и убедитесь в том, что разрешающая способность датчика (на уровне младшего значащего бита) не менее чем в пять раз лучше этого значения. В то же время, слишком чувствительный датчик может привести к колебательному характеру управляющего сигнала и т.п. В DL250-1 и DL260 поддерживаются 12-битовые, 15-битовые и 16-битовые форматы данных со знаком и без знака, а так же 16-битовые данные со знаком. Выбор формата данных влияет на формат уставки (SP), переменной процесса (PV), управляющего выхода и накопленного значения в интеграторе.

Шаг 4: выбор модулей ввода/ вывода

Выбрав количество контуров, измеряемые переменные процесса и значения заданий, можно приступить к выбору подходящих модулей ввода /вывода. Обратимся к рисунку на следующей странице. Во многих случаях модули ввода или вывода, либо комбинированные модули аналогового ввода /вывода можно использовать совместно в нескольких контурах управления. В приведенном примере сигналы PV и управляющего выхода для двух контуров проходят через один и тот же набор модулей. Компания «ПЛКСистемы» предлагает аналоговые модули с 2, 4 и 8 каналами на модуль для различных типов и диапазонов сигнала. Дополнительную информацию по этим модулям можно найти в нашем каталоге, а так же на сайте: **www.plcsystems.ru.**



Шаг 5: подключение цепей и установка

После того, как все компоненты контура и модули ввода /вывода выбраны, можно выполнить проводные соединения и монтаж. Следует руководствоваться указаниями по выполнению соединений, содержащимися в Главе 2 данного Руководства, а так же в Руководстве по модулям аналогового ввода/вывода DL205. Наиболее частыми ошибками при выполнении соединений для ПИД-контуров являются следующие:

- Перепутана полярность при подключении проводов датчиков или исполнительных механизмов.
- Неправильно выполнено заземление сигнальных цепей элементов контура.

Шаг 6: выбор параметров контура

Выполнив проводные соединения и монтаж, следует выбрать параметры настройки контура. Самым простым способом запрограммировать таблицы контура является использование диалогов настройки ПИД-регулятора (PID Setup) в *Direct*SOFT. При выборе тех или иных значений важно понимать физический смысл всех параметров контура, упомянутых в данной главе.

Шаг 7: проверка работы разомкнутого контура

Подключив цепи датчиков и исполнительных механизмов и введя параметры контура, мы должны тщательно проверить новую систему управления вручную (в режиме ручного управления).

Проверьте правильность измерения переменной процесса датчиком.

- Плавно увеличивая (если это не опасно) значения управляющего выхода, начиная с нулевого значения, проследите, изменяется ли PV (и в правильном ли направлении!).

Шаг 8: настройка контура

Если проверка разомкнутого контура показала, что PV изменяется правильно, а управляющий выход соответствующим образом воздействует на процесс, можно перейти к процедуре настройки контура (автоматический режим). На этом ключевом этапе мы настраиваем контур так, чтобы PV автоматически следовало за SP.См. раздел «Настройка контура» в данной главе.

Шаг 9: выполнение цикла процесса

Если проверка замкнутого контура показала, что PV отслеживает малые изменения SP, мы можем перейти к выполнению реального цикла процесса. Теперь необходимо запрограммировать контур, чтобы генерировать требуемое значение SP в реальном времени. На этом этапе с помощью установки можно изготовить небольшую партию продукции, следя за тем, чтобы SP изменялась в соответствии с технологией.

ПРЕДУПРЕЖДЕНИЕ: *Убедитесь в возможности быстрого отключения питания и аварийной остановки, если процесс выйдет из-под контроля. Выйдя из-под контроля, некоторые процессы могут привести к поломке оборудования и/или причинения серьезного вреда персоналу.*

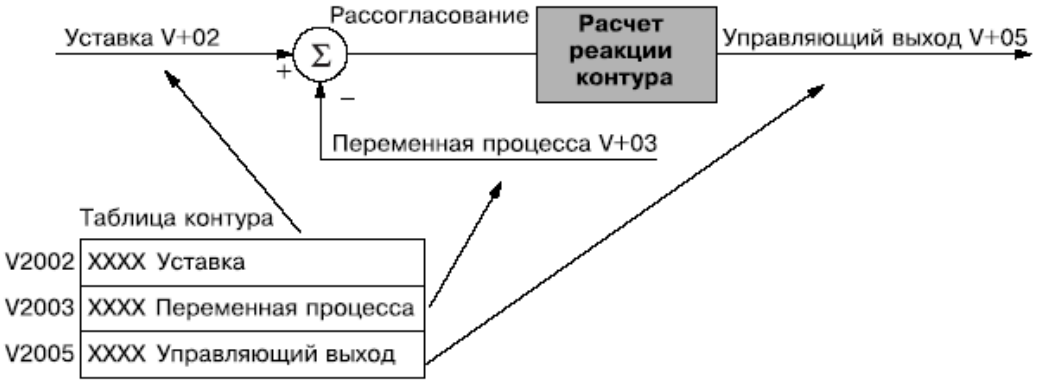
Шаг 10: сохранение параметров контура

Когда проверка и настройка контуров завершены, не забудьте сохранить все параметры контура на диск. Параметры контура являются результатом немалой работы и заслуживают бережного отношения.

Основы работы контура

Местонахождение данных

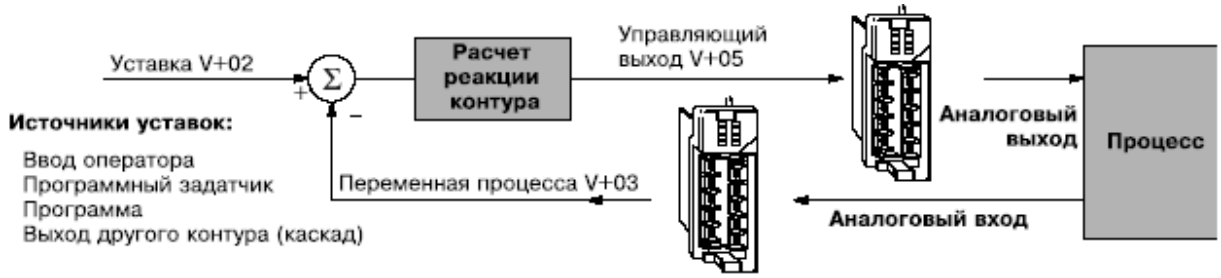
Каждый контур ПИД-регулирования полностью зависит от команд и данных, находящихся в соответствующей таблице контура. На следующем рисунке показаны ячейки таблицы контура, соответствующие трем основным переменным ввода/вывода контура: SP, PV и управляющий выход. В приведенном примере таблица контуров начинается с V2000 (произвольная ячейка, выбираемая пользователем). Уставка (задание) SP, переменная процесса PV и управляющий выход располагаются по показанным адресам.



Источники данных

Данные SP, PV и управляющего выхода должны взаимодействовать с реальными источниками и устройствами.

На приведенном ниже рисунке для каждой переменной контура показаны источники или пункты назначения. Значения управляющего выхода и переменной процесса передаются из соответствующего аналогового модуля для взаимодействия с собственно процессом. **Для копирования данных из таблицы контура в память аналогового модуля ввода/вывода и наоборот требуется небольшой фрагмент программы** на релейной логике**.** Не забывайте, что большинство аналоговых модулей мультиплексируют данные, используя два-три бита декодирования адреса канала. Примеры программ, показывающие, как выполнить обмен аналоговыми данными между аналоговыми модулями серии DL205 и произвольной ячейкой V-памяти, можно найти в Руководстве по аналоговому модулю.

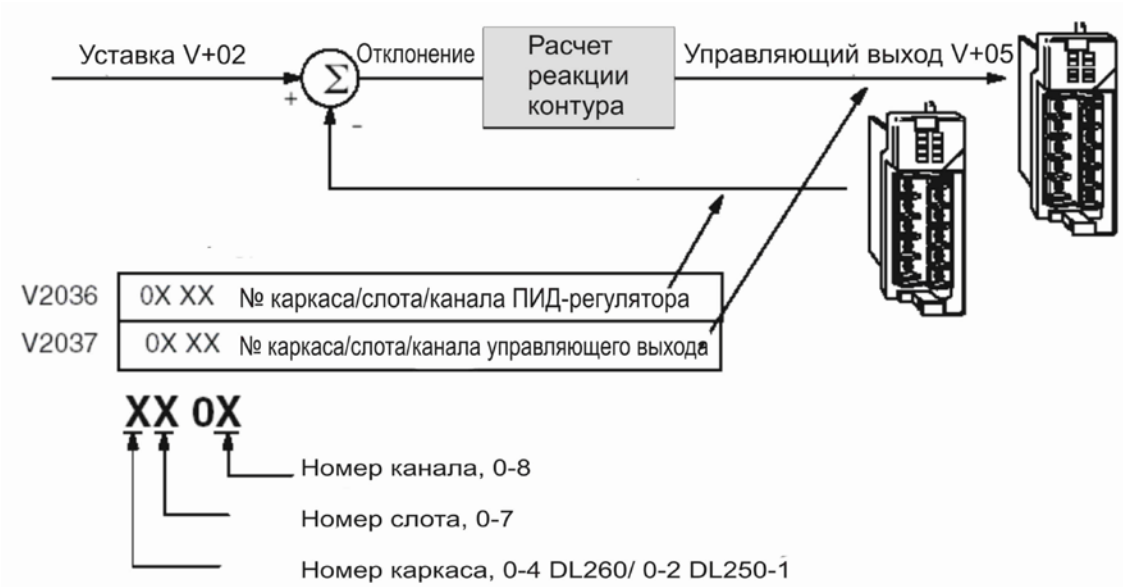


Как показано на рисунке, задание может быть введено из нескольких источников. Многие приложения в зависимости от режима работы контура в разное время будут использовать два или более источников. Кроме того, способ генерирования задания также определяется топологией контура и методом программирования. При использовании встроенного программного задатчика или при включении контуров каскадом ПИД-контроллер автоматически записывает данные задания в ячейку addr+02. Однако **при генерации задания из любого другого источника ее значение в ячейку таблицы должно записываться программно,** если только источник (человеко-машинный интерфейс) не обладает свойством непосредственной записи в ячейки V-памяти.

Очевидно, у каждого из трех основных параметров контура в любой конкретный момент времени будет только один источник или пункт назначения. Чтобы избежать ошибок, при разработке приложения неплохо нарисовать схемы контуров, показывающие источники данных и т.п.

Прямой доступ к аналоговому вводу/выводу

Контроллер контура процессора DL250-1 и DL260 обладает возможностью прямого доступа к аналоговым значениям ввода/вывода помимо сканирования программы. В частности, таким параметрами являются переменная процесса (PV) и управляющий выход. Такая возможность необходима, чтобы контроллер контура мог управлять замкнутым контуром, когда процессор находится в Программном режиме. Контроллер контура может считать аналоговое значение PV в выбранном формате данных из нужного аналогового модуля и записать значение управляющего выхода в том же формате данных в нужный модуль вывода. Эта возможность прямого доступа, если она разрешена, обеспечивает доступ к аналоговым значениям только один раз для каждого расчета реакции ПИД-регулятора для каждого соответствующего контура. Программа, однако, может одновременно получать доступ к одному и тому же вводу/выводу стандартным методом (команды RLL) или с помощью указателя, если процессор находится в Рабочем режиме. Можно дополнительно настроить каждый контур, обеспечивая доступ к его аналоговому вводу/выводу (PV и управляющий выход), изменяя соответствующие значения в соответствующих регистрах таблицы контура. На следующем рисунке показаны параметры таблицы контура с адресами addr+36 и addr+37 и их роль при прямом доступе к аналоговым значениям.



Эти параметры таблицы контура можно задавать непосредственно или, для простоты конфигурирования, с помощью соответствующего диалогового окна *Direct*SOFT. Например, значение «0102» в регистре V2036 указывает контроллеру контура считывать данные PV из слота номер 1 и второго канала. Обратите внимание, что **слот 1 — это второй слот справа от процессора**, так как слот 0 находится рядом с процессором. Значение «0000» в любом регистре сообщает контроллеру контура *о* невозможности прямого доступа к соответствующему аналоговому значению. В этом случае передачу значения между таблицей контура и модулем физического ввода/вывода должна обеспечивать программа.

Если значения PV или управляющего выхода требуют некоторой программной математической обработки, то использование функции прямого доступа контроллера контура может оказаться невозможным. В этом случае, выполнение обработки и передачи данных от аналоговых модулей или к ним может понадобиться реализовывать программно.

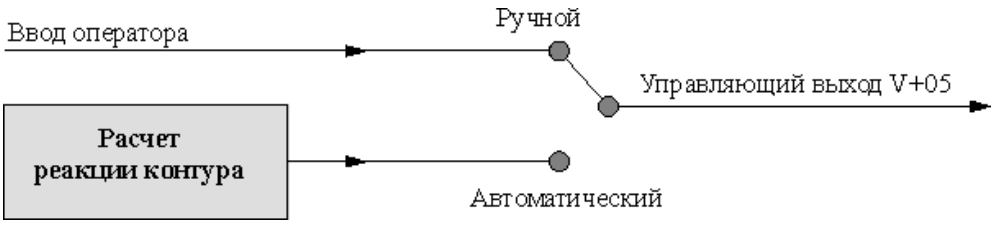
ПРИМЕЧАНИЕ: E*сли нужно, для некоторых переменных можно использовать прямой доступ к аналоговым модулям, а передачу других переменных от аналоговых модулей или к ним реализовывать программно.* ***Однако программное обеспечение контроллера контура не разрешает использовать различные методы при передаче аналоговых данных от одного аналогового модуля.***

Режимы контуров

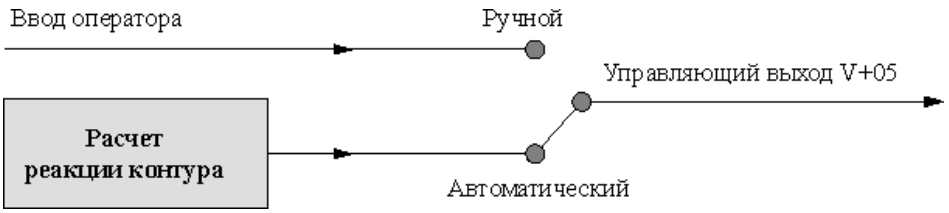
В промышленности часто возникают типовые ситуации использования контуров ПИД-регулирования. В каждом сценарии мы слегка изменяем источник данных для основных трех переменных: SP, PV и управляющего выхода, создавая имя режима для каждого сценария.

Для процессоров DL250-1 и DL260 возможны следующие режимы — Ручной, Автоматический и Каскадный. После введения в режимы мы рассмотрим, как переходить от режима к режиму.

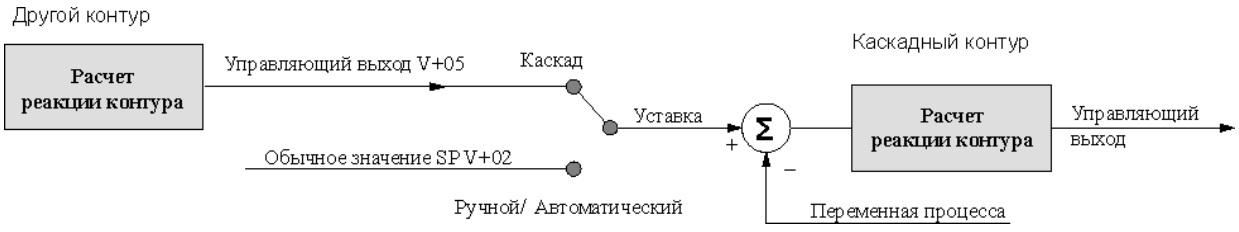
В **Ручном режиме** контур не выполняет расчет реакции ПИД-регулятора (однако, аварийные сигналы контура действуют). Для такого контура процессор прекращает записывать значения в ячейку addr+05 таблицы контура. Считается, что оператор (или другой интеллектуальный источник) вручную управляет выходом, отслеживая PV и записывая в addr+05 данные, необходимые для управления процессом. На приведенном ниже рисунке показана эквивалентная схема работы Ручного режима.



В **Автоматическом режиме** контур работает как обычно и генерирует новые значения управляющего выхода. В течение каждого периода опроса данного контура он решает уравнение ПИД-регулятора и записывает результат в ячейку addr+05. Ниже приведена эквивалентная схема работы.



В **Каскадном режиме** контур работает так же, как и в Автоматическом с одним важным отличием. Источником данных для SP в этом случае является не ячейка addr+02, а значение управляющего выхода другого контура (назначение каскадных контуров описано ниже в данной Главе). Итак, в Автоматическом или Ручном режимах для расчета контура используются данные addr+02. В каскадном режиме управляющий выход для расчета контура считывается из таблицы параметров другого контура.



То, как расчеты реакции ПИД-регулятора изменяют источники данных для Ручного/ Автоматического/Каскадного режимов, естественно приводит к возникновению определенных ограничений на изменение режимов. Как показано на рисунке ниже, контур может переходить из одного режима в другой, но **не может перейти непосредственно из Ручного режима в Каскадный.** Такое изменение режима запрещено, так как у контура одновременно менялись бы два источника данных, что может привести к потере управления.

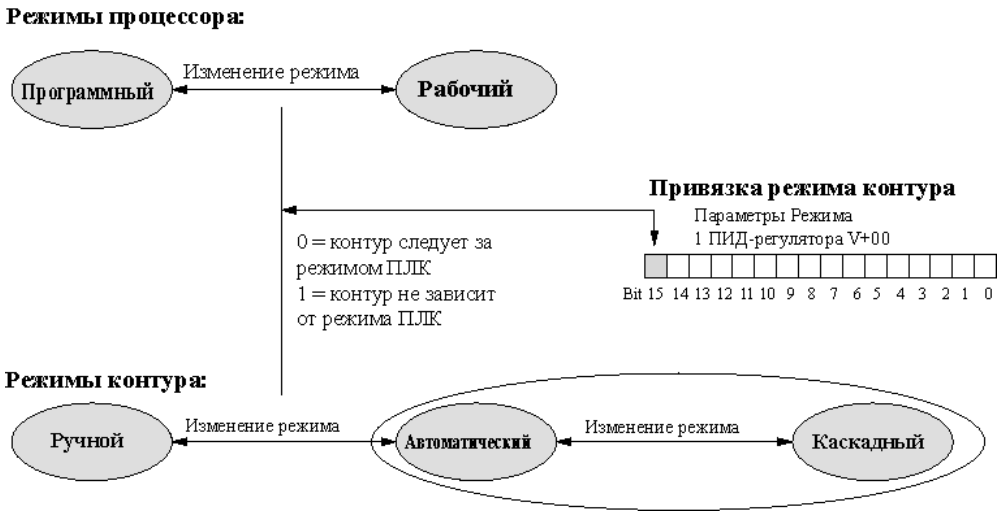
Режимы процессора и режимы контура

Очень мощным свойством контроллера контура на основе процессорных модулей DL250-1 и DL260 является возможность выполнять расчеты реакции ПИД-регулятора, когда процессор находится в Программном режиме. Обычно процессор, находящийся в Программном режиме, прекращает все операции. Однако, эти процессорные модули в Программном режиме в зависимости от конфигурации могут выполнять или не выполнять расчеты реакции ПИД-регулятора.

Возможность работать с контуром независимо от программы позволяет изменять программу во время работы процесса. Это особенно выгодно для непрерывных процессов с большой инерцией, которые трудно или дорого прерывать.

Конечно, у контуров, работающих независимо от сканирования программы, должна быть возможность прямого доступа к значениям PV и управляющего выхода в каналах аналоговых модулей. Контроллер контура обладает такой возможностью, описанной ранее в данной Главе в разделе о прямом доступе к аналоговому вводу/выводу.

Соотношения между режимами процессора и режимами контура приведены на следующем рисунке. Вертикальная пунктирная черта показывает необязательную связь между изменениями режима. Выбор определяется битом 15 Слова 1 (addr+00) настройки ПИД-регулятора. Если он равен нулю, контур следует за режимом процессора, и переход процессора в Программный режим переводит все контура в Ручной режим. Аналогично, переход процессора в режим исполнения позволит каждому из контуров вернуться в свой предыдущий режим (Ручной, Автоматический или Каскадный). В этом случае изменение режима процессора автоматически вызывает изменение режимов контуров.



Если бит 15 равен 1, контуры работают независимо от режима процессора. Это похоже на использование двух процессоров.- один исполняет программы релейной логики, а другой обеспечивает работу контуров регулирования.

ПРИМЕЧАНИЕ: *Если выбрано, что контуры работают независимо от режима процессорного модуля, то для изменения любых значений параметров таблицы контуров необходимо предпринять специальные действия. Необходимо временно заставить контуры следовать режиму процессора. В этот момент средство программирования (DirectSOFT) сможет перевести изменяемый контур в Ручной режим. После изменения значения параметра не забудьте восстановить независимую работу контура.*

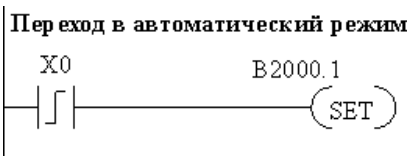
Как изменять режимы контуров

Первые три бита Слова 1 (addr+00) Настройки ПИД-регулятора запрашивают режим работы соответствующего контура. **Эти биты представляют собой запрос режима, а не команду на переход в него** (некоторые условия могут помешать изменению режима - см. ниже).

Обычным состоянием этих битов запроса является «000». Для запроса изменения режима необходимо на время одного сканирования установить соответствующий бит в «1». Контроллер контура с ПИД-регулятором автоматически вернет биты в «000» после считывания запроса на изменение режима. Существуют следующие способы запроса изменения режима:

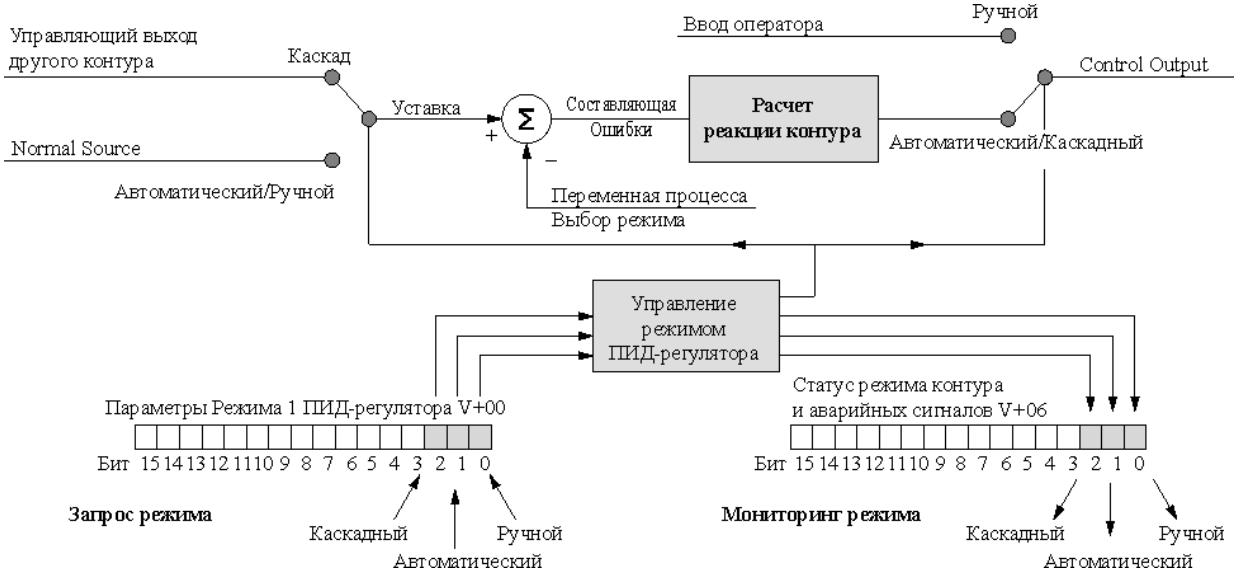
- **PID View** (просмотр ПИД-регулятора) в *Direct*SOFT – это самый простой способ. Щелкните по одной из кнопок переключателя, и *Direct*SOFT установит соответствующий бит.
- **Ручной Программатор.** Используйте режим Статус слова (WD ST) для отслеживания содержимого addr+00, которое является BCD/шестнадцатеричным значением из 4 цифр. Необходимо рассчитать и ввести новое значение для addr+00, и выполнить логическое ИЛИ для бита правильного режима и его текущего значения.
- **Программно.** Когда ПЛК находится в Рабочем режиме, программа может запросить любой режим контура. Это может понадобиться после запуска приложения.

Для установки бита режима воспользуйтесь показанным справа фрагментом программы (не используйте обмотки реле RLL). При переходе X0 из 0 ⇨ 1 цепь устанавливает бит автоматического режима = 1. Контроллер контура сбрасывает этот бит.



- **Панель оператора.** Стандартным способом свяжите панель оператора с программой, а затем воспользуйтесь описанными выше методами для установки бита режима.

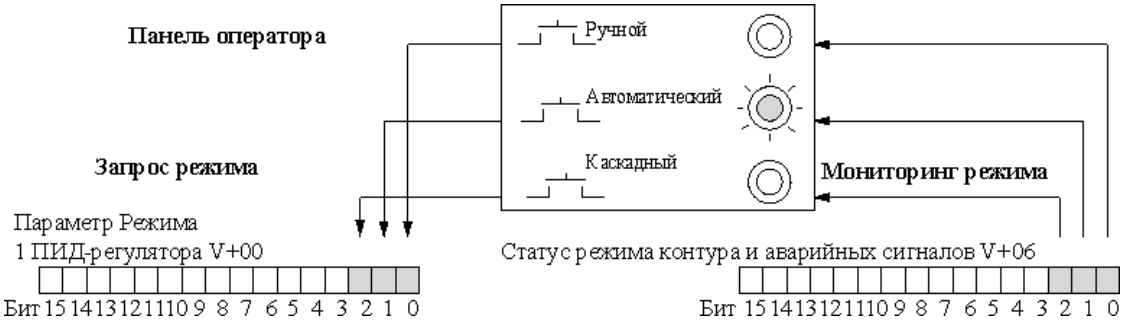
Так как изменение режима только запрашивается, контроллер контура ПИД-регулирования решает, когда разрешить изменение режима, и обеспечивает статус режима контура. Он устанавливает текущий режим в битах 0, 1 и 2 Слова Статуса режима контура и аварийных сигналов, ячейка addr+06 в таблице контура. Параллельные функции запроса / мониторинга показаны на следующем рисунке. На рисунке также показаны два зависимых от режима возможных источника задания SP и два возможных источника управляющего выхода.



Управление режимами ПИД-регулятора с панели оператора

Так как Ручной/Автоматический/Каскадный режимы являются наиболее фундаментальными и важными управляющими элементами контура с ПИД-регулятором, может потребоваться «вывести» переключатели управления режимами на операторскую панель. Для большинства приложений нужен только выбор Ручного и Автоматического режимов (Каскадный используется только в некоторых сложных приложениях). Помните, что управляющие элементы режимов в действительности являются битами запроса режима, а реальный режим контура отображается в другом месте.

На следующем рисунке показана операторская панель, использующая нефиксируемые кнопки для запроса изменения режима ПИД-регулятора. Индикаторы режима на панели не подключены к переключателям и связаны с соответствующими ячейками данных.



Влияние режимов работы ПЛК на режимы контура

Если выбрано, что контуры будут отслеживать режим ПЛК, то режимы ПЛК (Программирование, Рабочий) взаимодействуют с контурами как с группой. Вот как выглядит это взаимодействие:

- Когда ПЛК находится в Программном режиме, все контуры переводятся в Ручной режим, и расчеты контуров не производятся. Однако, обратите внимание, что в Программном режиме ПЛК отключаются модули вывода (включая аналоговые). Поэтому, если ПЛК находится в Программном режиме, действенное Ручное управление невозможно.
- Процессор позволяет изменять режим контура только, когда ПЛК находится в Рабочем режиме. По существу, процессор записывает режимы всех 16 контуров как желаемые режимы работы. Если, пока ПЛК находился в режиме исполнения, произошел сбой и восстановление питания, процессор возвращает все контуры к их предыдущему режиму (Ручному, Автоматическому или Каскадному).
- При смене Программного режима на Рабочий процессор заставляет каждый контур вернуться к своему предыдущему режиму, записанному, когда ПЛК последний раз находился в Рабочем режиме.
- Когда ПЛК находится в Программном режиме, можно добавить и конфигурировать новые контуры. Новые контуры автоматически начинают работать в Ручном режиме.

Подавление режима контура

В обычных условиях режим работы контура определяется запросом, устанавливаемым в addr+00 битами 0, 1 и 2. Однако, существуют определенные условия, препятствующие установке запрашиваемого режима:

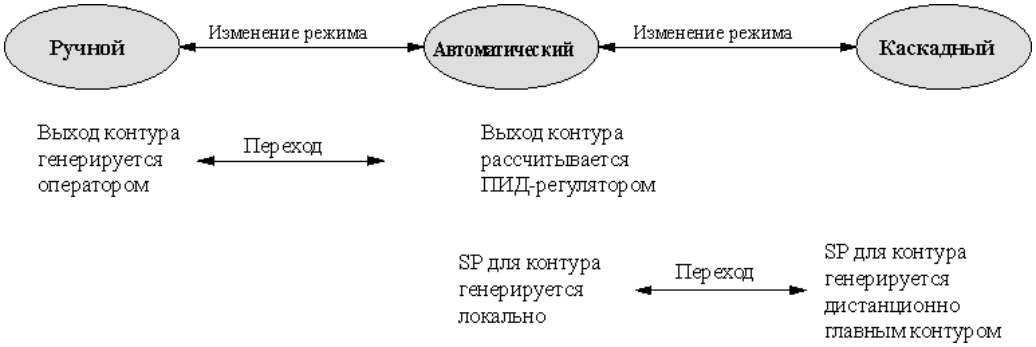
- Контур, который зависит от режима ПЛК, не может изменить режим, пока ПЛК находится в Программном режиме.
- Главный контур каскадной пары контуров не может перейти из Ручного в Автоматический режим, пока подчиненный контур находится в Каскадном режиме.

В других ситуациях контроллер контура с ПИД-регулятором автоматически изменит режим контура, обеспечивая безопасность работы:

- Контур, в котором возникло условие ошибки, автоматически переходит в Ручной режим.
- Если подчиненный контур каскадной пары по какой-то причине выходит из Каскадного режима, главный контур автоматически переходит в Ручной режим.

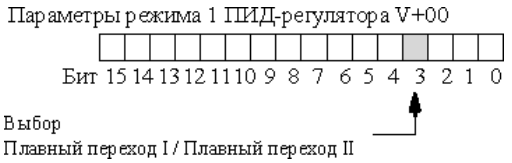
Безударные переходы

В применении к управлению процессом слово «переход» имеет конкретное значение. Переход контура возникает при изменении его режима работы, как показано ниже. Когда мы меняем режимы контура, на самом деле мы вызываем переход от одного источника управления некоторым параметром контура к другому. Например, при изменении режима контура с Ручного на Автоматический управление выходом переходит от оператора к контроллеру. При изменении режима контура с Автоматического на Каскадный управление SP переходит от первоначального источника в Автоматическом режиме к другому (главному контуру).



Основной проблемой переходов контуров является то, что у двух различных источников переходящего параметра контура будут различные численные значения. Это приведет к генерации ПИД-регулятором нежелательного скачка, или «удара», на управляющем выходе, нарушающего до некоторой степени работу контура. Функция «безударного перехода» уравнивает параметры на момент изменения режима контура, делая переход плавным (скачок на управляющем выходе отсутствует).

Существует два типа плавного перехода контроллера контура в процессорных модулях DL250-1 и DL260: Безударный переход I и Безударный переход II. Тип перехода выбирается с помощью диалогового окна PID Setup (Настройка ПИД-регулятора) *Direct*SOFT. Или можно, как показано справа, использовать бит 3 addr+00 Слова 1 настройки ПИД-регулятора.



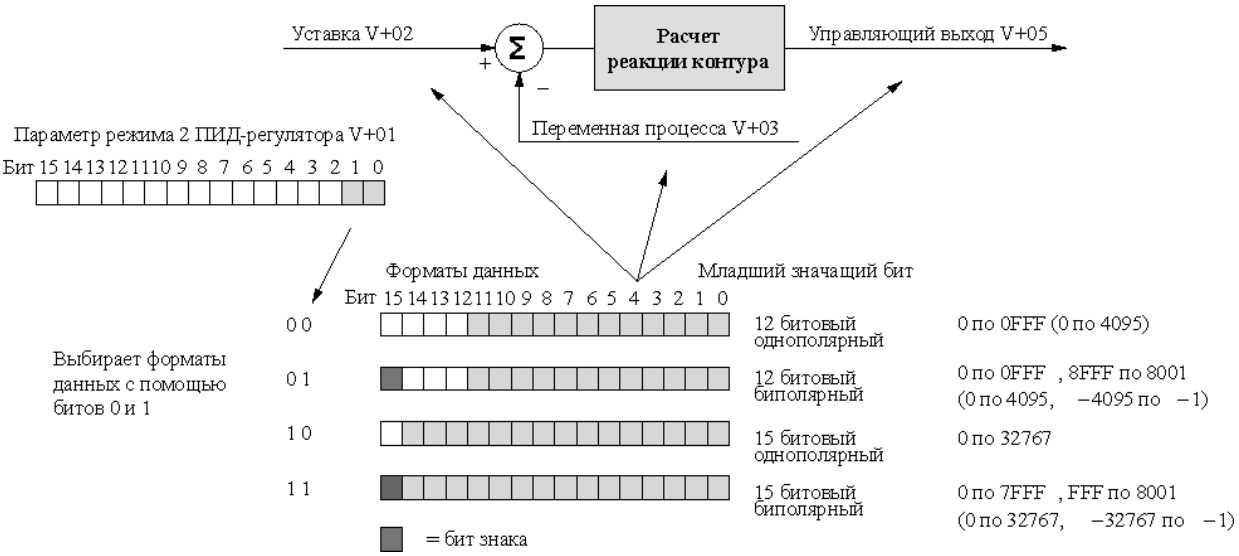
Характеристики типов Безударных переходов I и II приведены в следующей таблице. Обратите внимание, их работа также зависит от вида используемого алгоритма ПИД-регулятора (позиционный или скоростной). Заметим, что при использовании уравнения ПИД-регулятора по скорости необходимо использовать тип I Безударного перехода.

Тип перехода	Бит выбора перехода	Алгоритм ПИД-регулятора	Действия при переходе из режима Ручной в Автоматический	Действие при переходе из режима Автоматический в Каскадный
Безударный переход I	0	Позиционный	Устанавливает смещение = управляющему выходу Устанавливает SP = PV	Устанавливает выход основного контура= PV подчиненного контура
		Скоростной	Устанавливает SP = PV	Устанавливает выход основного контура = PV подчиненного цикла
Безударный переход I	1	Позиционный	Устанавливает смещение = управляющему выходу	отсутствует
		Скоростной	отсутствует	отсутствует

Конфигурирование данных ПИД- контуров

Форматы данных параметров контура

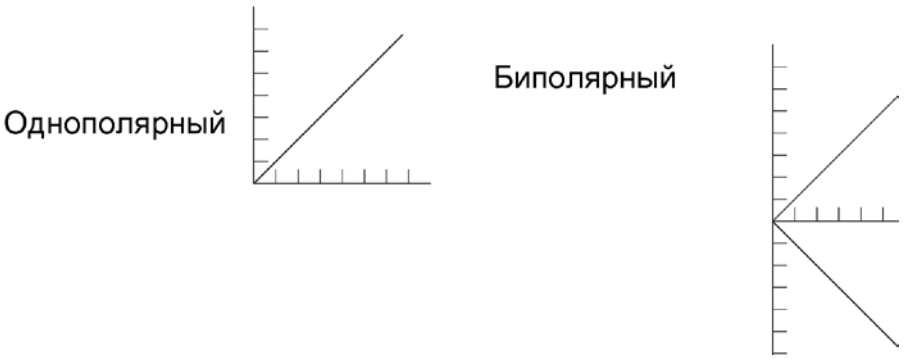
С выбором диапазона и разрешающей способности переменной процесса связан выбор формата данных трех основных переменных контура: уставки - SP, переменной процесса - PV и управляющего выхода (этот формат данных также используется смещением (интегратором) в addr+04). Четырьмя возможными форматами данных являются 12- или 15-битовый (выравнивание вправо), знаковый или беззнаковый (в биполярных форматах старший значащий бит является битом знака). Формат выбирается четырьмя двоичными комбинациями битов 0 и 1 Слова addr+01 настройки ПИД-регулятора. Диалог PID Setup (настройка ПИД-регулятора) *Direct*SOFT автоматически устанавливает эти биты при выборе формата данных в меню.



Форматы данных — это очень мощное средство, с помощью которого определяется численный интерфейс контура с ПИД-регулятором с датчиком PV и устройством управляющего выхода. Таким же должен быть и формат уставки. Обычно формат данных выбирается в ходе первоначальной настройки контура и затем не меняется.

Выбор однополярного или биполярного формата

Определение формата данных подразумевает выбор использования чисел без знака или со знаком. Большинство приложений, например, управление температурой, использует только положительные числа, следовательно, для них нужен однополярный формат. Обычно выбор однополярного/биполярного формата определяется управляющим выходом. Например, управление скоростью может включать управление в прямом и обратном направлениях. При нулевом значении задания скорости требуемый управляющий выход также равен нулю. В этом случае должен использоваться биполярный формат.



Обработка смещения данных

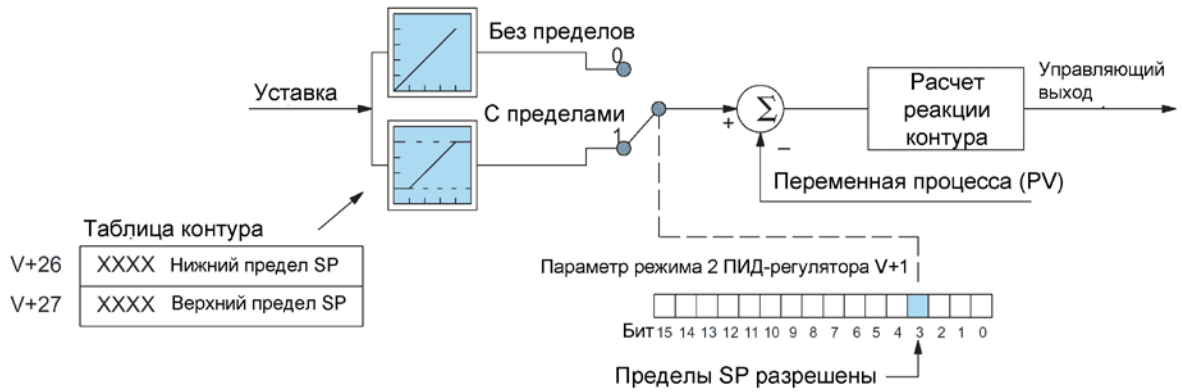
Во многих приложениях с периодическим технологическим процессом датчики или исполняющие механизмы взаимодействуют с аналоговыми модулями DL205, используя сигналы в диапазоне 4-20 мА. Этот тип сигнала обладает встроенным 20% смещением, так как нулевой точкой является 4 мА, а не 0 мА. Однако, следует помнить, что аналоговые модули преобразуют сигналы в данные *с* **одновременным устранением смещения**. Например, сигнал 4-20 мА часто преобразуется в 0000 - 0FFF (шестнадцатеричное), или 0 - 4095 (десятичное). В этом случае потребуется только выбрать 12-битовый однополярный формат данных и убедиться, что программа правильно выполняет обмен данными между таблицей контура и аналоговыми модулями.

- **Смещение PV.** Если значение PV поступает с 20% смещением, для преобразования к нулевому смещению вычтите 20% верхнего значения диапазона PV и умножьте результат на 1,25.

- **Управляющий выход.** Если значение управляющего выхода передается на устройство с 20% смещением, потребуется только, чтобы перед переходом из Ручного в Автоматический режим ваша программа записала значение, эквивалентное смещению, в регистр интегратора (addr+04). Тогда контур воспримет это значение как часть процесса, автоматически учитывая его.

Пределы уставки (SP)

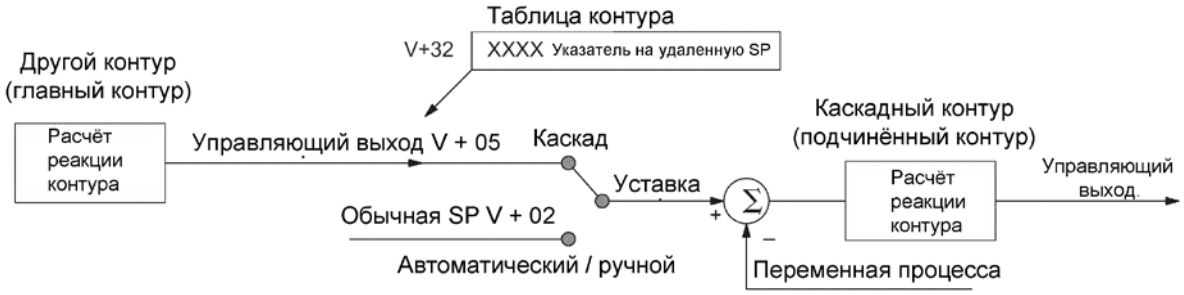
Уставка в ячейке addr+02 таблицы контура — это заданное значение переменной процесса. После выбора формата данных для этой переменной можно определить пределы диапазона значений SP, которые будут использоваться при расчете контура. Многие контура используют два или более возможных источника, в разное время записывающих значение уставки, и заданные пределы помогут защитить процесс от влияния ввода неправильного значения SP.
На следующем рисунке показана работа выбираемых пределов SP, разрешаемая битом 3 в Слове 2 (addr+01) настройки ПИД-регулятора. Если пределы включены, то ячейки addr+26 и addr+27 определяют нижний и верхний пределы SP, соответственно. Эти пределы используются при расчете реакции контура, поэтому в addr+02 можно записывать любое значение.



При расчете контура установленные верхний и нижний пределы SP проверяются перед каждым вычислением. Это означает, что программа в ходе выполнения процесса может изменить значения пределов, позволяя поддерживать узкий защитный диапазон входного значения SP.

Ячейка удаленной уставки (Remote SP)

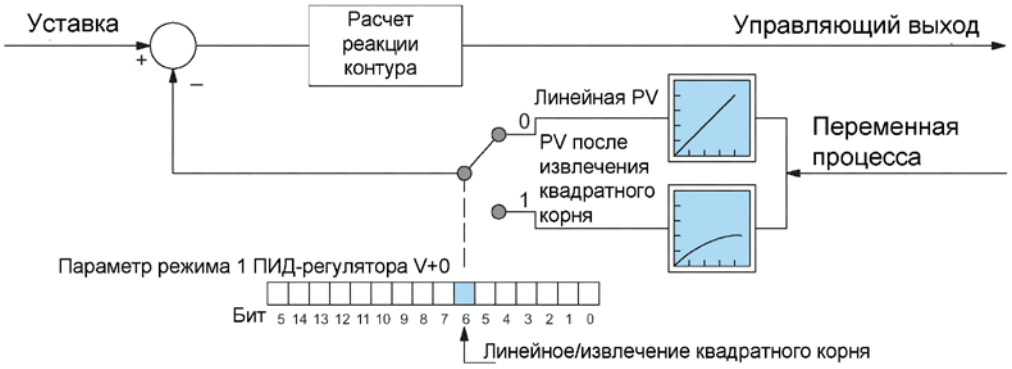
Вспомните, что в общем случае существует несколько возможных источников для значения SP. У контроллера контура с ПИД-регулятором существует встроенная возможность выбирать между двумя источниками в соответствии с текущим режимом. Взгляните на следующий рисунок. В Автоматическом или Ручном режимах контур считывает значение уставки из ячейки таблицы addr+02. Если вы собираетесь в какой-то момент использовать Каскадный режим работы контура, то вы должны задать в его таблице **указатель на удаленную уставку**.
Указатель на удаленное значение SP находится в ячейке addr+32 таблицы контура. Для контуров, которые будут управляться каскадно (являясь подчиненным контуром), понадобится записать в эту ячейку адрес управляющего выхода главного цикла. Найдите начальную ячейку таблицы параметров основного цикла и добавьте к ее адресу смещение +05.



Диалоговое окно Loop Setup (Настройка контура) *Direct*SOFT позволит задать указатель удаленной уставки (Remote SP), если известен соответствующий адрес. Можно ввести значение указателя также с помощью Ручного Программатора или задать его программно с помощью инструкции LDA.

Настройка переменной процесса (PV)

Большинство датчиков переменных процесса обеспечивают линейную зависимость параметра от сигнала. Большинство температурных датчиков линейны на всем диапазоне измерения. Однако датчики расхода, использующие измерительную диафрагму, дают сигнал, представляющий (приблизительно) квадрат расхода. Следовательно, перед использованием сигнала в линейной системе управления (например, в контуре с ПИД-регулятором) из него необходимо извлечь квадратный корень.
Есть преобразователи расхода, выполняющие извлечение квадратного корня, но они увеличивают общую цену измерительного канала. PV-вход контура с ПИД-регулятором поддерживает функцию извлечения квадратного корня, как показано ниже. Выбор между обычными (линейными) данными PV и данными, требующими извлечения квадратного корня, осуществляется с помощью бита 6 Слова 1 (addr+00) параметров режима ПИД-регулятора



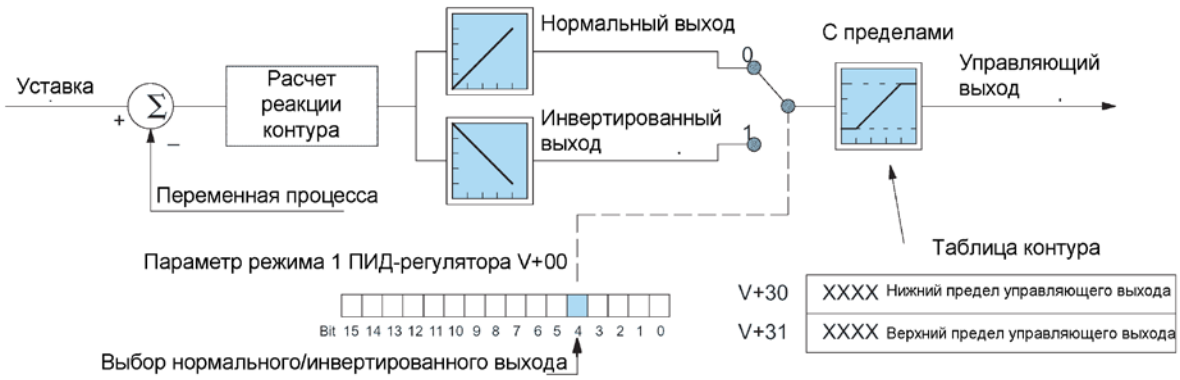
ПОДСКАЗКА: *При использовании извлечения квадратного корня PV необходимо масштабировать шкалу SP, потому что контур управляет выходом так, чтобы квадратный корень из PV равнялся входу SP. Разделите требуемое значение SP на квадратный корень из аналогового диапазона и используйте результат в ячейке dr+02 для SP. Это уменьшит разрешающую способность SP, но для большинства контуров управления потоком большая точность и не нужна (приемник потока интегрирует ошибки). Воспользуйтесь одной из следующих формул для SP в соответствии с используемым форматом данных. Неплохо будет установить верхний предел SP равным максимальному из допустимых значений.*

Формат данных	Масштабирование SP	Диапазон SP	Диапазон PV
12-битовый	SP = вход PV / 64	0 - 64	0 - 4095
15-битовый	SP = вход PV/181	0 - 181	0 - 32767
16-битовый	SP = вход PV/256	0 - 256	0 - 65535

Настройка управляющего выхода

Управляющий выход — это численный результат расчета реакции ПИД-регулятора. Выбор всех остальных параметров в конце концов влияет на значение управляющего выхода контура для каждого расчета. Ниже показаны доступные варианты конечной обработки управляющего выхода. Окончательный выход (у правого края рисунка) может быть ограничен заданными при настройке нижним и верхним пределами. Значения для addr+30 и addr+31 могут быть установлены один раз с помощью диалогового окна PID Setup (настройка ПИД-регулятора) *Direct*SOFT.

Верхний и нижний пределы управляющего выхода могут помочь предотвратить выдачу чрезмерного сигнала управления, из-за ошибки или сбоя работы контура (например, при потере сигнала датчика PV). Однако, не применяйте эти пределы для ограничения механического движения, которое иначе может повредить механизм (вместо этого воспользуйтесь концевыми выключателями).



Другим доступным вариантом является выбор нормального/инвертированного выхода (называемого в *Direct*SOFT «forward/reverse» — прямой/обратный). Для конфигурирования выхода используется бит 4 Слова 1 (addr +00) параметров режима ПИД-регулятора. Независимо от используемого формата (однополяр-ный/биполярный) нормальный выход увеличивается при положительном рассогласовании и уменьшается при отрицательном (где рассогласование (ошибка) = (SP-PV)). Инвертированный выход меняет направление изменения выхода.

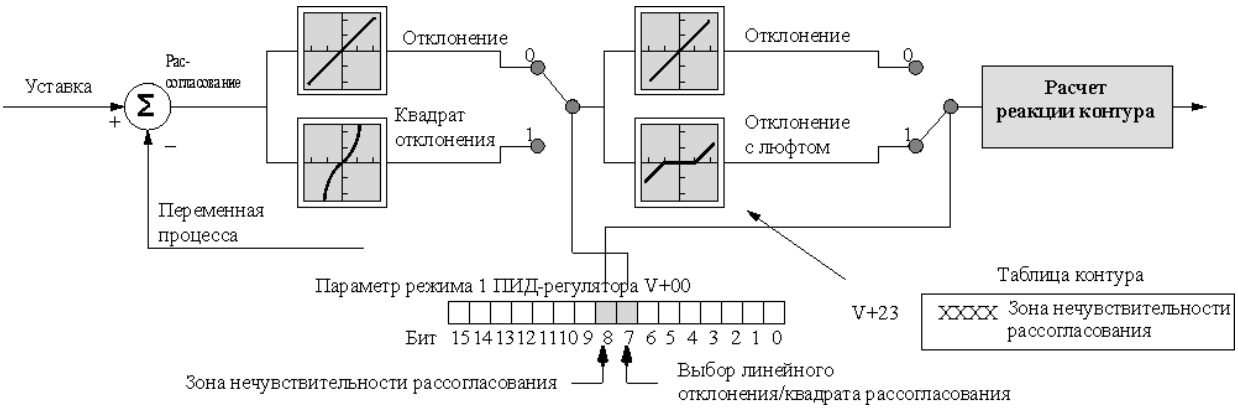
Выбор нормального/инвертированного выхода используется для настройки контуров с прямым/обратным действием. Этот выбор, в итоге, определяется направлением отклика переменной процесса на изменение управляющего выхода в определенном направлении. Подробно контуры с прямым/обратным действием описаны в разделе «Алгоритмы ПИД-регулятора».

Конфигурирование рассогласования

Рассогласование (Error) является внутренним значением контроллера контура с ПИД-регулятором и генерируется при каждом расчете реакции ПИД-регулятора.

Хотя значение рассогласования не является доступным непосредственно, его можно легко вычислить с помощью формулы: **Рассогласование = SP- PV**. Если включено извлечение квадратного корня из PV, то **Рассогласование = SP- (sqrt PV)**. В любом случае величина и алгебраический знак ошибки определяют последующее изменение управляющего выхода для каждого расчета реакции ПИД-регулятора.
Теперь мы наложим на рассогласование описанные ниже «специальные эффекты» (см. рисунок). Бит 7 Слова 1 (addr+00) настройки ПИД-регулятора позволяет выбрать линейную или квадратичную составляющую от рассогласования, а бит 8 включает или отключает зону нечувствительности рассогласования.

ПРИМЕЧАНИЕ: *При первом конфигурировании контура лучше всего использовать стандартное рассогласование. После подстройки контура можно будет сказать, улучшают ли эти функции управление.*



Квадрат отклонения. При выборе этого режима рассогласование просто возводится в квадрат (первоначальный алгебраический знак сохраняется), который и используется при вычислениях. Это влияет на управляющий выход, уменьшая его отклик на малые значения рассогласования, но сохраняя отклик на большие ошибки. Возведение в квадрат рассогласования может быть полезным, например, в следующих ситуациях:

- **Зашумленный сигнал PV.** Возведение рассогласования в квадрат может уменьшить эффект воздействия на PV низкочастотного электрического шума, который вызывает дрожание системы управления. Возведение рассогласования в квадрат сохраняет реакцию на большие отклонения.

- **Нелинейный процесс.** Для некоторых процессов (например, управление pH в химическом производстве) лучшие результаты дают нелинейные контроллеры. Другим приложением является управление промежуточным резервуаром, для которого требуется сглаженный сигнал управляющего выхода.

Зона нечувствительности рассогласования. При выборе этого режима функция зоны нечувствительности рассогласования просто подставляет ноль вместо значения отклонения, если оно не выходит за пределы заданного диапазона вблизи нуля. Если рассогласование выходит за границы зоны нечувствительности, то его значение используется как обычно.

Требуемое значение диапазона зоны нечувствительности должно быть задано в ячейке параметров контура addr+23. Единицы этого значения совпадают с единицами SP и PV (от 0 до FFF в 12-битовом режиме, и от 0 до 7FFF в 15-битовом режиме). Контроллер контура ПИД-регулирования автоматически симметрично использует зону нечувствительности относительно нулевого рассогласования.

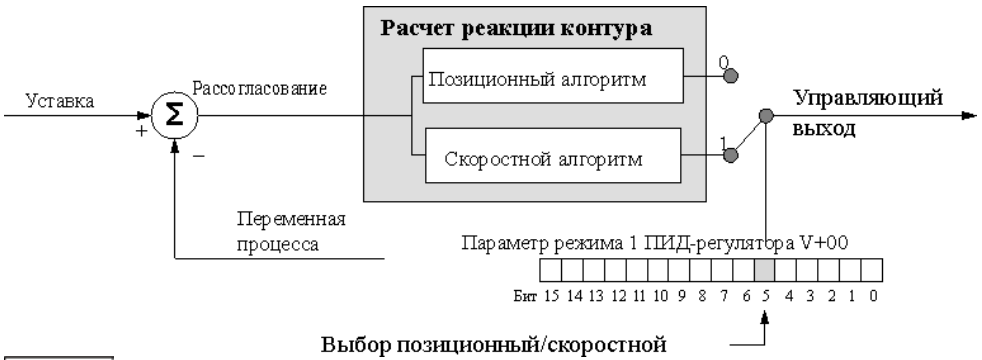
Алгоритмы ПИД-регулирования

Пропорционально-интегрально-дифференциальный (ПИД) алгоритм регулирования широко используется в управлении процессами. ПИД-метод управления хорошо адаптируется к электронным устройствам, он реализован как на аналоговых, так и цифровых (процессорных) компонентах. Процессорные модули DL250-1 и DL260 реализуют уравнение ПИД-регулирования цифровым образом, программным способом решая основное уравнение. Модули ввода/вывода служат только для преобразования электронных сигналов в цифровую форму (или наоборот).

Процессорные модули обеспечивают два типа ПИД-управления: «позиционный» и «скоростной». Эти термины обычно относятся к управлению движением, но мы будем использовать их в другом смысле:

- **Позиционный ПИД-алгоритм.** Управляющий выход рассчитывается так, чтобы он реагировал на смещение (позицию) PV относительно SP (рассогласование).
- **Скоростной ПИД-алгоритм (или алгоритм в приращениях).** Управляющий выход рассчитывается так, чтобы он представлял скорость изменения значения PV, с которой переменная процесса (PV) стремится стать равной значению уставки SP.

Огромное множество приложений будет использовать позиционную форму ПИД-уравнения. Если вы не уверены в том, какой алгоритм использовать, сначала попробуйте позиционный алгоритм. Для выбора нужного алгоритма воспользуйтесь диалоговым окном PID View Setup (Настройка вида ПИД-регулятора) в *Direct*SOFT. Или используйте для выбора алгоритма, как показано ниже, бит 5 Слова 1 (addr+00) параметров настройки ПИД-регулятора.



ПРИМЕЧАНИЕ: *Выбор типа алгоритма ПИД-регулирования является принципиальным моментом работы контура управления и обычно никогда не меняется после первоначального конфигурирования контура.*

Позиционный алгоритм

При позиционном алгоритме ПИД-уравнение вычисляет управляющий выход Мп:

$$M_n = K_c * e_n + K_i * \sum_{i=1}^{n} e_i + K_r * (e_n - e_{n-1}) + M_o$$

В приведенной выше формуле сумма интегральной составляющей и начального значения выхода объединяются в член «смещение», Мх (в установившемся режиме его значение определяет рабочую точку). С помощью члена смещения мы определяем формулы для смещения и управляющего выхода как функцию времени опроса:

$$Mx_0 = M_0$$
$$Mx_n = K_i * e_n + Mx_{n-1}$$

$$M_n = K_i * \sum_{i=1}^{n} e_i + M_0$$

$M_n = K_c * e_n + K_r * (e_n - e_{n-1}) + Mx_n$.....Выход для времени опроса “n”

Ниже перечислены переменные позиционного алгоритма и связанные с ним переменные:

T_s = период опроса

K_c = коэффициент пропорционального усиления

$K_i = K_c * (T_s/T_i)$ коэффициент интегральной составляющей

$K_r = K_c * (T_d/T_s)$ коэффициент дифференциальной составляющей

T_i = время интегрирования

T_d = время дифференцирования

SP_n = уставка для опроса «n» (значение SP)

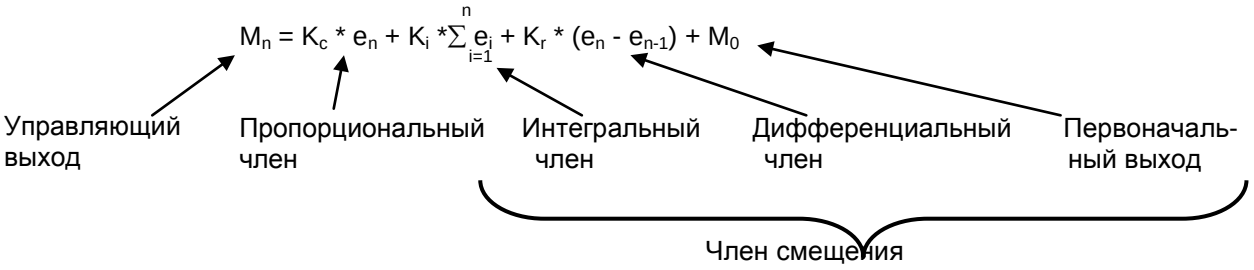
PV_n = переменная процесса для опроса «n» (PV)

$e_n = SP_n - PV_n$ = рассогласование для отсчета «n»

M_0 = управляющий выход для опроса «0»

M_n = управляющий выход для опроса «n»

Анализ этих уравнений можно найти в большинстве хороших книг по управлению процессами. С первого взгляда мы можем выделить части позиционного ПИД-алгоритма, соответствующие П-, И- и Д-членам, а также смещению (рабочей точке) так, как это показано на рисунке ниже.



Первоначальный выход — это значение выхода, получаемое контуром из Ручного режима управления при переходе в Автоматический режим. Сумма первоначального выхода и интегральной составляющей является членом смещения, который удерживает «положение » выхода. Соответственно, у скоростного алгоритма, рассматриваемого ниже, компонент смещения отсутствует.

Скоростной алгоритм

ПИД-уравнение для скоростного алгоритма может быть получено путем преобразования формулы позиционного алгоритма вычитанием уравнения степени (n-1) из уравнения степени n.

Ниже перечислены переменные алгоритма по скорости и связанные с ним переменные:

T_s = период опроса

K_c = коэффициент пропорционального усиления

$K_i = K_c * (T_s/T_i)$ коэффициент интегральной составляющей

$K_r = K_c * (T_d/T_s)$ коэффициент дифференциальной составляющей

T_i = время интегрирования

T_d = время дифференцирования

SP_n = задание для опроса «n» (значение SP)

PV_n = переменная процесса для опроса «n» (PV)

$e_n = SP_n - PV_n$ = рассогласование для отсчета «n»

M_n = управляющий выход для опроса «n»

Окончательные уравнения для алгоритма скоростного ПИД-уравнения выглядят так:

$\Delta M_n = M_n - M_{n-1}$

$\Delta M_n = K_c * (e_n - e_{n-1}) + K_i * e_n + K_r * (e_n - 2*e_{n-1} + e_{n-2})$

Контуры с прямым и обратным действием

Коэффициент усиления процесса определяет, в частности, как процесс должен управляться. Процесс, показанный на следующем рисунке, обладает положительным коэффициентом усиления, и контур регулирования в таком процессе мы называем контуром с «прямым действием». Это означает, что при росте управляющего выхода переменная процесса в итоге также растет. Конечно, настоящий процесс обычно обладает сложной передаточной функцией, включающей временные задержки. Здесь нас интересует только направление изменения переменной процесса в ответ на изменение управляющего выхода.

В большинстве случаев контуры процессов, например, температурные контуры регулирования, будут контурами с прямым действием. Рост подаваемого тепла приводит к росту PV (температуры). В соответствии с этим, контуры с прямым действием иногда называются контурами нагрева.

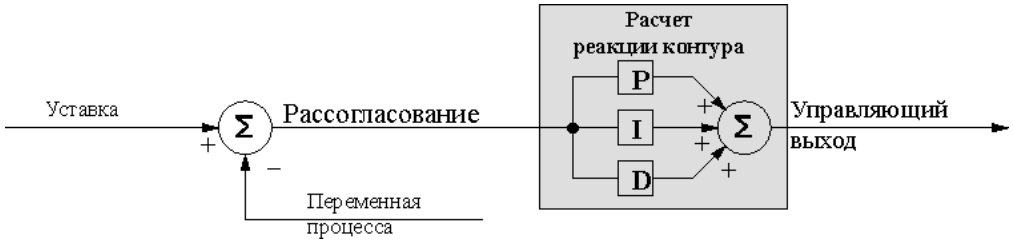
В контуре с обратным действием, как показано ниже, коэффициент усиления процесса отрицателен. Увеличение управляющего выхода приводит к уменьшению PV. Такие контуры обычно встречаются при управлении охлаждением, когда рост охлаждающего входа приводит к уменьшению PV (температуры). В соответствии с этим, контуры с обратным действием иногда называются контурами охлаждения.

Для конкретного контура важно знать тип его действия (прямое или обратное)! Если объект управления не является температурой, ответ не очевиден. Для контура управления потоком цепь позиционирования клапана может быть легко подключена и настроена как с прямым, так и с обратным действием. Простым способом найти направление действия является запуск контура в Ручном режиме, в котором придется управлять клапанами управляющего выхода вручную. Проследите, будет ли PV увеличиваться или уменьшаться в ответ на пошаговое увеличение управляющего выхода.

Для запуска контура в Автоматическом или Каскадном режиме управляющий выход должен быть правильно запрограммирован (настройка управляющего выхода описана в предыдущем разделе). Для контуров с прямым действием используйте «нормальный выход», а для контуров с обратным действием — «инвертированный выход». Чтобы сбалансировать контур с обратным действием контроллер ПИД-регулятора должен знать, что управляющий выход необходимо инвертировать. При наличии выбора сконфигурируйте и подключите контур как контур с прямым действием. Так будет проще просматривать и интерпретировать данные контура при его настройки.

П-, И- и Д-составляющие контура регулирования

В уравнениях, описывающих поведения контура регулирования, как по отклонению, так и по скорости, обычно представлены три составляющих расчета реакции ПИД-регулятора. На следующем рисунке представлена схема расчета реакции ПИД-регулятора, в которой управляющий выход является суммой пропорциональной, интегральной и дифференциальной составляющих. При каждом расчете реакции контура для каждой составляющей используется одно и то же значение сигнала рассогласования (Error).

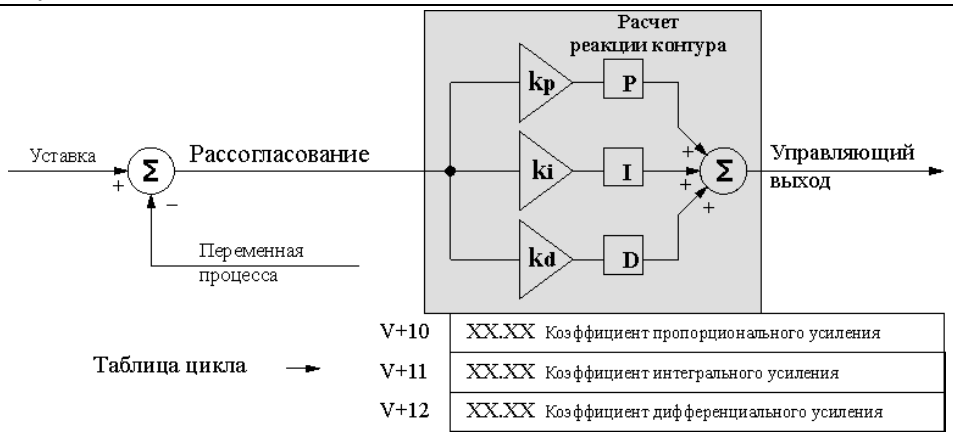


П-, И- и Д- составляющие при управлении выполняют следующие функции:

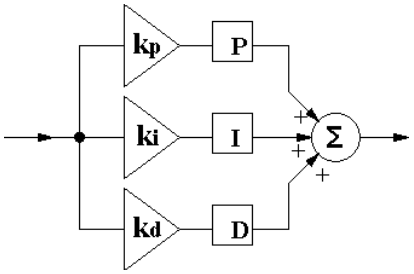
- **Пропорциональная.** Пропорциональная составляющая просто обеспечивает отклик, пропорциональный текущей величине рассогласования. Контроллер контура вычисляет значение пропорциональной составляющей при каждом расчете реакции ПИД-регулятора. Когда рассогласование равно нулю, пропорциональная составляющая также равна нулю.
- **Интегральная.** Интегрирующая (Reset) составляющая интегрирует (суммирует) значения рассогласования. Начиная с первого расчета реакции ПИД-регулятора при переходе в автоматический режим, интегратор постоянно хранит промежуточную сумму значений рассогласований. Для позиционной формы уравнения ПИД-регулятора, когда контур достигает равновесия и рассогласование отсутствует, промежуточная сумма представляет собой постоянное значение на выходе - смещение, необходимое для поддержания текущего значения PV.
- **Дифференциальная.** Дифференциальная (Rate) составляющая реагирует на изменение значения текущего рассогласования по сравнению с рассогласованием, используемой при предыдущем расчете реакции ПИД-регулятора. Ее задача состоит в том, чтобы предвидеть возможный рост рассогласования и заранее вносить соответствующий вклад в управляющий выход.

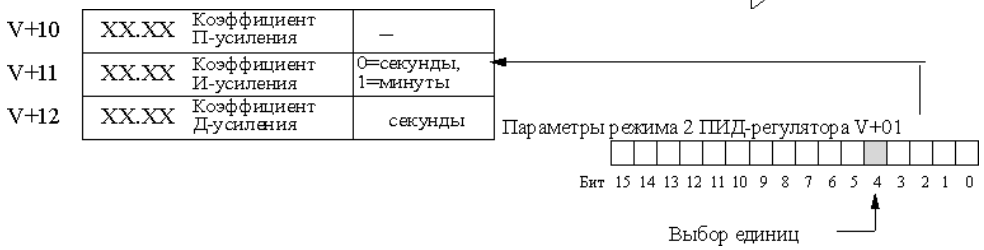
П-, И- и Д- составляющие работают вместе как одна команда. Для повышения их эффективности потребуются некоторые дополнительные инструкции. На следующем рисунке П-, И- и Д- составляющие содержат настраиваемые значения **коэффициентов усиления:** kp, ki и kd, соответственно. Эти значения размещаются в таблице контура в соответствующих ячейках. Цель процесса настройки контура (описываемого ниже) состоит в том, чтобы получить значения коэффициентов усиления, приводящие к хорошей общей эффективности контура.

ПРИМЕЧАНИЕ: *Коэффициент пропорционального усиления в терминологии контура с ПИД-регулятором также называется просто «коэффициентом усиления (Gain).*



Коэффициенты П-, И- и Д- усиления — это 4 разрядные числа в формате BCD со значениями от 0000 до 9999. В середине они содержат предполагаемую десятичную точку, поэтому действительные значения лежат в диапазоне от 00.00 до 99.99. Для некоторых коэффициентов усиления задаются единицы измерения — коэффициент интегрального усиления может задаваться в секундах или минутах с помощью показанного ниже бита. Коэффициент дифференциального усиления задается в секундах.





В окне просмотра тренда пакета программирования *Direct*SOFT можно задать значения коэффициентов усиления и единиц в реальном времени, это обычно делается во время настройки контура регулирования.

Пропорциональный коэффициент усиления. Это основной из трех коэффициентов. Диапазон значений от 0000 до 9999, но внутри значения используются как xx.xx. Задание «0000» удаляет пропорциональную составляющую из уравнения ПИД-регулятора. Это позволяет подстроиться под приложения, для которых требуются только контуры с интегральным регулированием.

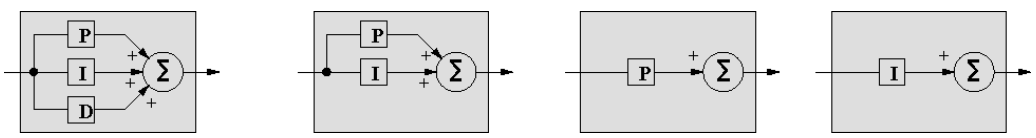
Интегральный коэффициент усиления (Время интегрирования). Диапазон значений от 0001 до 9998, но внутри значения используются как xx.xx. Задание «0000» или «9999»обнуляет коэффициент интегрального усиления, удаляя интегральную составляющую из уравнения ПИД-регулятора. Это позволяет подстроиться под приложения, для которых требуются только контуры с пропорциональным регулированием. Единицы коэффициента интегрального усиления могут быть или секундами или минутами, как показано на верхнем рисунке.

Дифференциальный коэффициент усиления (Время дифференцирования). Диапазон значений от 0001 до 9999, но внутри значения используются как xx.xx. Задание «0000» позволяет удалить дифференциальную составляющую из уравнения ПИД-регулятора (обычная практика). Это позволяет подстроиться под приложения, для которых требуются только контуры с пропорциональным и/или интегральным регулированием. Имеется необязательная возможность ограничения коэффициента дифференциального усиления, обсуждаемая в следующем разделе.

ПРИМЕЧАНИЕ: *Очень важно знать, как правильно увеличивать и уменьшать коэффициенты усиления. коэффициенты пропорционального и дифференциального усиления ведут себя, как и можно было ожидать: маленькие числа означают маленькое усиление, а большие — большое. Однако интегральная составляющая включает обратное значение (1/Ts), поэтому меньшие числа приводят к большему усилению, а большие числа — к меньшему усилению. Это очень важно помнить при настройке контура.*

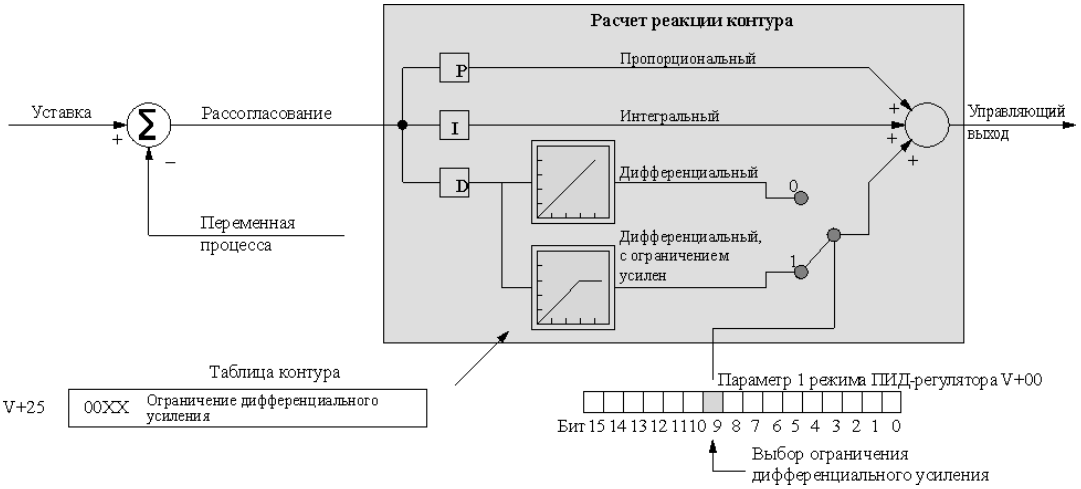
Использование подмножества управляющих элементов ПИД-регулятора

Для каждого из коэффициентов П-, И- и Д- усиления существует значение, удаляющее этот член из уравнения ПИД-регулятора. Многие приложения в действительности лучше работают под управлением подмножества управляющих элементов ПИД-регулятора. На следующем рисунке показаны различные комбинации управляющих элементов ПИД-регулятора, реализуемые в DL250-1 и DL260. Мы не советуем пользоваться другими комбинациями управляющих элементов, так как большинству подобных комбинаций присуща неустойчивость.



Ограничение дифференциального коэффициента усиления

Дифференциальный член уравнения отличается тем, что имеет дополнительную возможность ограничения усиления. Это обусловлено тем, что дифференциальная составляющая сильно реагирует на шум сигнала PV или другие случаи неожиданных пульсаций PV. Функция ограничения усиления показана на приведенном ниже рисунке. Для включения ограничения усиления используется бит 9 Слова 1 (addr +00) настройки ПИД-регулятора.



Ограничение дифференциального усиления в ячейке addr+25 должно иметь значение от 0 до 20, в формате BCD. Эта настройка работает, только если бит разрешения = 1.

Ограничение усиления может быть особенно полезно при настройке контура. Большинство контуров могут допускать, без возникновения неуправляемых колебаний, только небольшие значения дифференциального усиления.

Составляющая смещения

В широко используемой позиционной форме уравнения ПИД-регулятора (по отклонению) важным компонентом значения управляющего выхода является составляющая смещения, показанная ниже. В таблице контура она находится в ячейке addr+04. Контроллер контура записывает новую составляющую смещения после каждого расчета реакции контура.

$$M_n = K_c * e_n + K_i * \sum_{i=1}^{n} e_i + K_r * (e_n - e_{n-1}) + M_o$$

Управляющий выход — Пропорциональный член — Интегральный член — Дифференциальный член — Первоначальный выход

V+04 XXXX Член смещения

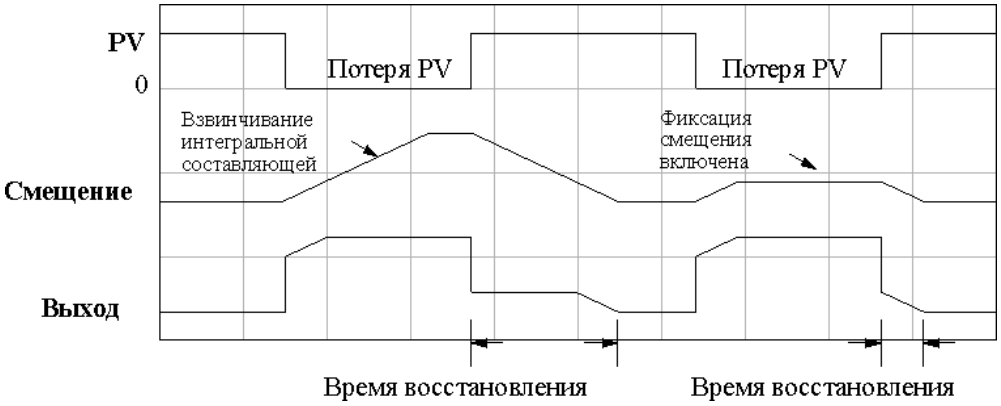
Член смещения

Если мы для двух или более периодов отсчетов сводим отклонение (e_n) к нулю, пропорциональная и дифференциальная составляющие обнуляются. Составляющая смещения равна сумме интегральной составляющей и начального выхода (Mo). Она представляет собой устойчивую, постоянную часть значения управляющего выхода и аналогична компоненте постоянного тока комплексного волнового представления сигнала.

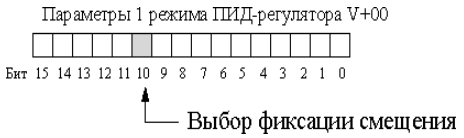
Значение составляющей смещения задает «рабочую точку» управляющего выхода. **Когда рассогласование колеблется вокруг нулевого значения, выход колеблется вокруг значения смещения.** Это понятие очень важно, так как оно показывает, почему интегральная составляющая должна медленнее откликаться на рассогласование, чем пропорциональная или дифференциальная составляющие.

Фиксация смещения

Термин «взвинчивание интегральной составляющей» или «интегральная яма» (reset windup) относится к нежелательным характеристикам поведения интегратора, естественного при определенных условиях (см. следующий рисунок). Предположим, что сигнал PV пропадает, и значение PV становится равным нулю. Последствия этого серьезного сбоя: управляющий выход контура увеличивается за счет взвинчивания интегральной составляющей. Обратите внимание, что составляющая смещения (интегральная) продолжает при потере сигнала PV интегрироваться как обычно, пока не будет достигнут ее верхний предел. При восстановлении сигнала PV значение смещения насыщается, и для возвращения в нормальное состояние требуется длительное время. Следовательно, время восстановления увеличивается. До восстановления выходной уровень остается неправильным, что приводит к дальнейшим проблемам.



При второй потере сигнала PV на рисунке включена функция фиксации смещения. Это приводит к замораживанию значения смещения, когда управляющий выход превосходит некоторые пределы. Большей части взвинчивания интегральной составляющей таким образом удается избежать, и время восстановления выхода становится намного меньше.



В большинстве приложений функция фиксации смещения будет работать с контуром, как описано выше. Ее можно включить при помощи диалогового окна PID View Setup (Настройка просмотра ПИД-регулятора) в *Direct*SOFT, или устанавливая бит 10 Слова 1 (addr+00) настройки ПИД-регулятора, как показано справа.

ПРИМЕЧАНИЕ: *Функция фиксации смещения прекращает изменение составляющей смещения, когда управляющий выход достигает границы диапазона данных. Если для управляющего выхода заданы пределы, отличные от границ диапазона (например, 0-4095 для однополярного/12-битового контура), составляющая смещения в качестве точки останова продолжит использовать границы диапазона, и фиксация смещения не будет работать.*

В методе управления с предварением, обсуждаемом ниже в данной главе, программа непосредственно записывает значение составляющей смещения. Однако это не приводит к конфликту с фиксацией смещения, так как случаи записи составляющей смещения из-за предварения относительно редки.

Процедура настройки контура

Это важный этап управления процессом в замкнутом контуре. Цель настройки контура состоит в том, чтобы настроить коэффициенты усиления так, чтобы качество работы контура в динамическом режиме было оптимальным. О качестве работы контура обычно можно судить по тому, насколько хорошо PV следует за SP после ступенчатого изменения SP.

Автоматическая настройка или ручная. Изменять значения коэффициентов усиления ПИД-регулятора можно непосредственно (ручная настройка), или можно использовать в процессоре механизм обработки ПИД-регулятора, автоматически рассчитывающий коэффициенты усиления (автоматическая настройка). У каждого опытного инженера есть свой любимый метод, и процессорный модуль может подстроиться под любые запросы. Использование автоматической настройки поможет устранить большую часть ошибок ручного подхода, который использует метод проб и ошибок, особенно для тех, чей опыт настройки невелик. Однако обратите внимание, что применение процедуры автоматической настройки даст значения коэффициентов, близкие к оптимальным, а дополнительная ручная подстройка может сделать значения коэффициентов равными оптимальным.

Если PV быстро и неустойчиво колеблется при автонастройке, это свидетельствует о том, что параметры контура регулирования выбраны неправильно. Для предотвращения влияния шума на характеристики контура необходимо использовать встроенный аналоговой фильтр для PV или фильтр для PV на релейной логике. Когда контур будет правильно настроен, то фильтр для PV можно выключить.

ПРЕДУПРЕЖДЕНИЕ: *Вносить изменения, затрагивающие константы настройки контура, имеет право только авторизованный персонал, полностью ознакомленный со всеми аспектами процесса. Применение процедуры автоматической настройки контура окажет влияние на процесс, включая значительные изменения значения управляющего выхода. Убедитесь, что вы тщательно изучили последствия любых изменений, чтобы минимизировать риск травмирования персонала или повреждения аппаратуры. Автоматическая настройка в DL250-1 и DL260 не сможет заменить изучение процесса.*

Тестирование разомкнутого контура

Используется ли ручная или автоматическая настройка, очень важно проверить основные характеристики нового процесса до его настройки. Для каждого нового контура проверьте в ручном режиме следующие пункты.

- **Уставка.** Убедитесь, что источник, который должен генерировать уставки может это делать. Можно перевести ПЛК в рабочий режим, но оставьте контур в ручном режиме. Затем следите за ячейкой addr+02 таблицы контура, чтобы видеть значение (значения) SP. В этот момент также необходимо проверить программный задатчик (если он используется).

- **Переменная процесса.** Убедитесь, что значение PV измеряется точно, и что значения PV, попадающие в ячейку addr +03 таблицы контура, правильны. Если сигнал PV сильно зашумлен, подумайте об установке входного фильтра аппаратного (низкочастотного RC-фильтра) либо цифрового программного.

- **Управляющий выход.** Если это можно сделать безопасно, вручную измените выход на небольшое значение (например, 10%) и проследите влияние изменения на переменную процесса. Проверьте, является ли процесс процессом с прямым или с обратным действием, и правильно ли задан управляющий выход (инвертированный или не инвертированный). Убедитесь, что верхний и нижний пределы управляющего выхода не равны друг другу.

- **Частота опроса.** Пока контур разомкнут, самое время найти оптимальную частоту опроса (процедура описана выше в данной главе). Однако, если вы собираетесь использовать автоматическую настройку, обратите внимание, что кроме коэффициентов усиления процедура автоматической настройки автоматически вычисляет и частоту отсчетов.

  Далее, рассматривается процедура ручной настройки. Если вы собираетесь пользоваться только автоматической настройкой, пропустите следующий раздел и сразу переходите к разделу об автоматической настройке.

Процедура ручной настройки

Наконец наступает волнующий момент, когда мы впервые действительно замкнем контур (перейдем в автоматический режим). Перед переключением в автоматический режим сверьтесь со следующим списком проверок:

- Проконтролируйте параметры контура с помощью средств анализа тренда контура. Рекомендуется воспользоваться функцией просмотра ПИД-регулятора в *Direct*SOFT.

ПРИМЕЧАНИЕ: *Рекомендуется использовать меню установки просмотра ПИД-регулятора для выбора «ручной» установки вертикального масштаба для областей SP/PV и смещение/управляющий выход. В противном случае функция автоматического масштабирования изменит вертикальный масштаб параметров процесса и добавит путаницу в процесс настройки контура.*

- Настройте коэффициенты усиления так, чтобы коэффициент пропорционального усиления был равен 10, коэффициент усиления интегратора — 9999, а коэффициент дифференциального усиления — 0000. Эти значения отключат интегральную и дифференциальную составляющую и обеспечат небольшой коэффициент пропорционального усиления.
- Проверьте значение составляющей смещения в таблице параметров контура (addr+04). Если оно не равно нулю, установите его в ноль с помощью *Direct*SOFT или ручного программатора, и т.д.

Теперь мы можем перевести контур в автоматический режим. Проверьте биты контроля режима для того чтобы убедиться, что переход произошел. Если контур не перейдет в Автоматический режим, обратитесь к советам по поиску неисправностей в конце этой главы.

ПРЕДУПРЕЖДЕНИЕ: E*сли значения PV и управляющего выхода начинают колебаться, немедленно уменьшите значения коэффициентов усиления. Если контур тут же не стабилизируется, переведите его обратно в ручной режим и вручную запишите безопасное значение в управляющий выход.* ***В течение процедуры настройки контура всегда находитесь около кнопки аварийной остановки, которая управляет питанием исполнительного механизма контура, чтобы при необходимости отключить его.***

- В этот момент должно выполняться условие SP = PV благодаря функции безударного перехода. Немного увеличьте SP, чтобы получить рассогласование. Если активен только коэффициент пропорционального усиления, и составляющая смещения равна 0, можно легко проверить значение управляющего выхода.

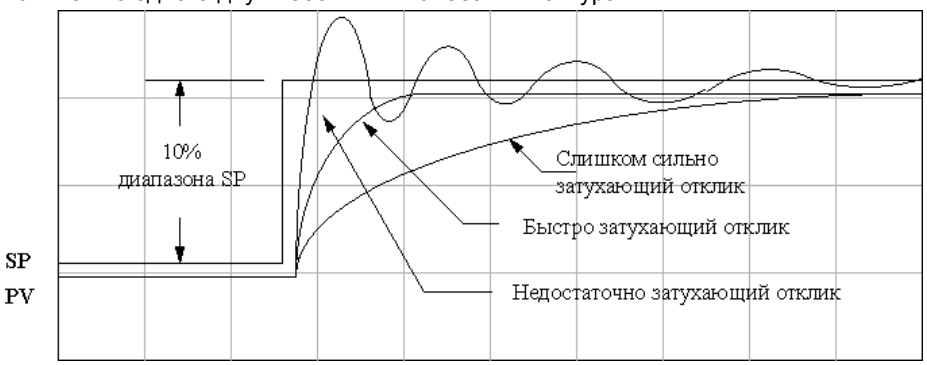
- Управляющий выход = (SP - PV) x коэффициент пропорционального усиления

- Если значение управляющего выхода изменилось, контур должен получить больше энергии от исполнительного механизма, нагревателя или другого устройства. Скоро значение PV должно сместиться в направлении SP. Если PV не меняется, то увеличивайте коэффициент пропорционального усиления, пока PV не начнет изменяться.

- Теперь немного увеличьте коэффициент интегрального усиления. **Помните, что большие числа соответствуют малым коэффициентам интегрального усиления, а малые числа — большим коэффициентам!** После этого должно выполняться условие PV = SP или их значения должны быть очень близки.

До сих пор мы использовали только коэффициенты пропорционального и интегрального усиления. Теперь можно задать «ступеньку» (изменить SP на 10 %), и настроить коэффициенты усиления, добиваясь оптимального отклика PV. Взгляните на следующий рисунок. Настройте коэффициенты усиления в соответствии с тем, что видно в окне просмотра тренда ПИД-регулятора. Показанный быстро затухающий отклик приводит к самому быстрому отклику PV без колебаний.

- **Слишком сильно затухающий отклик.** Коэффициенты усиления слишком малы, поэтому постепенно увеличивайте их, сосредоточившись в первую очередь на коэффициенте пропорционального усиления.
- **Недостаточно затухающий отклик.** Коэффициенты усиления слишком велики. Сначала уменьшите коэффициент интегрального усиления, а затем при необходимости — коэффициент пропорционального усиления.
- **Быстро затухающий отклик.** При этом коэффициенты усиления оптимальны. Можно проверить, что этот отклик является наилучшим, незначительно увеличив коэффициент пропорционального усиления. Это приведет к появлению одного-двух небольших колебаний контура.



Теперь вы можете пожелать добавить небольшой коэффициент дифференциального усиления, чтобы улучшить описанный выше быстро затухающий отклик. Обратите внимание, в этот момент коэффициенты пропорционального и интегрального усиления будут очень близки к своим окончательным значениям. Добавление некоторого дифференциального воздействия позволит слегка увеличить коэффициент пропорционального усиления без появления осцилляции контура. Дифференциальное воздействие стремится в какой-то степени ослабить пропорциональный отклик, поэтому настраивайте их совместно.

Процедура автонастройки

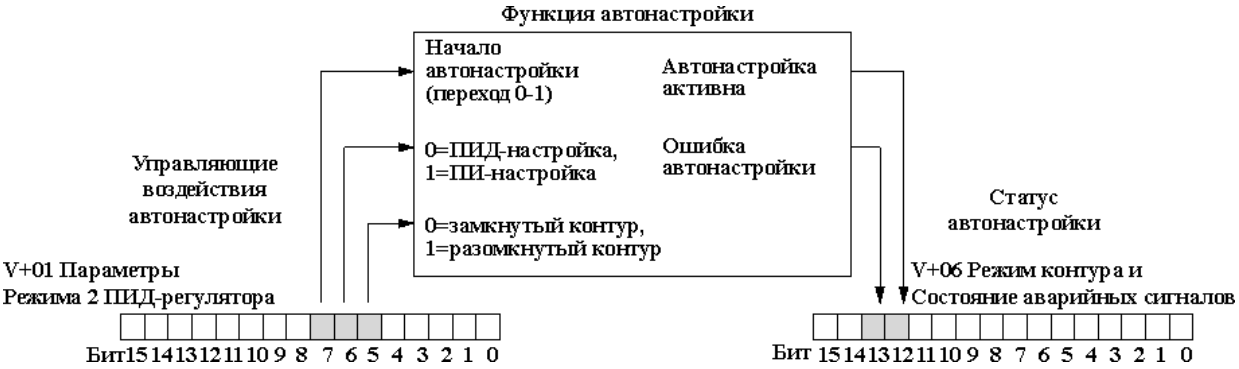
Функция автонастройки контроллера контура процессорных модулей DL250-1 и DL260 запускается только по команде инженера, который управляет процессом. Таким образом, автонастройка не выполняется непрерывно в ходе процесса (такое управление было бы **адаптивным)**. При возникновении в динамике контура заметных изменений (инерционность процесса, мощность исполнительного механизма и т.п.), понадобится повторно выполнить процедуру настройки, чтобы получить новые коэффициенты усиления, требуемые для оптимального управления.

ПРЕДУПРЕЖДЕНИЕ: *Изменения, которые влияют на константы настройки контура, должны выполняться только уполномоченным персоналом, в совершенстве знакомым со всеми аспектами процесса. Применение процедур автонастройки контура оказывает влияние на процесс, в том числе вызывая большие изменения значения управляющего выхода. Убедитесь, что вы тщательно изучили последствия любых изменений, чтобы минимизировать риск травмирования персонала или повреждения аппаратуры. Автоматическая настройка в DL250-1 и DL260 не сможет заменить изучение процесса.*

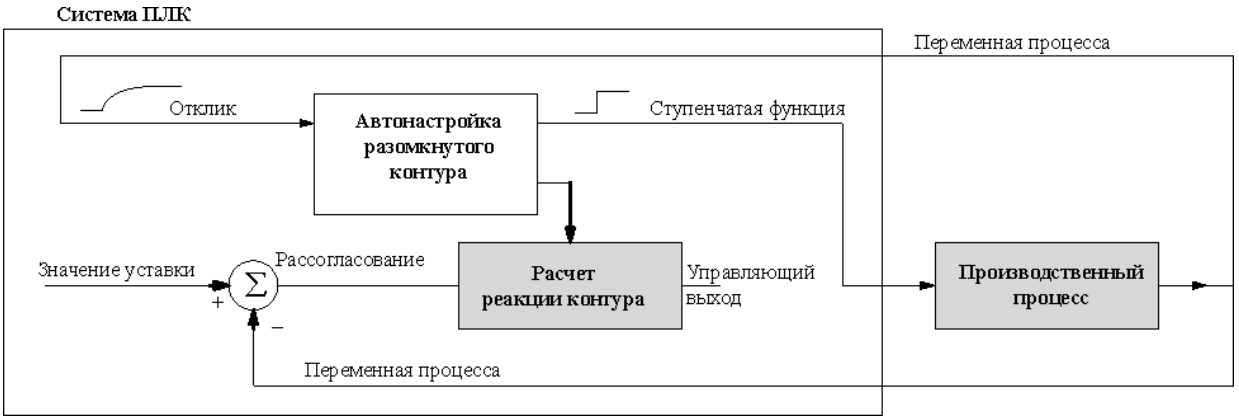
Контроллер контура обеспечивает автонастройку как по методу замкнутого контура, так и по методу разомкнутого контура. Если вы собираетесь применить процедуру автонастройки, мы рекомендуем сначала воспользоваться методом разомкнутого контура. Это позволит использовать для автонастройки метод замкнутого контура, если контур является действующим (автоматический режим) и не может быть отключен (ручной режим). Далее описано, как использовать возможности автонастройки и что при этом происходит.

Управляющие воздействия при автонастройке используют три бита Слова 2 (addr+01) настройки ПИД-регулятора, как показано ниже. При использовании функции автонастройки пакета *Direct*SOFT это программное обеспечение автоматически управляет битами. Или можно задать эти биты непосредственно из программы, что позволяет реализовать управление из другого источника, например, с помощью специализированного интерфейса оператора. Эти биты управления позволяют начать процедуру автонастройки, выбрать ПИД- или ПИ-настройку, настройку с разомкнутым или замкнутым контуром. При выборе ПИ-настройки процедура автонастройки приравнивает коэффициент дифференциального усиления нулю. Слово addr+06 (режим контура и

состояние аварийных сигналов) содержит информацию, как показано ниже, о статусе автонастройки. В течение цикла автонастройки бит 12 будет установлен (1), а после окончания цикла он будет автоматически сброшен (0).



Автонастройка разомкнутого контура. В течение цикла автонастройки разомкнутого контура контроллер контура работает так, как показано на следующей схеме. Перед началом процедуры переведите контур в ручной режим и убедитесь, что значения PV и управляющего выхода находятся в середине соответствующих диапазонов (удалены от крайних точек)

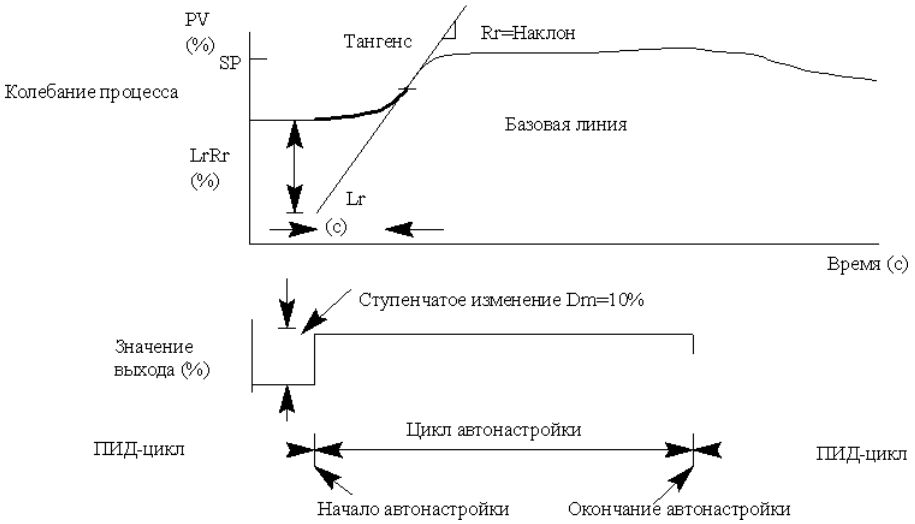


ПРИМЕЧАНИЕ: *По теории в данном случае значение SP не важно, так как контур не замкнут. Однако программное обеспечение ПЛК требует, чтобы перед началом цикла автонастройки значение SP отстояло от значения PV не менее чем на 205 единиц счета (на 205 и более единиц счета ниже SP для контуров с прямым действием или на 205 и более единиц счета выше SP для контуров с обратным действием).*

При автонастройке контроллер контура вызывает ступенчатое изменение выхода и просто следит за откликом PV. По этому отклику функция автонастройки рассчитывает коэффициенты усиления и период отсчетов. Результаты автоматически помещаются в соответствующие регистры таблицы контура.

На следующей временной диаграмме приведены события, происходящие во время цикла автонастройки открытого контура. Функция автонастройки берет на себя управление управляющим выходом и вызывает ступенчатое изменение на 10% диапазона. Если изменение PV, наблюдаемое контроллером контура, меньше 2%, то амплитуда ступенчатого изменения выхода увеличивается до 20% диапазона.

Цикл автонастройки разомкнутого контура: Метод отклика на ступенчатое изменение



- *В начале автонастройки происходит ступенчатое изменение выхода на ∆m = 10%
- *В ходе автонастройки выход контроллера достигает положительного предела диапазона. Автонастройка прекращается и в слове Аварийных сигналов устанавливается бит Ошибка автонастройки.
- *Если изменение PV меньше 2%, выход меняется на 20%.

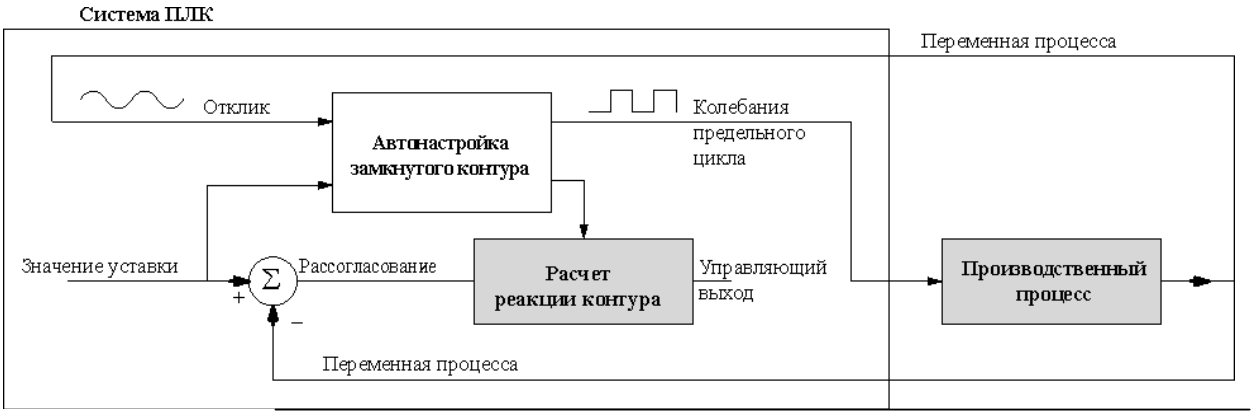
По завершении наблюдений за настройкой контура контроллер вычисляет Rr (максимальный наклон в %/c) и Lr (время запаздывания в c). Функция автонастройки вычисляет коэффициенты усиления в соответствии с приведенными ниже уравнениями Зиглера-Никольса:

ПИД-настройка	ПИ-настройка
P = 1.2 * Δm/LrRr	P = 0.9 * Δm/LrRr
I = 2.0 * Lr	I = 3.33 * Lr
D = 0.5 * Lr	D = 0
Период отсчетов = 0.056 * Lr	Период отсчетов = 0.12 * Lr
Δm = Ступенчатое изменение выхода (10% = 0.1, 20% = 0.2)	

Настоятельно рекомендуется использовать в качестве интерфейса автонастройки пакет *Direct*SOFT. Длительность цикла автонастройки будет зависеть от инерционности процесса. Медленное изменение PV приведет к увеличению длительности цикла автонастройки. По окончании автонастройки значения пропорционального, интегрального и дифференциального коэффициентов усиления в ячейках таблицы контура addr+10, addr+11 и addr+12, соответственно, будут автоматически обновлены. Кроме того, автоматически обновляется период отсчетов в addr+07. Правильность результатов автонастройки можно проверить, измеряя отклик PV замкнутого цикла на ступенчатое изменение выхода. Соответствующие инструкции находятся в разделе, описывающем процедуру ручной настройки (расположенном перед данным разделом об автонастройке).

Ошибка автонастройки: Если бит ошибки автонастройки (бит 13 Режима контура и аварийных состояний addr+06) включен, то проверьте PV, и значение SP - в пределах 5% от полного диапазона, как требуется функции автонастройки. Бит будет также включен, если используется режим замкнутого контура, и выход вышел на пределы диапазона.

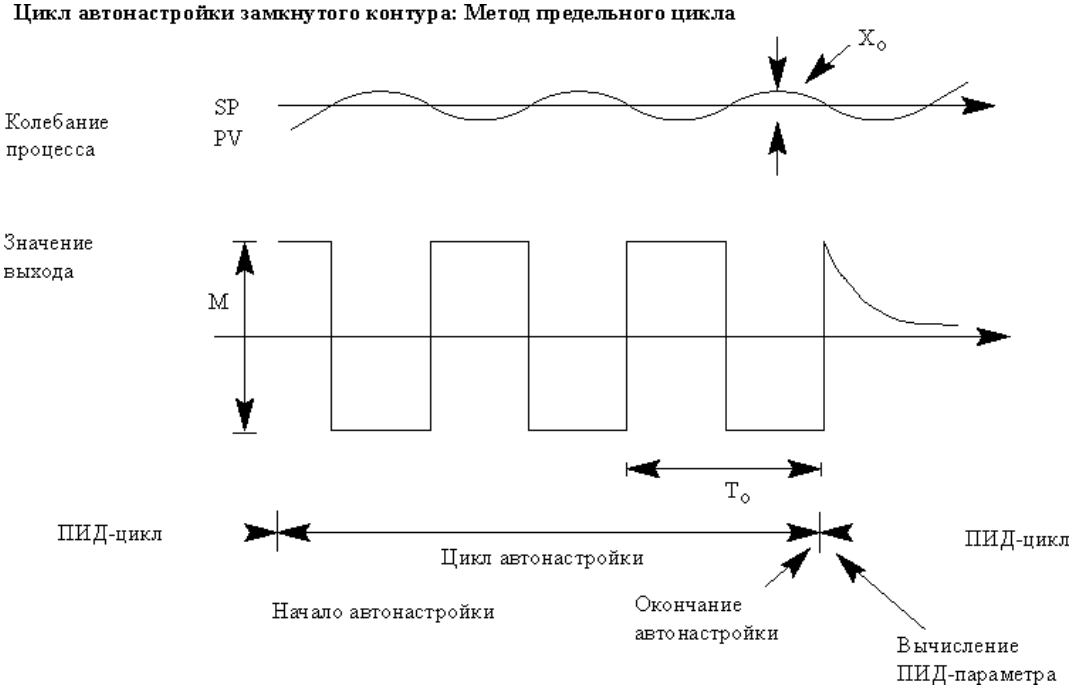
Автонастройка замкнутого контура. На приведенной ниже схеме показана работа контроллера в течение цикла автонастройки замкнутого контура.



В ходе автонастройки контроллер контура подает на выход меандр. Переключение выхода возникает, когда значение PV проходит (вверх или вниз) через значение SP. Следовательно, частота предельного цикла примерно пропорциональна инерционности процесса. По отклику PV функция автонастройки рассчитывает коэффициенты усиления и период отсчетов. Результаты автоматически помещаются в соответствующие регистры таблицы контура.

На следующей временной диаграмме показаны события, возникающие в течение цикла автонастройки замкнутого контура. Функция автонастройки проверяет направление смещения PV относительно SP. Затем функция автонастройки перехватывает управление управляющим выходом и вызывает ступенчатое изменение в противоположном направлении на весь диапазон возможных значений. При каждом изменении знака рассогласования (SP – PV) выход изменяется в противоположном направлении на максимально возможное значение. Так происходит три полных цикла.

Цикл автонастройки замкнутого контура: Метод предельного цикла



- *M_{max} = верхний предел значения выхода, M_{min} = нижний предел значения выхода.
- *В примере показан контур с прямым действием. Для контура с обратным действием выход инвертируется

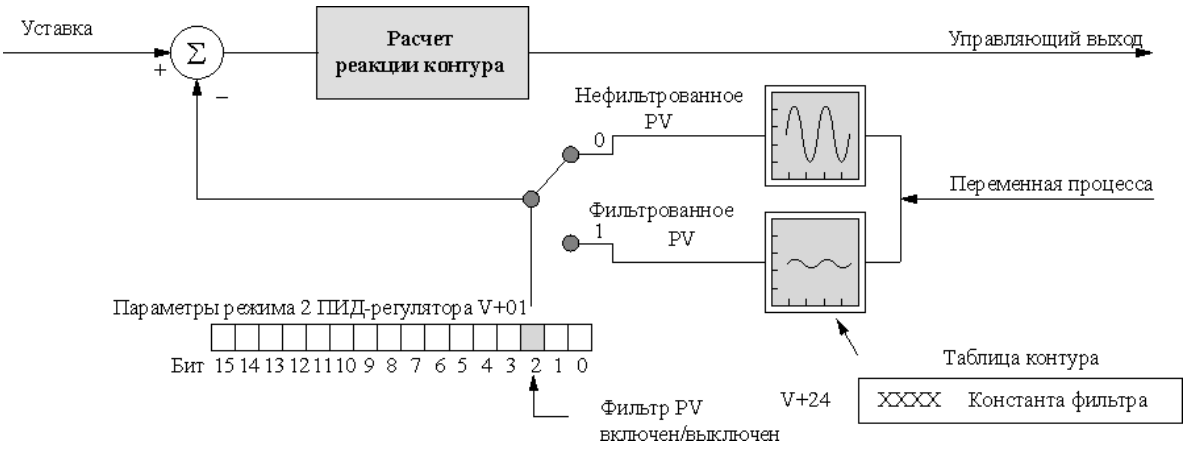
По завершении наблюдений за настройкой контура контроллер вычисляет To (период возмущений) и Xo (амплитуду PV). Затем он использует эти значения для вычисления Kpc (предел чувствительности) и Tpc (предел периода). По этим значениям функция автонастройки контроллера контура вычисляет ПИД-коэффициенты усиления и период отсчетов в соответствии с приведенными ниже уравнениями Зиглера-Никольса:

ПИД-настройка	ПИ-настройка
P = 0.45 * Kpc	P = 0.30 *Kpc
I = 0.60 * Tpc	I = 1.00 * Tpc
D = 0.10 * Tpc	D = 0
Период отсчетов = 0.014	Период отсчетов = 0.03 * Tpc

Ошибка автонастройки: Если бит автонастройки (13-ый бит в статусном слове addr+06 Режим контура и Аварийные сигналы) включен, то проверьте, что значения PV и SP отличаются друг от друга не более, чем на 5 процентов их полной шкалы, как это требуется для функции автонастройки. Этот бит также будет включен, если используется метод разомкнутого контура, и выходной сигнал выходит за пределы диапазона.

Аналоговый фильтр для PV

Как можно видеть из временной диаграммы, приведенной на предыдущей странице, важны точки нулевых значений разности SP и PV. Очевидно, зашум-ленный сигнал PV может создать дополнительные нулевые точки и привести к неправильному определению контроллером характеристик контура. DL250-1 и DL260 обеспечивают для PV входной низкочастотный фильтр первого порядка, который можно использовать при автонастройке методом замкнутого контура (см. рисунок ниже). Настоятельно рекомендуется использовать этот фильтр во время автонастройки. После автонастройки фильтр можно отключить или продолжать использовать для зашумленного входного сигнала PV.



Бит 2 Слова 2 (addr+01) параметров Режима ПИД-регулятора обеспечивает управление включением/выключением низкочастотного фильтра PV (0=выключен, 1=включен). Частота спада однополюсного низкочастотного фильтра задается с помощью регистра addr+24 таблицы параметров цикла, константа фильтра. Эта константа задается в формате BCD, с подразумеваемой десятичной точкой OOX.X следующим образом:

- Константа фильтра может принимать значения от 000.1 до 001.0.
- Задание значения 000.0 или от 001.1 до 999.9 полностью отключает фильтр.
- Значения, близкие к 001.0, дают более высокие частоты спада, а значения, близкие к 000.1, дают более низкие частоты спада.

Настоятельно рекомендуется использовать в качестве интерфейса автонастройки пакет программирования *Direct*SOFT. Длительность цикла автонастройки будет зависеть от инерционности процесса. Медленное изменение PV приведет к увеличению длительности цикла автонастройки.

По окончании автонастройки значения пропорционального, интегрального и дифференциального коэффициентов усиления в ячейках таблицы контура addr+10, addr+11 и addr+12, соответственно, будут автоматически обновлены. Кроме того, автоматически обновляется период отсчетов в addr+07. Правильность результатов автонастройки можно проверить, измеряя отклик PV замкнутого цикла на ступенчатое изменение выхода. Соответствующие инструкции находятся в разделе, описывающем процедуру ручной настройки (расположенном перед данным разделом об автонастройке).

Алгоритм встроенного фильтра таков:

$Yi = k (Xi - Yi-1) + Yi-1;$

Где: Yi-сигнал на выходе фильтра

Xi- сигнал на входе фильтра

Yi-1-предыдущее значение сигнала на выходе фильтра

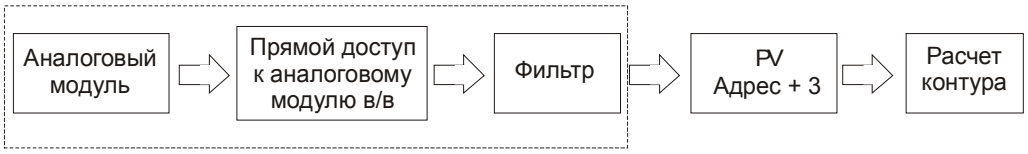
k- аналоговый коэффициент фильтрации ввода переменной PV

Функции прямого доступа к PV с фильт-фильт-рующими возможно-стями

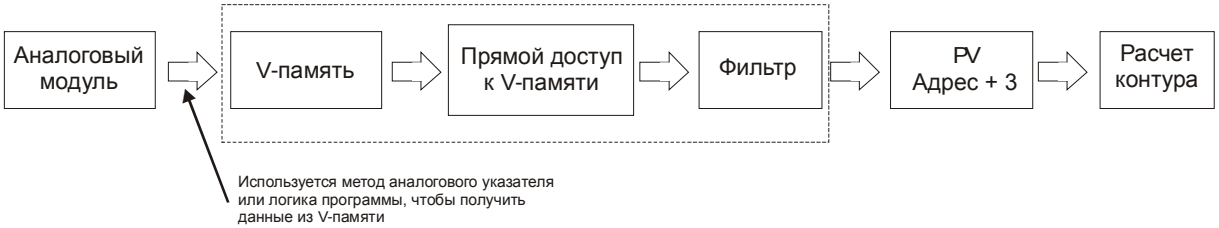
На рисунке внизу показано как взаимодействуют функции прямого доступа к PV (addr + 36) и фильтрации (addr + 01, бит 2). Есть следующие варианты:

- Прямой доступ к каналу аналогового модуля с включенным или выключенным фильтром. Когда эта функция используется, не применим метод аналогового указателя для чтения данных в каналах модуля.
- Прямой доступ к ячейке V-памяти с включенным или выключенным фильтром. Когда эта функция используется, как метод аналогового указателя, так и логику программы можно применить для записи значения в заданную ячейку V-памяти.

Прямой доступ к аналоговому модулю (фильтрация включена)



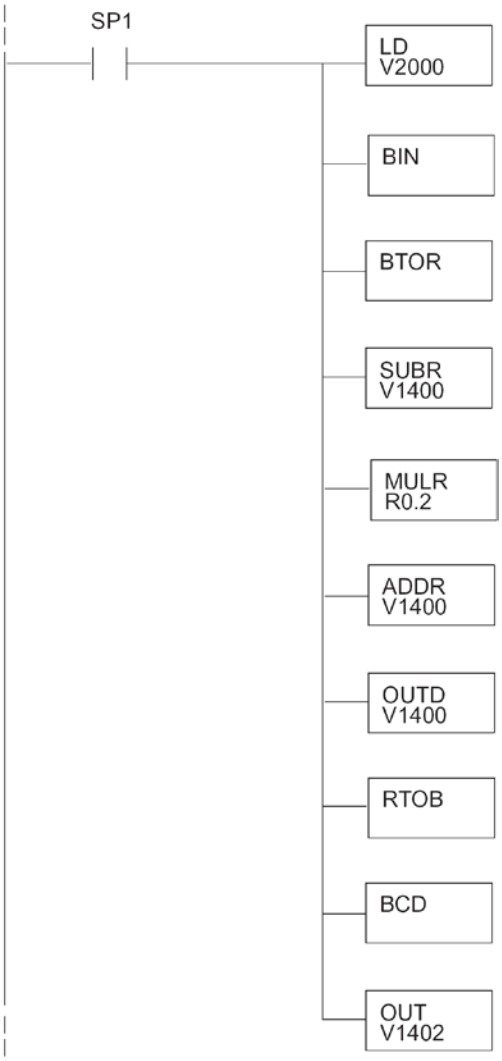
Прямой доступ к V-памяти (фильтрация включена)



Создание аналогово-го фильтра в релейной программе

Подобный алгоритм может быть реализован программно на релейной логике. Ваши аналоговые входы могут быть эффективно отфильтрованы, используя любой метод. Далее приведен пример программы, использующей необходи-мую релейную логику. Не забудьте изменить адреса ячеек памяти в соответст-вии с вашим приложением.

- Фильтрация может внести одну тысячную долю погрешности из-за округ-ления. Если ваш процесс не допускает такой погрешности - не используй-те фильтрацию.
- Из-за ошибки округления, вы не сможете использовать ноль и конец шка-лы, как точки аварийной сигнализации. Кроме того, уменьшение коэффи-циента фильтрации улучшает эффект сглаживания, но замедляет время реакции. Убедитесь в том, что это замедление не повлияет на качество управления вашим процессом.



Загрузка аналогового сигнала, который является двоично-десятичным числом, и загрузка из ячейки V-памяти V2000, в аккумулятор. Контакт SP1 всегда включен.

Преобразование двоично-десятичного значения в аккумуляторе в двоичное. Эта команда не требуется, если первоначально аналоговое значение приведено как двоичное число.

Преобразование двоичного значения в аккумуляторе в вещественное число

Вычитание вещественного числа, сохранённого в ячейке V1400, из вещественного числа в аккумуляторе, и сохранение результата в аккумуляторе. В этом примере V1400–указанная рабочая область памяти.

Умножение вещественного числа в аккумуляторе на 0.2 (коэффициент фильтра), и сохранение результата в аккумуляторе. Это –отфильтрованное значение.

Сложение вещественного числа, сохранённого в ячейке V1400, и вещественного числа, отфильтрованного значения, в аккумуляторе и сохранение результата в аккумуляторе.

Копирование значения из аккумулятора в ячейку V1400.

Преобразование вещественного числа в аккумуляторе в двоичное значение, и сохранение результата в аккумуляторе.

Преобразование двоичного значения в аккумуляторе в двоично-десятичное число.

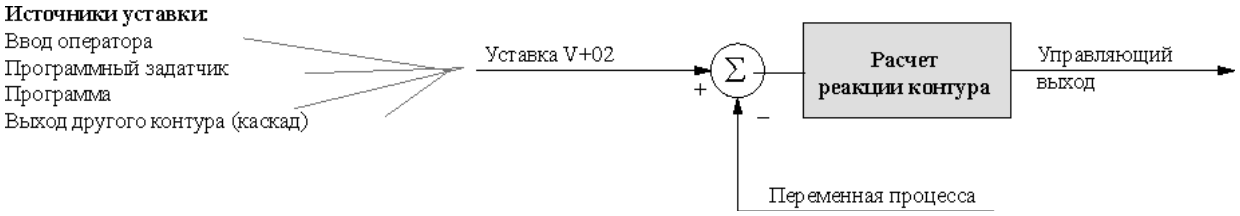
Примечание: команда преобразования не требуется для переменной PV ПИД-контура (PV контур – двоичное число).

Загрузка отфильтрованного значения двоично-десятичного числа из аккумулятора в ячейку V1402 для использования в приложении или ПИД-контуре.

Программный задатчик

Введение

При обсуждении работы контура отмечалось, что задание для контура может генерироваться различным образом, в зависимости от режима работы контура и программных предпочтений. На следующем рисунке одним из способов создания SP является программный задатчик. **За то, что в конкретный момент времени только один источник будет пытаться записать в addr+02 значение SP, отвечает пользовательская программа.**

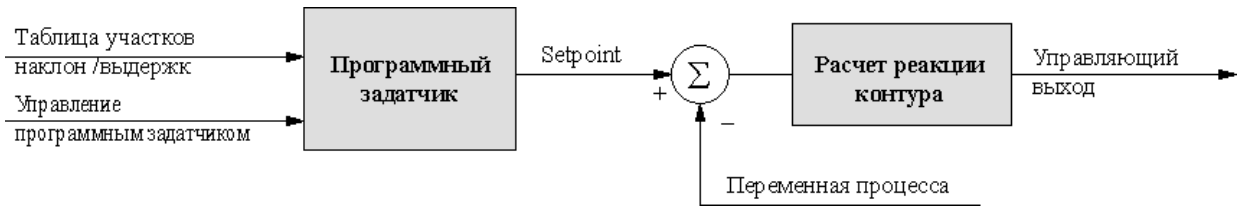


Если SP процесса меняется редко или допускает ступенчатые изменения, использование программного задатчика, возможно, не потребуется. Однако, для некоторых процессов потребуется точно управлять изменениями значения SP. Программный задатчик может значительно снизить объем программирования, необходимый для подобных приложений.

В управлении процессами термины «наклонный участок» («ramp») и «участок выдержки» («soak») имеют специальные значения и относятся к требуемому значению задания SP в приложениях, которые управляют температурой. На рисунке справа задание возрастает на наклонном участке и остается постоянным на участке выдержки.

Сложные профили SP могут быть созданы с помощью задания последовательности таких участков. Наклон линейных участков задается в единицах SP в секунду. Длительность участка выдержки задается в минутах.

Стоит рассматривать программный задатчик, как специальную функцию для генерации значений SP, как показано ниже. У нее есть две категории входов, определяющих генерируемые значения SP. Заранее необходимо запрограммировать **таблицу задатчика**, содержащую значения, которые будут определять профиль сигнала наклон/выдержка. По мере необходимости контур считывает эти значения из таблицы при каждом расчете ПИД-уравнения. За управление программным задатчиком отвечают биты в специальном слове таблицы контура, в реальном времени управляющие возможностью запускать/останавливать программный задатчик. Пользовательская программа может следить за состоянием профиля сигнала (текущим номером участка наклона).



Теперь, после описания общих принципов работы программного задатчика, перечислим его основные характеристики:

- У каждого контура есть свой программный задатчик (его использование необязательно).
- Можно задать до восьми шагов наклон/выдержка (16 участков).
- Программный задатчик может работать всегда, когда ПЛК находится в рабочем режиме. Его работа не зависит от режима контура (ручной или автоматический).
- Управляющие воздействия реального времени включают Start (запуск), Hold (остановка), Resume (продолжение) и Jog (толчок).
- Контроль сигнала программного задатчика включает Завершение профиля, Отклонение PV на участке-выдержке (SP минус PV) и номер текущего шага профиля наклон /выдержка.

На следующем рисунке показан профиль SP, состоящий из пар участков наклон/выдержка. Каждый из участков получает номер от 1 до 16. Наклон каждого из линейных участков может быть положительным или отрицательным. Программный задатчик автоматически узнает, увеличивать или уменьшать SP, на основании относительных значений конечных точек линейных участков. Эти значения считываются из таблицы программного задатчика.

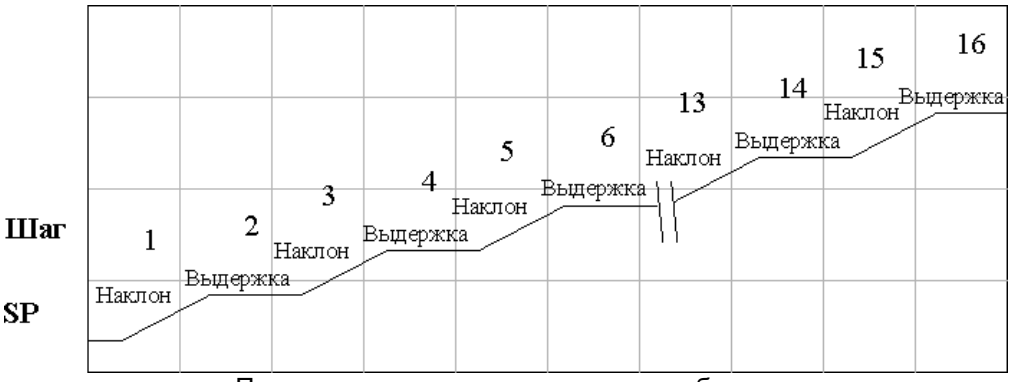
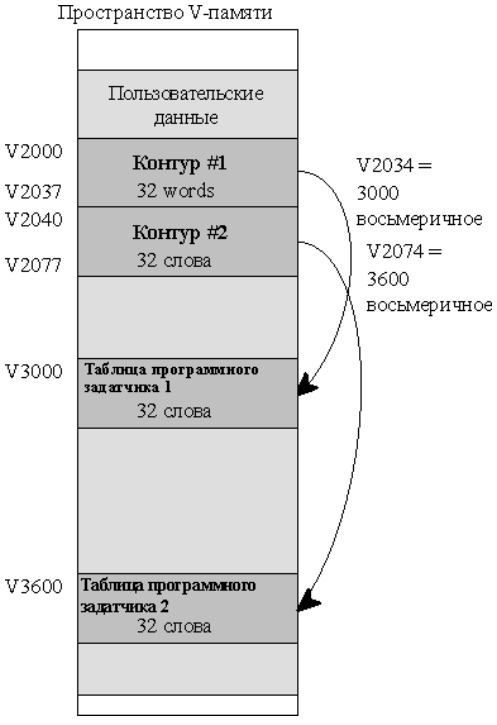


Таблица программного задатчика

Параметры, определяющие пилообразный профиль контура, хранятся в таблице программного задатчика. У каждого контура может быть своя таблица программного задатчика, но это необязательно. Вспомним, что таблица параметров каждого контура состоит из 32 слов, занимающих непрерывную область памяти. Однако, таблица программного задатчика контура располагается отдельно, так как не является обязательной. Указатель в ячейке addr+34 таблицы контура задает начальную ячейку таблицы программного задатчика.

В приведенном справа примере таблицы параметров контура 1 и контура 2 занимают, как показано, непрерывные блоки из 32 слов. Каждая содержит указатель на свою таблицу программного задатчика, независимо располагающуюся где-то в другом месте пользовательской V-памяти. Конечно, можно разместить все таблицы в одной группе, поскольку они не перекрываются.



Параметры в таблице программного задатчика должны определяться пользователем. Удобнее всего использовать пакет программирования *Direct*SOFT, включающий специальный редактор для этой таблицы. Для определения пары участков наклон/выдержка требуется задать четыре параметра, как показано на нижеследующем рисунке.

- **Окончание участка наклона (Ramp End Value).** Задает значение SP для конечной точки участка наклона. Формат этого числа должен совпадать с форматом, используемым для SP. Может быть ниже или выше начального значения SP, поэтому наклон может быть положительным или отрицательным (нам не нужно знать начальное значение SP для участка наклона 1).
- **Крутизна наклона (Ramp Slope).** Задает увеличение SP в единицах счета в секунду. Это число в формате BCD, в диапазоне от 00.00 до 99.99 (с подразумеваемой десятичной точкой).
- **Длительность участка выдержки (Soak Duration)**. Задает время участка выдержки в минутах, в диапазоне от 000.1 до 999.9 минут и в формате BCD (с подразумеваемой десятичной точкой).
- **Отклонение PV на участке выдержки (Soak PV Deviation).** Необязательный параметр, задает допустимое отклонение PV вверх и вниз от значения SP во время выдержки. Бит состояния аварийного сигнала отклонения PV устанавливается программным задатчиком.

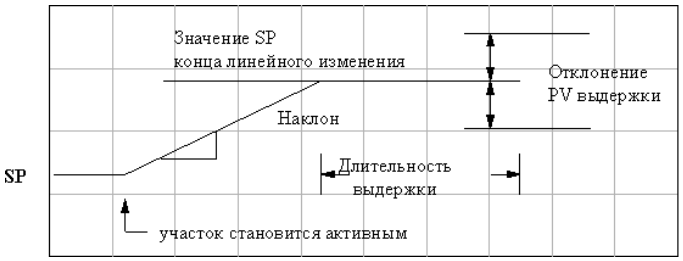


Таблица программного задатчика

V+00	XXXX	Значение SP конца линейного изменения
V+01	XXXX	Наклон линейного изменения
V+02	XXXX	Длительность выдержки
V+03	XXXX	Отклонение PV выдержки

Наклонный участок становится активным, когда заканчивается предыдущий участок выдержки. Первый наклонный участок активизируется при запуске программного задатчика, автоматически рассматривая текущее значение SP как свое **начальное** значение.

Смещение адреса	Шаг	Описание		Смещение адреса	Шаг	Описание
+ 00	1	Значение SP окончания участка наклона		+ 20	9	Значение SP окончания участка наклона
+ 01	1	Крутизна наклона		+ 21	9	Крутизна наклона
+ 02	2	Длительность участка выдержки		+ 22	10	Длительность участка выдержки
+ 03	2	Отклонение PV на участке выдержки		+ 23	10	Отклонение PV на участке выдержки
+ 04	3	Значение SP окончания участка наклона		+ 24	11	Значение SP окончания участка наклона
+ 05	3	Крутизна наклона		+ 25	11	Крутизна наклона
+ 06	4	Длительность участка выдержки		+ 26	12	Длительность участка выдержки
+ 07	4	Отклонение PV на участке выдержки		+ 27	12	Отклонение PV на участке выдержки
+ 10	5	Значение SP окончания участка наклона		+ 30	13	Значение SP окончания участка наклона
+ 11	5	Крутизна наклона		+ 31	13	Крутизна наклона
+ 12	6	Длительность участка выдержки		+ 32	14	Длительность участка выдержки
+ 13	6	Отклонение PV на участке выдержки		+ 33	14	Отклонение PV на участке выдержки
+ 14	7	Значение SP окончания участка наклона		+ 34	15	Значение SP окончания участка наклона
+ 15	7	Крутизна наклона		+ 35	15	Крутизна наклона
+ 16	8	Длительность участка выдержки		+ 36	16	Длительность участка выдержки
+ 17	8	Отклонение PV на участке выдержки		+ 37	16	Отклонение PV на участке выдержки

Флаги таблицы программного задатчика

Многим приложениям не требуются все 16 шагов наклона/выдержки. Для всех неиспользуемых шагов в таблице установите 0. Программный задатчик заканчивает выполнение профиля, когда обнаруживает, что наследующем шаге крутизна наклона =0.

Определения отдельных битов слова Флагов таблицы программного задатчика (Addr+33) перечислены в следующей таблице

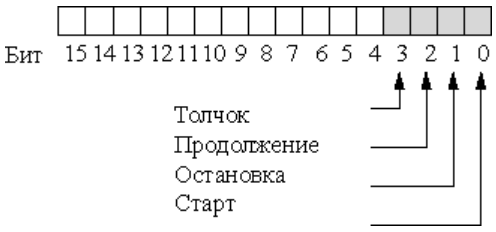
Бит	Описание битов флагов программного задатчика	Чтение/ Запись	Бит = 0	Бит = 1
0	Начать работу программного задатчика	запись	-	0=>1 Старт
1	Приостановить профиль программного задатчика	запись	-	0=>1 Остановка
2	Продолжить профиль программного задатчика	запись	-	0=>1 Продолжение
3	Толкнуть программный задатчик	запись	-	0=>1 Толчок
4	Завершение профиля программного задатчика	чтение	-	Завершение
5	Отклонение входа PV от профиля сигнала	чтение	Выкл	Вкл
6	Остановка программного задатчика	чтение	Выкл	Вкл
7	Зарезервирован	чтение	Выкл	Вкл
8-15	Текущий шаг профиля программного задатчика	чтение	Декодируется как байт (шестнадцатеричный)	

Включение программного задатчика

Включение программного задатчика осуществляется с помощью бита 11 Слова 1 (addr+00) параметров режима ПИД-регулятора, как показано на рисунке. Другие средства управления программным задатчиком, показанные в приведенной выше таблице, не будут действовать, если данный бит не равен 1.

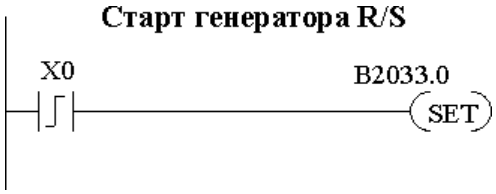
Средства управления программным задатчиком

Четыре основных управляющих воздействия на программный задатчик реализуются с помощью битов 0-3 слова параметров сигнала наклон/выдержка в таблице параметров контура. *Direct*SOFT непосредственно управляет этими битами, используя диалог задания параметров программного задатчика. Однако в ходе выполнения программы эти биты должны управляться программно. Рекомендуем использовать команду бит-слова.



Для выполнения нужной функции пользовательская программа должна установить соответствующий бит равным 1. Когда контроллер контура считывает значение программного задатчика, он автоматически сбрасывает этот бит. Следовательно, сброс бита, когда процессор находится в рабочем режиме, не требуется.

Приведенный справа пример цепи программы показывает, как после включения внешнего переключателя X0 контакт PD использует передний фронт для установки соответствующего бита управления, запускающего программный задатчик. При этом для бита слова используется команда Set (Установить).



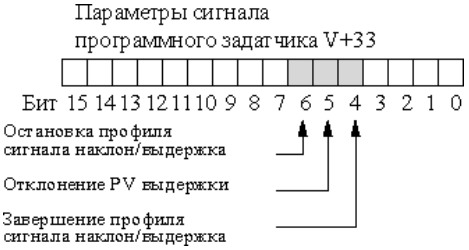
В нормальном состоянии все биты управления программного задатчика равны 0. Программа в каждый момент времени должна устанавливать не более одного бита управления.

- **Старт.** Переход 0-1 запустит программный задатчик. Процессорный модуль должен быть в рабочем режиме, а контур может быть в ручном или автоматическом режиме. Если профиль не прерывается командой Остановка или Толчок, он нормально завершается.
- **Остановка.** Переход 0-1 остановит программный задатчик в текущем состоянии, при этом фиксируется значение SP.
- **Продолжение.** Переход 0-1 инициирует продолжение работы программного задатчика, если он был в состоянии Остановки. Значение SP будет равно своему предыдущему значению.
- **Толчок.** Переход 0-1 заставляет программный задатчик прервать текущий участок (шаг) и перейти к следующему участку.

Мониторинг профиля сигнала наклон/ выдержка

Отслеживать состояние профиля сигнала программного задатчика можно с помощью других битов слова параметров сигнала наклон/выдержка, addr+33, как показано справа.

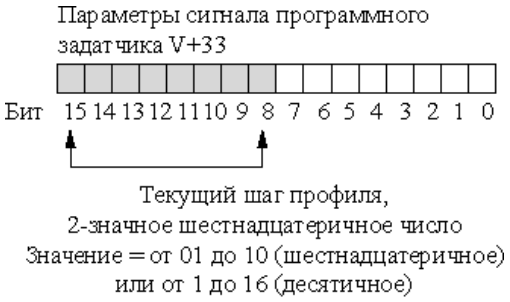
- **Завершение профиля программного задатчика.** Равен 1, если программа выполнила последний запрограммированный шаг.
- **Отклонение PV на участке выдержки.** Равен 1, если ошибка



(SP-PV) превышает значение, заданное в таблице программного задатчика.

- **Остановка профиля программного задатчика.** Равен 1, если профиль активен, но в данный момент приостановлен.

Номер текущего шага хранится в верхних 8 битах слова addr+33 параметров задатчика. Биты представляют собой двузначное шестнадцатеричное число в диапазоне от 1 до 10. Программа может следить за этими битами, чтобы синхронизировать другие части программы с профилем сигнала программного задатчика. Загрузите это слово в аккумулятор, сдвиньте его вправо на 8 битов и получите номер шага.



Ошибки программирования программного задатчика

Начальный адрес таблицы программного задатчика должен быть доступной ячейкой. Если адрес указывает на ячейку, находящуюся вне пользовательского диапазона V-памяти, то при запуске программного задатчика будет установлен один из битов, показанных справа. Для настройки таблицы программного задатчика рекомендуется воспользоваться пакетом программирования *Direct*SOFT, который автоматически выполняет проверку диапазонов.

Тестирование профиля сигнала программного задатчика

Перед использованием профиля сигнала программного задатчика для управления процессом рекомендуется выполнить его проверку. Это несложно, так как программный задатчик будет работать, даже когда контур находится в ручном режиме. Сберечь время поможет окно Просмотр ПИД-регулятора (PID View) пакета программирования *Direct*SOFT, так как в нем профиль сигнала выводится на экран. Не забудьте выбрать достаточно медленную временную развертку, чтобы полностью вывести пары участков сигнала наклон/выдержка на экран.

Каскадное управление

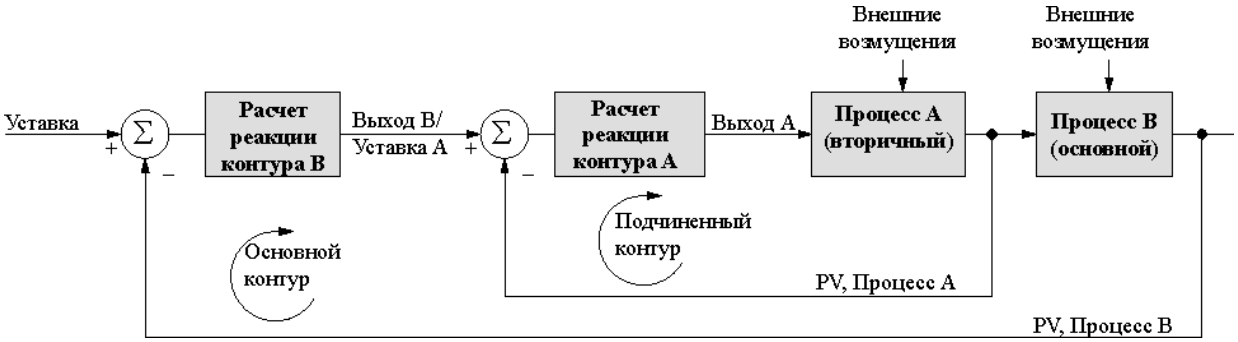
Введение Каскадные контуры представляют собой усовершенствованный метод управления, в определенных ситуациях более выгодный, чем управление с помощью отдельных контуров. Как следует из названия, каскад означает, что один контур подключен к другому. Помимо Ручного (разомкнутый контур) и Автоматического (замкнутый контур) режимов процессорные модули DL250-1 и DL260 также обеспечивают каскадный режим.

ПРИМЕЧАНИЕ: *Каскадные контуры — это более сложный метод управления. Поэтому их использование рекомендуется только для опытных инженеров.*

Если производственный процесс является сложным и включает задержку между управляющим выходом и изменением переменной процесса, использование даже самого идеально настроенного отдельного контура может приводить к медленному и неточному управлению. Такое может случиться, если исполнительный механизм, использующий одно физическое свойство, в конечном счете воздействует на переменную процесса, измеряемую с помощью другого физического свойства. Задание промежуточной переменной позволит разделить процесс на две части, как показано на следующем рисунке.

Принцип каскадных контуров просто заключается в том, что мы добавляем контур еще одного процесса для более точного управления промежуточной переменной! Это также разбивает источник запаздывания управления на две части.

На следующем рисунке показана каскадная система управления, представляющая собой просто вложение одного контура в другой. Внутренний контур называется подчиненным, а внешний контур — главным. Для максимальной устойчивости из двух контуров более быстрым откликом должен обладать внутренний. Для измерения промежуточной переменной (PV процесса A) понадобится добавить дополнительный датчик. Обратите внимание, что задание для подчиненного контура создается автоматически, с помощью выхода основного контура. После программирования и отладки каскадного управления предстоит работать только с первоначальными заданием и переменной процесса на системном уровне. Каскадные контуры ведут себя как один контур, но обеспечивают лучшие характеристики по сравнению с решением, использующим один контур.



Одно из преимуществ каскадного управления можно обнаружить, проверяя отклик на внешние возмущения. Вспомним, что подчиненный контур реагирует быстрее основного. Следовательно, если возмущение оказывает влияние на процесс A в подчиненном контуре, расчет ПИД-коэффициентов контура A может исправить получающуюся ошибку до того, как эффект будет обнаружен основным контуром.

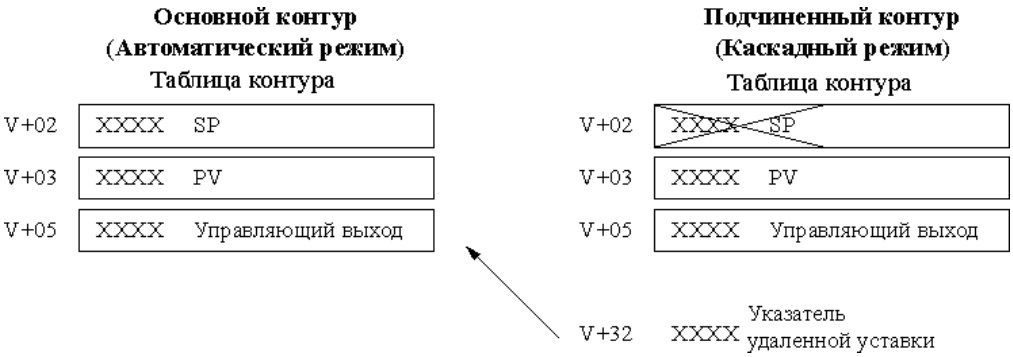
Каскадные контуры в процессорах DL250-1 и DL260

Используя термин «каскадные контуры», необходимо учитывать важное замечание. Реально только подчиненный контур будет работать в каскадном режиме. При нормальной работе главный контур должен находиться в автоматическом режиме. Если количество контуров в каскаде больше двух, при нормальной работе в автоматическом режиме должен находиться самый внешний (главный) контур, а все внутренние контуры работают в каскадном режиме.

ПРИМЕЧАНИЕ: *Формально в соответствии со строгой терминологией управления процессом «каскадными» являются как подчиненный, так и основной контура. К сожалению, задавая режимы контуров, об этом соглашении приходится забыть. Помните, что все подчиненные контуры находятся в каскадном режиме, и только самый внешний (главный) контур будет работать в автоматическом режиме.*

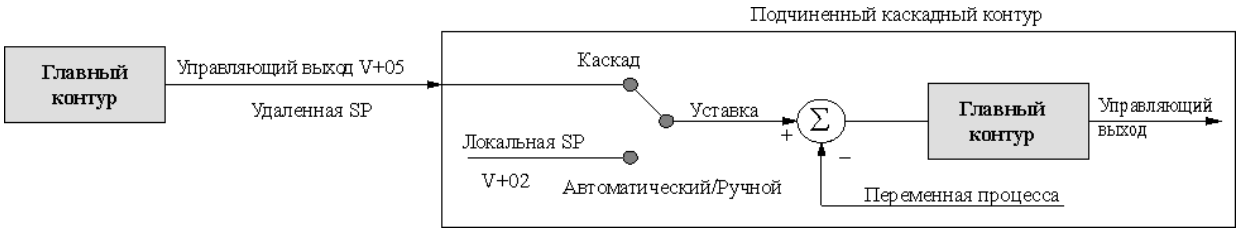
С помощью DL250-1 и DL260 можно объединить в каскад любое нужное число контуров, а также создать несколько групп каскадных контуров. Для правильной работы каскадных контуров необходимо использовать для главного и подчиненного контуров одинаковые диапазоны данных (12/15 битовые) и настройки полярности.

Для подготовки подчиненного контура к работе в каскадном режиме необходимо, как показано ниже, задать в ячейке addr+32 его таблицы контура значение Указателя удаленной уставки. Значением указателя должен быть адрес ячейки addr+05 (управляющий выход) главного контура. В каскадном режиме подчиненный контур будет игнорировать собственный локальный регистр задания SP (addr+02), считывая в качестве своего задания значение управляющего выхода основного контура.



При использовании для просмотра значения SP подчиненного контура окна Просмотр ПИД-регулятора (PID View) *Direct*SOFT автоматически считывает значение управляющего выхода главного контура и выводит это значение как SP подчиненного контура. Обычная ячейка SP подчиненного контура, addr+02, остается без изменений.

Теперь воспользуемся приведенными выше параметрами контура и изобразим эквивалентную схему.



Помните, что если подчиненный контур перестает работать в каскадном режиме, основной контур автоматически переходит в ручной режим.

Настройка каскадных контуров

При настройке каскадных контуров Вам необходимо разъединить соединение каскадных контуров и настраивать контура индивидуально, используя одну из описанных ранее процедур.

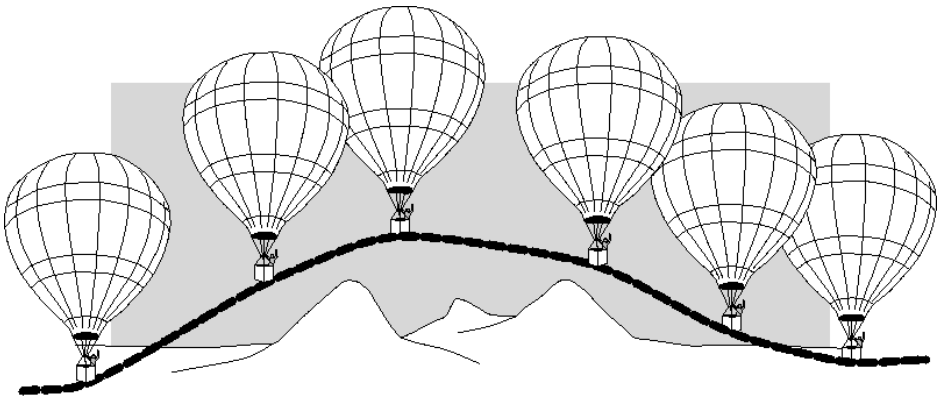
1. Если Вы не используете автонастройку, тогда определите период опроса контура (*sample rate*) ведомого контура (*minor loop*), используя метод, рассмотренный ранее в этой главе. Затем установите период опроса ведущего контура (*major loop*) в 10 раз медленнее ведомого контура. Используйте эти значения в качестве первоначальных значений.
2. Настройте сначала ведомый контур. Оставьте ведущий контур в ручном режиме (*Manual Mode*) и генерируйте значение SP для ведомого контура вручную , как описано в процедуре настройки контура.
3. Убедитесь в том, что ведомый контур дает затухающий отклик на 10% изменение значения SP, находясь в режиме автоматического управления (*Auto Mode*). Настройка ведомого контура завершена.
4. Теперь Вам необходимо перевести ведомый контур в каскадный режим (*Cascade Mode*), а, затем, перевести ведущий контур в *Auto Mode*. При настройке ведущего мы будем рассматривать как последовательно соединенный компонент в общем процессе. Поэтому не возвращайтесь, и настраивайте ведомый контур при настройке ведущего контура.
5. Настраивайте ведущий контур, следуя стандартной процедуре. Отклик переменной PV ведущего контура является общим (полным) откликом всего каскадного контура.

Широтно-импульсное управление

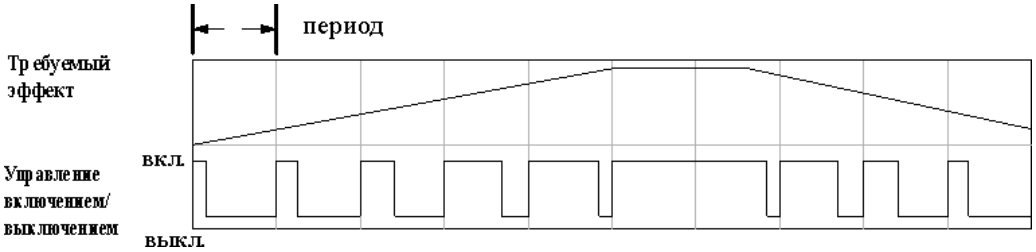
Контроллер контура ПИД-регулирования процессорных модулей DL250-1 и DL260 создает непрерывный сигнал на управляющем выходе в некотором численном диапазоне. Значение управляющего выхода годится для управления аналоговым выходным модулем, связанным с процессом. В управлении процессами такой способ называется непрерывным управлением, потому что выход включен (на некотором уровне) постоянно.

Хотя непрерывное управление может быть гладким и устойчивым, стоимость компонентов контура (например, исполнительных механизмов или нагревателей) может быть достаточно велика. Более простая форма управления называется дискретным управлением. В этом методе применяются исполнительные механизмы, которые либо включены, либо нет (без промежуточных состояний). Компоненты контура для систем управления с включением/выключением дешевле, чем их аналоги, предназначенные для непрерывного управления.

В данном разделе мы покажем, как преобразовать управляющий выход контура в дискретное управление для требующих этого приложений. Посмотрим, как можно управлять с помощью чередующегося включения/выключения нагрузки. На следующей диаграмме показан пересекающий горы шар с горячим воздухом. Требуемый путь является уставкой. Пилот шара чередует включение/выключение горелки, являющейся управляющим выходом. Большая масса воздуха в шаре эффективно усредняет действие горелки, преобразуя вспышки тепла в непрерывное действие, медленно меняя температуру шара и, в конечном счете, высоту, которая является переменной процесса.



Широтно-импульсное управление является приближением непрерывного управления за счет особенностей рабочего цикла — отношения времени включения к времени выключения. На следующем рисунке приведен пример того, как рабочий цикл обеспечивает приближение к непрерывному уровню при усреднении с помощью большой инерционности процесса.



Если бы мы нарисовали времена включения/выключения горелки шара с горячим воздухом, мы, вероятно, получили бы очень похожий процесс, связанный с температурой и размером шара.

Пример программы с дискретным управлением

В следующем примере программа реализует функцию дискретного управления. Она преобразует непрерывный выход в V2005 в управление Y0 с помощью включения/выключения.

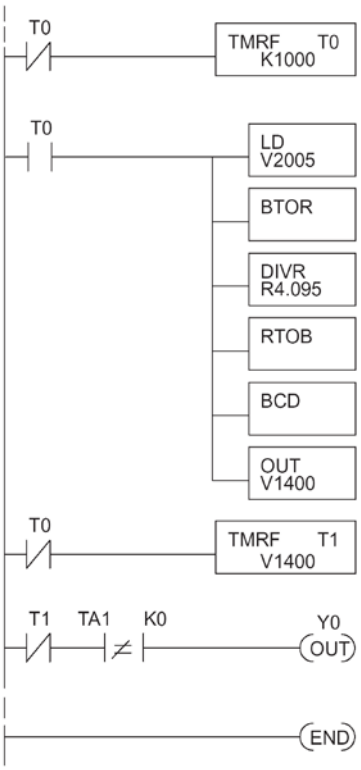
Программа данного примера для управления с помощью включения/выключения использует два таймера. Используются следующие **предположения**, которые могут меняться для конкретного приложения:

- Таблица контура начинается в V2000, поэтому управляющий выход находится по адресу V2005.
- Для управляющего выхода используется 12-битовый униполярный формат данных (0 – FFF).
- Временная развертка (время одного цикла) для временной диаграммы включения/выключения равна 10 с. Это число должно быть приблизительно равно периоду отсчетов контура. Используется быстрый таймер (0.01 с/единицу счета), считающий до 10.00. Это позволяет задавать включение/выключение с разрешением 1/1000. Если частота опроса вашего контура намного выше, понадобится пропорционально уменьшить разрешение для управляющего выхода (и, возможно, перейти к непрерывному управлению).
- Выходом, управляемым с помощью включения/выключения, является Y0.

Управляющая программа должна «согласовать» дискретность выхода с дискретностью временного интервала.

Временной интервал (время одного цикла) для временной диаграммы включения/выключения равен 10 с.

ПРИМЕЧАНИЕ: *Некоторые процессы изменяются слишком быстро и способ широтно-импульсного управления к ним не применим. Учитывайте скорость процесса, когда выбирается метод управления. Применяйте непрерывное управление для процессов, которые меняются слишком быстро для метода широтно-импульсного управления.*



Использует для основной временной развертки быстрый таймер (с разрешением 0.01 с). K1000 обеспечивает предварительно установленное значение 10 с. Нормально-закрытый контакт TO реализует автоматический сброс. TO каждые 10 секунд истинно для одного сканирования. В начале 10-секундного периода TO истинно. Из ячейки v+05, V2005, таблицы контура загружается двоичное значение управляющего выхода.

Преобразует это значение в действительное число, так как необходимо выполнить некоторое масштабирование. Значение управляющего выхода делится на 4.095. Это преобразует используемый диапазон 0 - 4095 в 0 -1000, приводя его в соответствие с 10-секундным диапазоном таймера.

Преобразует действительное число обратно в двоичное, так как инструкции, преобразующей действительное число в BCD — не существует.

Преобразует число - содержимое аккумулятора в формат BCD, чтобы обеспечить соответствие требуемому формату таймера.

Выводит результат в V1400. Это (в нашем примере) произвольно заданная ячейка, содержащая установку для второго таймера.

Второй быстрый таймер также подсчитывает единицы счета (0,01 с) от 0 до 1000, что соответствует 10 секундам. Однако его выход, T1, переходит от Вкл к Выкл по достижении заданного значения.

Выход таймера инвертируется, поэтому управляющий выход включения/выключения оказывается включен в начале 10-секундной временной развертки. Y0 выключается, когда T1 отработает свое задание. Контакт STRNE препятствует включению Y0 в течение одного цикла, когда T0 сбрасывает T1. Y0 – это фактически управляющий выход.

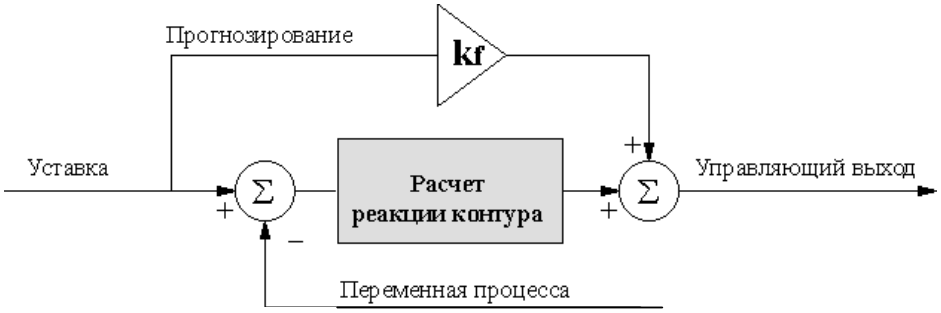
Обмотка END обозначает окончание основной программы

Управление с упреждением

Основный принципы

Управление с упреждением (по возмущению) – Feedforward Control представляет собой улучшение стандартного управления по отклонению в замкнутом контуре (которое было описано ранее). Оно подходит для уменьшения влияния, поддающегося количественному определению, и предсказуемого возмущения или резкого изменения уставки.

DL250-1 и DL260 поддерживают возможность управления по возмущению. Однако лучше сначала реализовать и настроить контур без упреждения, добавляя упреждение только для улучшения характеристик цикла. Термином «по возмущению» называется используемый метод управления, показанный на следующей схеме. Входное значение уставки подается напрямую, в обход уравнения ПИД-регулятора, и складывается с выходом.

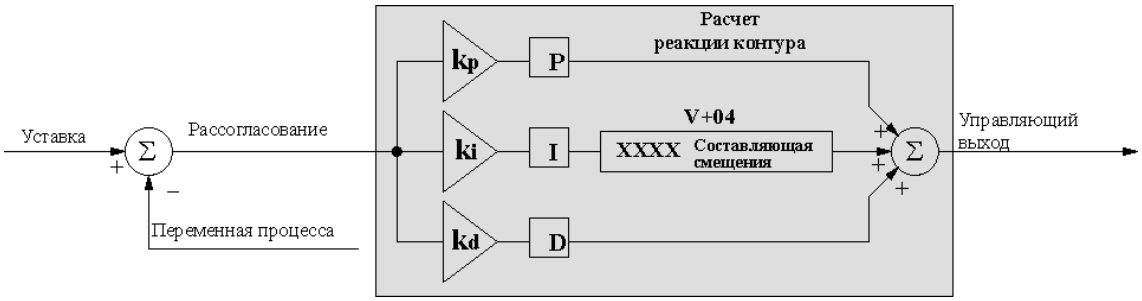


В предыдущем разделе при рассмотрении составляющей смещения, говорилось, что значение этой составляющей задает для управляющего выхода «рабочую область» или рабочую точку. **При колебании рассогласования вокруг нулевого значения, величина выхода колеблется вокруг значения смещения.** Теперь при изменении задания возникает рассогласование, и рабочая точка выхода должна измениться. К такому же результату приводит возмущение, вызывающее новое смещение контура. В действительности контуру «неизвестно, как перейти» к новой рабочей точке. Смещение должно по шагам увеличиваться/уменьшаться до исчезновения рассогласования, и так будет найдена новая рабочая точка.

Предположим, что нам известно о резком изменении задания (обычный случай для некоторых приложений). Если мы можем быстро перевести выход в новую рабочую точку, мы сможем значительно снизить первоначальное отклонение. Если известно (из предыдущего тестирования), какой станет после изменения задания рабочая точка (значение смещения), можно искусственно прямо изменить выход (что и является упреждением). Упреждение обеспечивает следующие преимущества:

- При предсказуемых изменениях задания или возмущениях смещения контура уменьшается рассогласование SP-PV.
- Правильное использование упреждения позволяет уменьшить коэффициент интегрального усиления. Уменьшение коэффициента интегрального усиления дает нам более устойчивую систему управления.

В контроллере контура процессорных модулей DL250-1 и DL260, как показано ниже, использовать упреждение очень легко. Пользователь получает доступ к составляющей смещения, расположенной в специальной ячейке чтения/записи, ячейке таблицы параметров ПИД-регулятора addr+04.

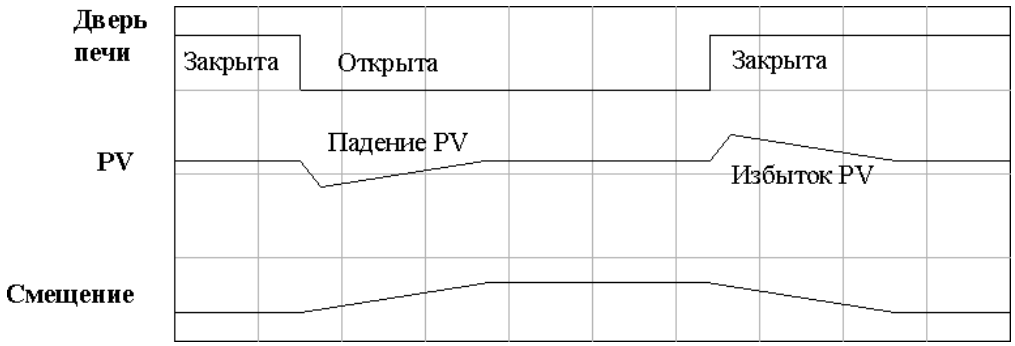


Для изменения смещения (рабочей точки) программе нужно только записать требуемое значение в addr+04. При расчете контура с ПИД-регулятором значение смещения считывается из addr+04 и изменяется в зависимости от текущего расчета коэффициента интегратора. Затем результат записывается обратно в ячейку addr+04. Такой порядок обеспечивает «прозрачность» составляющей смещения. Для реализации регулирования по возмущению нужно только вовремя записать правильное значение составляющей смещения (как показано в следующем примере).

ПРИМЕЧАНИЕ: *При записи составляющей смещения нужно быть аккуратным, обеспечивая только однократную запись значения в момент возникновения новой рабочей точки смещения. Если программа записывает значение смещения при каждом сканировании, то интегратор контура по сути дела отключается*.

Пример управления с упреждением

Но когда и какое значение составляющей смещения записывать? Предположим, что мы используем контур управления температурой печи и уже настроили контур для оптимальной производительности (см. следующий рисунок). Обратите внимание, что когда оператор открывает дверь печи, температура немного падает, пока смещение контура не подстроится с учетом потери тепла. Затем, когда дверь закрывается, температура превышает SP, пока контур не подстроится заново. Управление по возмущению может помочь уменьшить этот эффект.



Сначала запишем величину изменения смещения, создаваемого контроллером контура при открытии или закрытии двери. Затем напишем программу, контролирующую положение концевого выключателя двери печи. При открытии двери наша программа считывает текущее значение смещения из addr+04, добавляет нужное изменение и записывает результат обратно в addr+04. Когда дверь закрывается, мы повторяем процедуру, но при этом вычитаем нужное изменение. Результаты показаны на следующем рисунке.

Ступенчатые изменения смещения являются результатом записи наших двух упреждающих составляющих смещения. Можно заметить, что отклонения PV заметно уменьшаются. Такой же метод можно использовать и для изменений уставки.

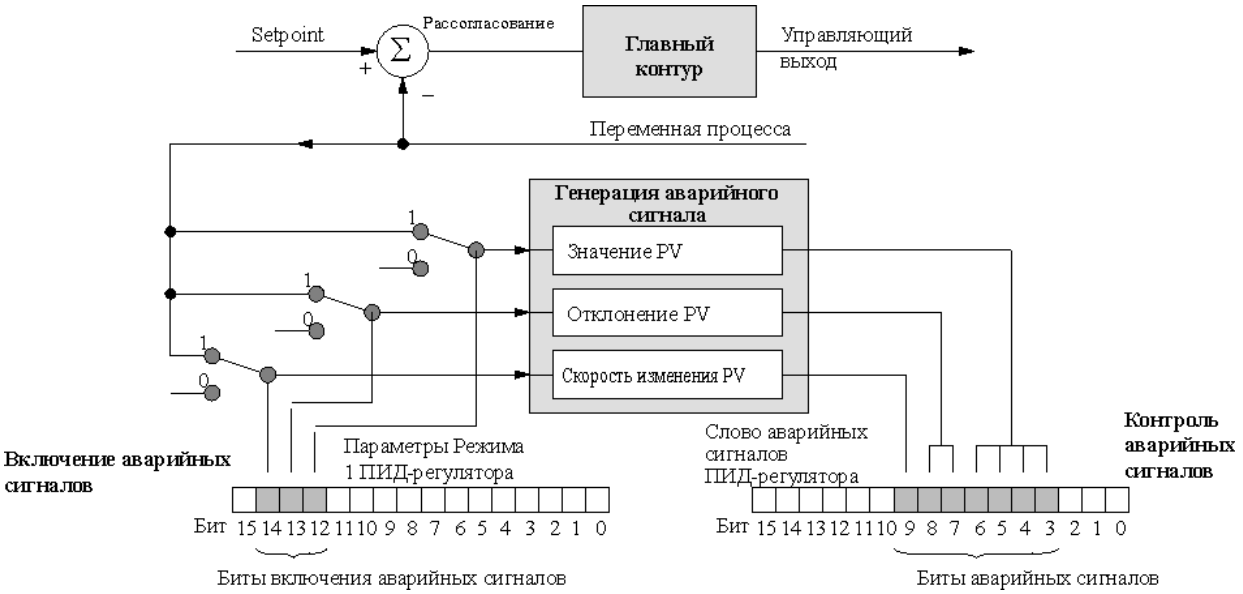
Аварийные сигналы управления процесса

Эффективность контура управления процессом в общем случае можно измерить, проверяя, насколько переменная процесса соответствует уставке. В промышленности большинство контуров управления процессом работают непрерывно и могут потерять управление PV из-за какой-нибудь ошибки. Аварийные сигналы процесса очень важны для раннего обнаружения сбоя контура и могут предупредить персонал завода о необходимости перехода на ручное управление или принятия других мер, пока сбой не будет устранен.

Процессорные модули DL250-1 и DL260 обладают развитым набором функций аварийных сигналов для каждого контура:

- **Аварийные сигналы абсолютного значения PV.** Отслеживают значение PV относительно двух значений нижних пределов и двух значений верхних пределов. Аварийные сигналы генерируются при каждом выходе PV за эти заданные пределы.
- **Аварийный сигнал рассогласования PV.** Отслеживает отклонение значения PV относительно SP. Аварийный сигнал генерируется, когда разность между PV и SP превышает заданное значение.
- **Аварийный сигнал скорости изменения PV.** Вычисляет скорость изменения PV и генерирует аварийный сигнал, если превышается заданное значение скорости изменения.
- **Гистерезис аварийных сигналов.** Работает вместе с функциями аварийных сигналов абсолютного значения и рассогласования, устраняя «дребезг» аварийного сигнала вблизи пороговых значений сигналов

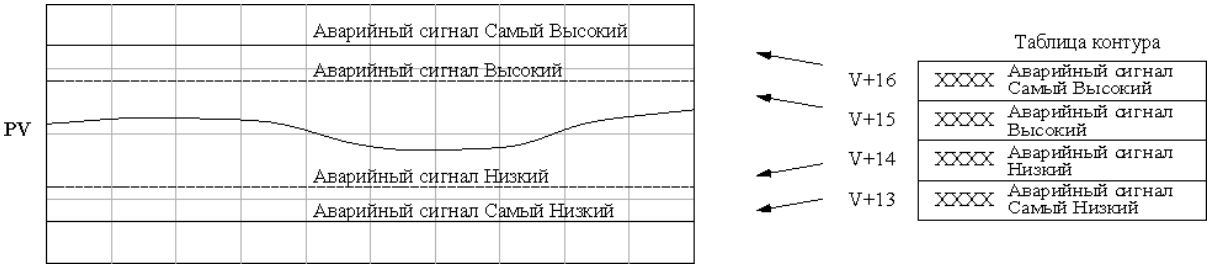
Пороговые значения аварийный сигналов являются полностью программируемыми, и каждый тип аварийного сигнала может включаться и контролироваться независимо. На следующей схеме показана функция контроля PV. Аварийные сигналы включают/выключают биты 12, 13 и 14 Слова 1 (addr+00) в таблице параметров контура. Диалоговые окна настройки Просмотр ПИД-регулятора (PID View) в *Direct*SOFT позволяют легко программировать, включать и контролировать аварийные сигналы. Программа на релейной логике может контролировать статус аварийного сигнала, проверяя биты 3-9 слова Режим ПИД-регулятора и Статус аварийного сигнала (слово addr+06 в таблице контура).



В отличие от расчета ПИД-коэффициентов аварийные сигналы работают всегда, когда процессор находится в рабочем режиме. Контур может находиться в ручном, автоматическом или каскадном режиме, и аварийные сигналы будут работать, если соответствующие биты установлены в «1»(см. рисунок выше).

Аварийные сигналы абсолютного значения PV

Аварийные сигналы абсолютного значения PV делятся на два верхних и два нижних аварийных сигнала. Аварийный сигнал имеет состояние выключено, пока значение PV остается в области между верхними и нижними аварийными сигналами, как показано ниже. Ближайшие к безопасной зоне аварийные сигналы называются высоким сигналом (High Alarm) и низким сигналом (Low Alarm). Если контур потеряет управление, PV сначала пересечет один из этих порогов. Следовательно, можно задать соответствующие значения порогов сигналов в ячейках таблицы контура, показанных на рисунке справа. Формат данных совпадает с форматом PV и SP (12-битовый или 15-битовый). Пороговые значения данных аварийных сигналов устанавливаются так, чтобы заранее предупредить оператора о выходе процесса из-под контроля.

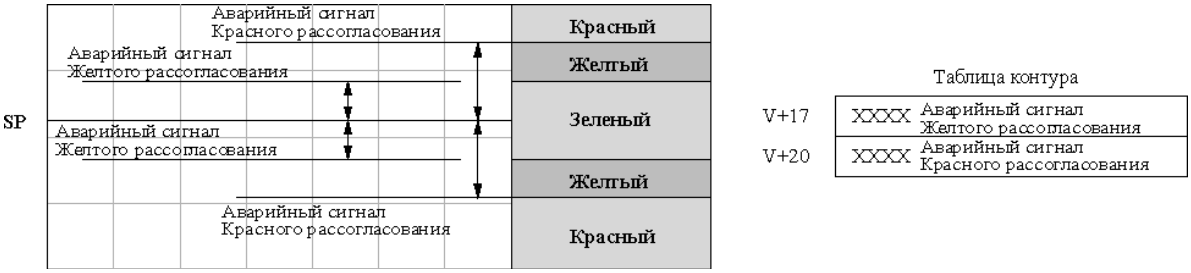


Если процесс останется какое-то время неуправляемым, PV в конце концов пересечет один из внешних порогов, которые называются верхним аварийным сигналом (High-High Alarm – Аварийный сигнал Самый высокий) и нижним аварийным сигналом (Low-Low Alarm – Аварийный сигнал Самый низкий). Их пороговые значения задаются с помощью приведенных выше регистров таблицы контура. Аварийный сигнал Самый Высокий или Самый Низкий сообщает о возникновении серьезного сбоя и требует немедленного вмешательства оператора.

Аварийные сигналы абсолютного значения PV индицируются четырьмя битами слова Режим ПИД-регулятора и Статус аварийных сигналов в таблице контура, как показано справа. Настоятельно рекомендуется программно контролировать эти биты, что легко выполняется с помощью команд бит-из-слова. Кроме того, аварийные сигналы ПИД-регулятора можно контролировать, используя *Direct*SOFT.

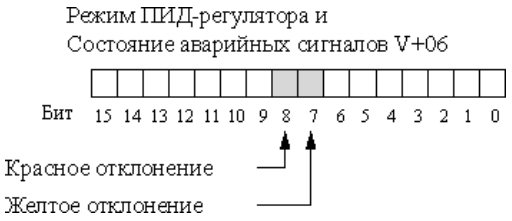
Аварийные сигналы рассогласования PV

Аварийные сигналы рассогласования PV контролируют отклонение PV от значения SP. Аварийный сигнал рассогласования использует два задаваемых порога, и каждый порог применяется как выше, так и ниже текущего значения SP. На следующем рисунке меньший аварийный сигнал рассогласования, называемый «Желтым рассогласованием» («Yellow Deviation»), сообщает о возникновении предупреждающего условия контура. Больший аварийный сигнал, называемый «Красным рассогласованием» («Red Deviation»), сообщает о возникновении значительной ошибки контура. Пороговые значения хранятся в ячейках addr+17 и addr+20 таблицы параметров контура.



Эти пороги определяют зоны, которые колеблются вместе со значением SP. Зеленая зона, окружающая значение SP, представляет собой безопасную область (аварийные сигналы отсутствуют). Желтые зоны лежат за пределами зеленой, а самыми внешними являются красные зоны.

Аварийные сигналы отклонения PV индицируются двумя битами слова Режим ПИД-регулятора и Статус аварийных сигналов в таблице контура, как показано на рисунке справа. Настоятельно рекомендуется контролировать эти биты с помощью релейной программы, что легко выполняется с помощью команд бит-из-слова. Кроме того, аварийные сигналы ПИД-регулятора можно контролировать, используя *Direct*SOFT.



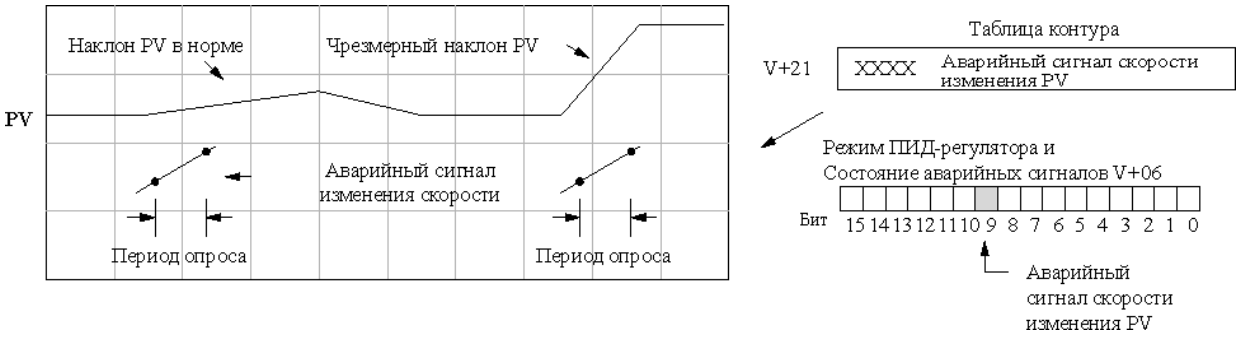
Аварийный сигнал рассогласования PV может включаться и отключаться независимо от других аварийных сигналов PV с помощью бита 13 слова addr+00 параметров режима ПИД-регулятора.

Помните, что вместе с аварийными сигналами рассогласования и абсолютного значения PV может включаться функция запаздывания сигнала, обсуждаемая в конце данного раздела.

Аварийный сигнал скорости изменения PV

Мощным средством раннего оповещения о сбое процесса является контроль скорости изменения переменной PV. Инерционность большинства технологических процессов велика, и значения PV меняются медленно. Относительно быстрое изменение PV является результатом обрыва сигнального провода управляющего выхода или PV, ошибочного значения SP или других причин. При быстрой и эффективной реакции оператора на аварийный сигнал скорости изменения PV абсолютное значение PV не достигнет аварийного значения.

Контроллер контура DL250-1 и DL260 позволяет, как показано ниже, задавать аварийный сигнал скорости изменения PV. Скорость изменения задается как изменение единиц PV за период опроса контура. Это значение записывается в ячейку addr+21 таблицы контура.



Например, пусть PV — это температура нашего процесса, и нам нужен аварийный сигнал, сообщающий, что температура меняется быстрее, чем 15 градусов/минуту. Мы должны знать число единиц счета PV на градус и период отсчетов контура. Теперь предположим, что значение PV (в ячейке addr+03) обозначает 10 единиц счета на градус, а период отсчетов контура составляет 2 секунды. Для преобразования наших инженерных единиц в единицы счета/период отсчетов воспользуемся следующей формулой:

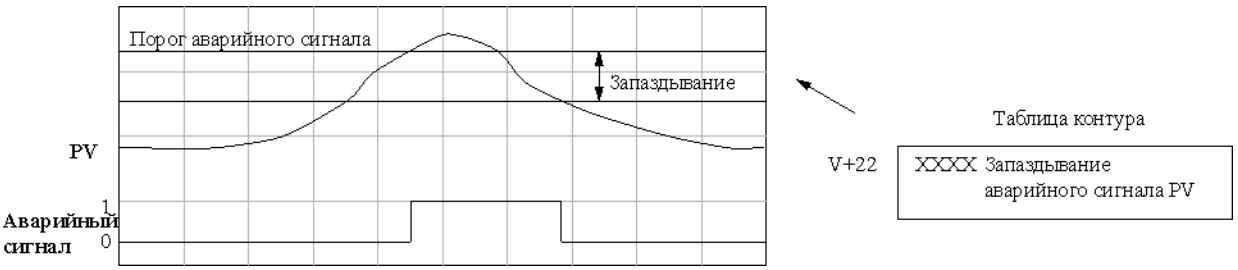
$$\text{Аварийное значение скорости изменения} = \frac{\text{15 градусов}}{\text{1 минута}} \times \frac{\text{10 единиц счета / градус}}{\text{30 опросов контура / минута}} = \frac{150}{30} = 5 \text{ единиц счета / период опроса}$$

Исходя из результатов вычислений, запишем для скорости изменения в таблицу контура значение «5». Аварийный сигнал изменения скорости PV может включаться и отключаться независимо от других аварийных сигналов PV с помощью бита 14 Слова 1 (addr+00) параметров режима ПИД-регулятора. Функция запаздывания сигнала (обсуждаемая ниже) не влияет на Аварийный сигнал скорости изменения.

Гистерезис аварийных сигналов PV

Аварийные сигналы абсолютного значения PV и рассогласование PV программируются с помощью пороговых значений. Когда абсолютное значение или рассогласование превышает порог, устанавливается бит статуса аварийного сигнала. Реальные сигналы PV подвержены некоторому шуму, который приводит к некоторым колебаниям значения PV. Когда значение PV пересекает порог аварийного сигнала, его колебания заставляют сигнал пульсировать, что раздражает операторов процесса. Решение состоит в том, чтобы воспользоваться функцией гистерезиса (запаздывания) аварийных сигналов PV.

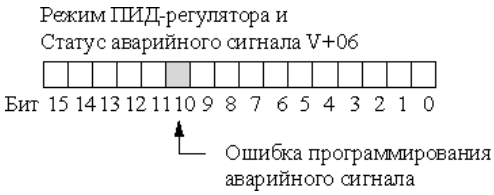
Значение гистерезиса аварийных сигналов PV принимает значения от 1 до 200 (шестнадцатеричное). При использовании аварийного сигнала отклонения PV, заданное значение запаздывания должно быть меньше заданного значения отклонения. На следующем рисунке показано, как работает гистерезис, когда значение PV пересекает порог туда и обратно.



Значение гистерезиса используется после пересечения порога по направлению к безопасной зоне. Таким образом, аварийный сигнал активизируется сразу же после пересечения заданного порога. Его отключение задерживается, пока значение PV не отойдет от порога в безопасную зону на значение гистерезиса.

Ошибка программирования аварийного сигнала

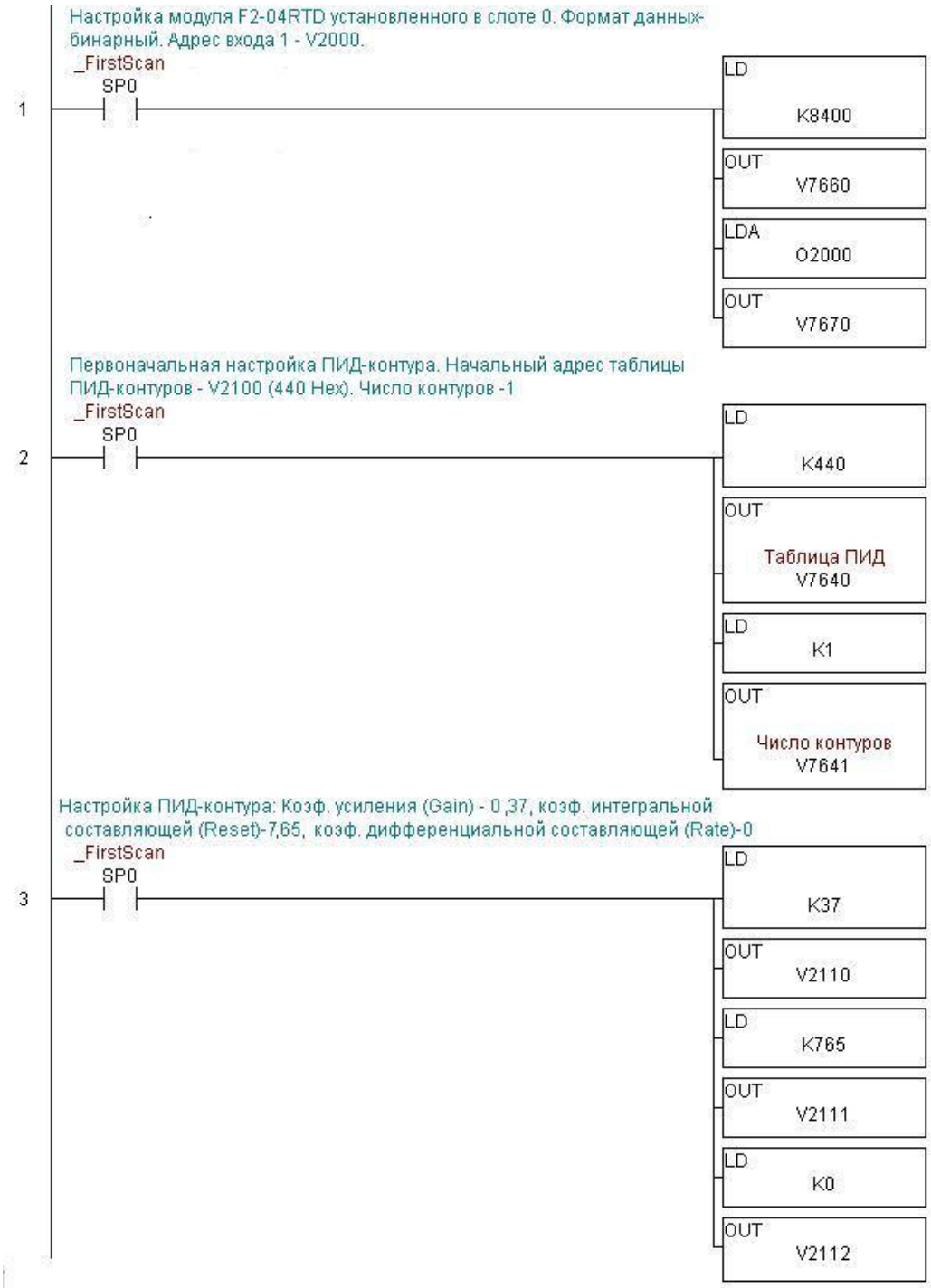
Чтобы значения порогов аварийных сигналов PV были заданы правильно, они должны подчиняться некоторым математическим соотношениям, перечисленным ниже. Если эти требования не выполняются, устанавливается, как показано справа, бит Ошибки программирования аварийного сигнала.

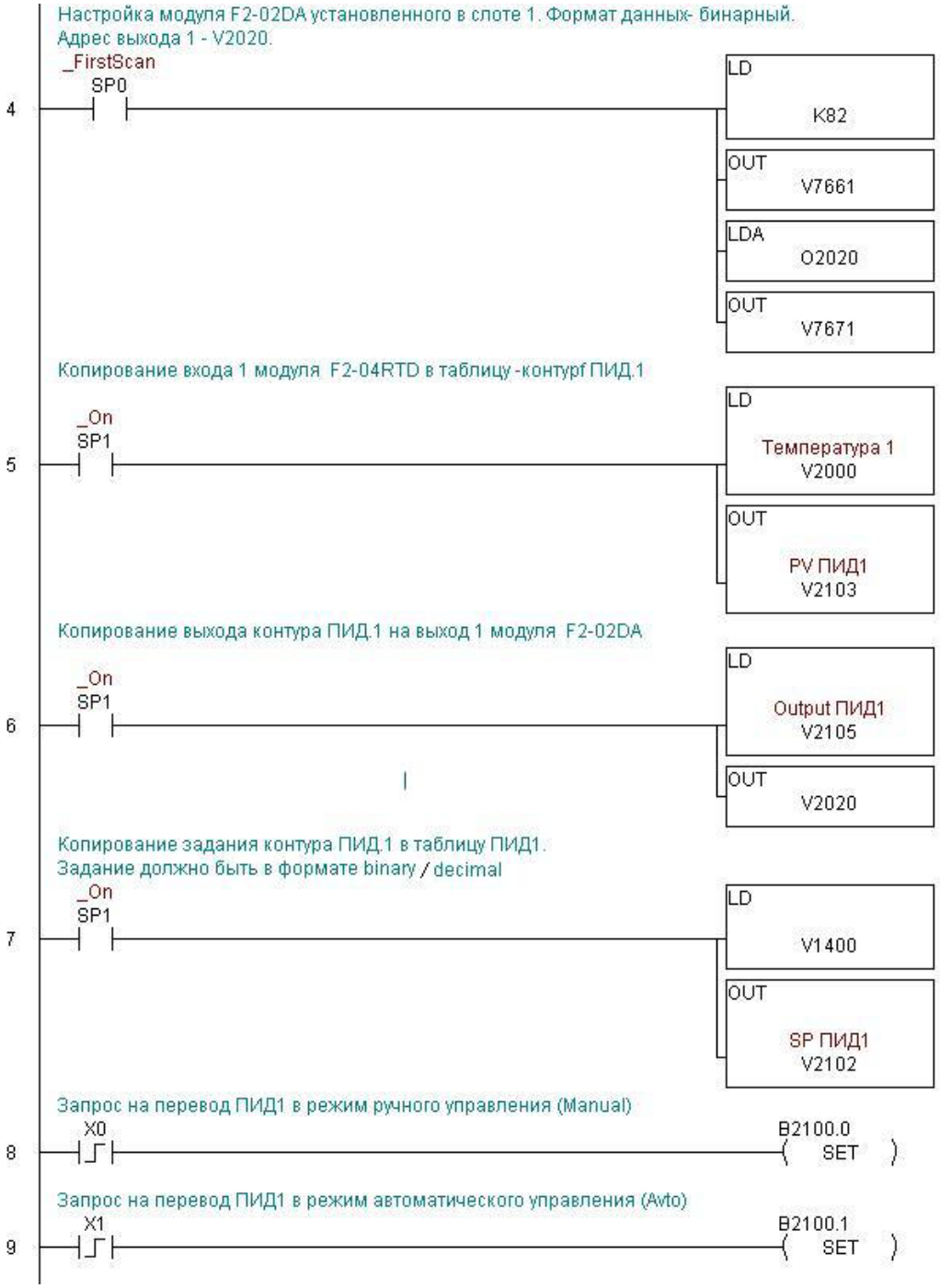


- Требования к аварийному сигналу абсолютного значения PV: Самый Низкий < Низкий < Высокий < Самый Высокий
- Требования к аварийному сигналу рассогласования PV: Желтый < Красный

Пример программы настройки ПИД-контура

Первоначальную настройку ПИД-контура (выбор числа контуров и начального адреса таблиц ПИД-контуров) можно произвести, используя механизм DirectSOFT: *PLC* > *Setup* > *PID* или вводя эти настройки при первом сканировании программы, как показано в этом примере (цепь 2). Затем необходимо настроить модули аналогового ввода и вывода (в примере использованы модули F2-04RTD и F2-02DA-2) и организовать обмен данными между модулями и таблицей контура ПИД1.
Все другие настройки контура (например, направление действия выхода, ограничение выхода и др.) должны быть введены в теле программы или при помощи DirectSOFT для обеспечения корректной работы контура.





Примечание переводчика: в этой программе использованы одновременно два способа настройки ПИД- контура: Setup в DirectSOFT и в теле программы. Функционирование контура зависит, в том числе, от способа настройки. Так, например, если за время работы возникла необходимость изменения настроек и Вы сделали это, то после перехода из режима программирования в RUN настройки будут заменены на первоначальные. Поэтому внимательнее подойдите к выбору способа настройки всех компонентов ПИД-контура.

Советы по поиску неисправностей

Контур не переходит в автоматический режим.

Проверьте следующие возможные причины:

- Возник аварийный сигнал PV или ошибка программирования аварийного сигнала PV.
- Контур является главным контуром каскадной пары, а подчиненный контур не находится в каскадном режиме.

Когда контур находится в автоматическом режиме, управляющий выход все время остается нулевым.

Проверьте следующие возможные причины:

- Верхний предел управляющего выхода в ячейке addr+31 таблицы контура равен нулю.
- Контур перешел в насыщение, так как ошибка никогда не становится равной нулю и меняет (алгебраический) знак.

Управляющий выход не равен нулю, но он неправильный.

Проверьте следующие возможные причины:

- Неправильно заданы значения коэффициентов усиления. Помните, коэффициенты усиления в таблице контура задаются в формате BCD, а SP и PV — в двоичном формате. Пакет программирования *Direct*SOFT выводит значения SP, PV, смещения и управляющего выхода в десятичном (BCD) формате, преобразуя их в бинарный формат перед обновлением таблицы контура.

Программный задатчик не работает после установки бита запуска.

Проверьте следующие возможные причины:

- Не установлен бит включения программного задатчика. Проверьте состояние бита 11 ячейки addr+00 таблицы параметров контура. Он должен быть равен 1.
- Установлен бит остановки или другие биты управления программным задатчиком.
- Начальное значение SP и конечное значение SP первого линейного изменения одинаковы, поэтому у первого линейного участка отсутствует наклон и, следовательно, у него нет длительности. Программный задатчик быстро переходит к фиксированному участку, при этом создается иллюзия, что первое линейное изменение не работает.
- Контур находится в каскадном режиме и пытается получить удаленное значение SP.
- Слишком низкое значение верхнего предела SP в ячейке addr+27 таблицы контура.
- Проверьте свою программу, чтобы убедиться, что она ничего не записывает в ячейку SP (addr+02 в таблице контура). Быстро это можно сделать, временно помещая конечную обмотку в начало вашей программы, затем переводя ПЛК в рабочий режим и вручную запуская программный задатчик.

Значение PV в таблице остается постоянным, хотя аналоговый модуль принимает сигнал PV.

Ваша релейная программа должна успешно считывать аналоговое значение и записывать его в ячейку addr+03 таблицы контура. Убедитесь, что аналоговый модуль генерирует значение, и что программа работает.

Кажется, что дифференциальный коэффициент усиления никак не влияет на выход.

Возможно, включено дифференциальное ограничение (см. раздел об ограничении дифференциального коэффициента).

Кажется, что задание контура изменяется само по себе.

Проверьте следующие возможные причины:

- Включен программный задатчик, который и генерирует уставки.
- Если такой эффект возникает при переходе контура из ручного в автоматический режим, контур автоматически устанавливает SP=PV (функция безударного перехода).
- Проверьте свою программу, чтобы убедиться, что она ничего не записывает в ячейку SP (addr+02 в таблице контура). Быстро это можно сделать, временно помещая команду END в начале вашей программы и затем переводя ПЛК в рабочий режим.

Значения SP и PV, вводимые из *Direct*SOFT, работают правильно, но эти же значения, записываемые из пользовательской программы, работают неправильно.

Окно Просмотр ПИД-регулятора (PID View) в *Direct*SOFT позволяет вводить значения SP, PV и смещения в десятичном виде, для удобства также отображая их в десятичном виде. Например, при использовании 12-битового униполярного формата данных значения лежат в диапазоне от 0 до 4095. Однако, в таблице контура эти значения хранятся в шестнадцатеричном виде, поэтому *Direct*SOFT выполняет необходимые преобразования. При использовании 12-битового униполярного формата значения в таблице лежат в диапазоне от 0 до FFF.

Контур кажется неустойчивым и не поддающимся настройке независимо оттого, какие коэффициенты усиления используются.

Проверьте следующие возможные причины:

- Период опроса контура слишком велик. Обратитесь к разделу в начале данной главы, чтобы выбрать интервал обновления контура.
- Коэффициенты усиления слишком велики. Начните с уменьшения до нуля дифференциального коэффициента. Затем при необходимости уменьшите интегральный и пропорциональный коэффициенты.
- В вашем процессе слишком велика задержка передачи. Это означает, что PV медленно реагирует на изменения управляющего выхода. Причиной этого может быть слишком большое «расстояние» между исполнительным механизмом и датчиком PV, или мощность исполнительного механизма недостаточна.
- Может существовать возмущение процесса, которое контур не может преодолеть. Убедитесь, что PV относительно устойчиво при постоянном SP.

Словарь терминов ПИД-регулирования

Аварийный сигнал абсолютного значения PV (PV Absolute Alarm)	Программируемый аварийный сигнал, сравнивающий значение PV с заданными пороговыми значениями аварийного сигнала.
Аварийный сигнал отклонения PV (PV Deviation Alarm)	Программируемый аварийный сигнал, сравнивающий разность значений SP и PV с заданным пороговым значением рассогласования.
Автоматический режим (Automatic Mode)	Режим работы контура, в котором контур выполняет расчет уравнения ПИД-регулятора и изменяет управляющий выход.
Безударный переход (Bumpless Transfer)	Метод изменения режима работы контура, позволяющий избежать резкого изменения уровня управляющего выхода за счет искусственного приравнивания в момент изменения режима значений задания (SP) и переменной (PV), или составляющей смещения и управляющего выхода.
Взвинчивание (выбег) интегральной составляющей (Reset Windup)	Условие, возникающее, когда контур не может найти равновесие, и постоянная ошибка приводит к чрезмерному увеличению (взвинчиванию) суммы интегратора. Взвинчивание интегральной составляющей вызывает дополнительную задержку при устранении сбоя первоначального контура.
Главный контур (Major Loop)	В каскадном управлении это контур, генерирующий уставку - SP для каскадного контура.
Дискретное управление (On/Off Control)	Простой метод управления процессом с помощью включения/выключения подачи энергии в систему. За счет инерционности процесса для относительно медленноменяющейся PV влияние включения/выключения усредняется. Непрерывный выход DL250 преобразуется в управление включением/выключением с помощью небольшой программы.
Дифференциальный коэффициент усиления (Derivative Gain)	Константа, определяющая величину дифференциальной составляющей ПИД-регулятора, соответствующую текущему рассогласованию.
Зона нечувствительности рассогласования (Error Dealband)	Необязательная возможность, которая делает контур нечувствительным к небольшим отклонениям. Размер зоны нечувствительности задается пользователем.
Интегральный коэффициент усиления (Integral Gain)	Константа, определяющая величину интегральной составляющей ПИД-регулятора, соответствующую текущему рассогласованию.
Интегратор (Reset)	Интегральное звено. Интегральная составляющая добавляет каждую текущую ошибку к предыдущей, поддерживая вычисление текущей суммы, называемой смещением.
Каскадные контура (Cascaded Loops)	Каскадный контур получает значение задания с выхода другого контура. Каскадные контуры связаны соотношением главный/подчиненный и работают совместно, в конечном счете, управляя одной PV.
Каскадный режим Cascade Mode)	Режим работы контура, в котором контур получает значение SP с выхода другого контура.
Квадрат отклонения Error Squared)	Необязательная функция, которая умножает отклонение само на себя, но сохраняет первоначальный знак. Она уменьшает влияние малых отклонений, увеличивая влияние больших.
Контур обратного действия (Reverse-Acting Loop)	Контур, в котором PV увеличивается в ответ на уменьшение управляющего выхода. Другими словами, коэффициент усиления процесса отрицателен.
Контур ПИД-регулирования (PID-Loop)	Математический метод управления замкнутым контуром, включающий сумму трех составляющих, которые используют значения пропорционального, интегрального и дифференциального отклонений. Коэффициенты усиления трех составляющих - независимые константы, они позволяют оптимизировать (настраивать) контур для конкретной физической системы.
Контур прямого действия (Direct-Acting Loop)	Контур, в котором PV возрастает с ростом управляющего выхода. Другими словами, коэффициент усиления процесса является положительным.
Непрерывное управление (Continuous Control)	Управление процессом, использующее в качестве управляющего выхода непрерывный (аналоговый) сигнал.
Отклонение PV на участке выдержки (Soak Deviation)	Отклонение PV на участке выдержки — это мера разности между SP и PV на участке выдержки профиля программного задатчика.
Переменная процесса (PV) (Process Variable)	Количественная мера физического свойства материала, которое влияет на качество окончательного продукта. Используется для контроля и управления.
Переход (Transfer)	Изменение одного режима работы контура на другой (ручной, автоматический или каскадный). Слово «переход» относится к переходу управления управляющим выходом или SP, в зависимости от конкретного изменения режима.
Период опроса (Sampling Time)	Интервал между вычислениями ПИД-регулятора. Метод, с помощью которого процессор управляет процессом, называется квазианалоговым управлением, так как он выполняет расчеты только периодически.

Подчиненный контур (Minor Loop)	В каскадном управлении это контур, получающий уставку SP от главного контура.
Позиционный алгоритм (Алгоритм по отклонению) (Position Algotithm)	Управляющий выход рассчитывается в соответствии с позицией (отклонением) PV относительно SP.
Предварение, управление по возмущению (Feedforward)	Метод оптимизации управляющей реакции контура на известное изменение уставки или возмущение, влияние которых на составляющую смещения поддается количественному определению.
Программный задатчик (Ramp/Soak Profile)	Набор значений SP, называемый профилем, генерируемых в реальном времени при каждом расчете контура. Профиль состоит из последовательности пар участков наклон/выдержка, заметно упрощающих задачу программирования ПЛК для генерации подобных последовательностей SP.
Пропорциональный коэффициент усиления (Proportional Gain)	Константа, определяющая величину пропорциональной составляющей ПИД-регулятора, соответствующую текущему рассогласованию.
Процесс (Process)	Производственная процедура, увеличивающая стоимость сырья. В частности управление процессом связано с внесением в ходе процесса в сырье *химических* изменений.
Рассогласование (Error)	Разность значений SP и PV, Рассогласование (отклонение, ошибка)) = SP – PV.
Ручной режим (Manual Mode)	Режим работы контура, в котором расчет ПИД-регулятора прекращается. Оператор вручную управляет контуром, напрямую записывая значение управляющего выхода.
Скоростной ПИД-алгоритм, алгоритм по скорости (Velocity Algorithm)	Управляющий выход рассчитывается так, чтобы представлять изменение (скорость), с которой PV становится равной SP.
Скорость (Rate)	Другое название – дифференцирующее звено, составляющая скорости, которая реагирует на *изменение* рассогласования.
Составляющая смещения (Bias Term)	В позиционной форме уравнения ПИД-регулятора это сумма интегрального члена и начального значения управляющего выхода.
Ступенчатый отклик (Step Response)	Поведение переменной процесса в ответ на ступенчатое изменение SP (во время работы замкнутого контура), или ступенчатое изменение управляющего выхода (во время работы разомкнутого контура)
Удаленная уставка (Remote Setpoint)	Ячейка, из которой контур считывает свою уставку, если он является подчиненным контуром в каскадной включении контуров.
Управляющий выход (Control Output)	Численный результат решения уравнения ПИД-регулятора, выдаваемый контуром с целью достижения нулевого текущего рассогласования.
Уставка (SP) (Setpoint)	Требуемое значение переменной процесса. Уставка (SP) — это сигнал, подаваемый на вход контроллера контура при работе замкнутого контура**.**
Фиксация смещения (Bias Freeze)	Метод сохранения значения смещения (рабочей точки) управляющего выхода путем замораживания значения интегральной составляющей, когда выход выходит за пределы диапазона. Преимуществом является быстрое восстановление контура.

Глава 9. Обслуживание и поиск неисправностей

В этой главе…

- Обслуживание технических средств
- Диагностика
- Индикаторы состояния процессора
- Неисправности в коммуникациях
- Поиск неисправностей в модулях ввода/вывода
- Поиск и устранение помех
- Запуск машин и поиск ошибок в программах

Обслуживание технических средств

Нормальное обслуживание

Контроллеры DL205 не требуют регулярного обслуживания, необходимо только проводить периодические проверки, чтобы уменьшить риск возникновения неисправностей. Регулярные проверки должны производиться, учитывая два момента:

- Состояние окружающей среды (температура помещения, вентиляция и др.)
- Батареи процессора.

Поддержание состояния окружающей среды

Состояние окружающей вашу систему среды может влиять на рабочие характеристики системы. Если ваша система размещена в шкафу, температура воздуха в нем должна быть в пределах рабочего диапазона температур. Если в шкафу установлены фильтры, очистите их, а при необходимости замените, чтобы обеспечить достаточный воздушный поток. Рекомендуется производить проверку окружающей среды один раз в два месяца. Обеспечьте работу вашей системы в рамках установленных технических условий.

Индикатор низкого напряжения батареи

У модуля процессора есть светодиод «BATT», который показывает, когда напряжение на батарее низкое. Вы должны периодически контролировать, нужна ли замена батареи. Низкое напряжение на батарее можно определить с помощью программ процессора. Специальное реле SP43 включается, когда требуется замена батареи.

Если у вас процессор DL240, вы можете измерить напряжение на батарее. V7746 — ячейка напряжения батареи. Например, значение 32 в ячейке V7746 показывает, что напряжение на батарее равно 3.2 В.

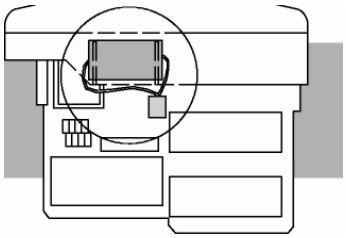
Замена батареи ЦПУ

Батареи процессорного модуля используются для сохранения V-памяти программ и системных параметров. Срок службы батареи — 5 лет.

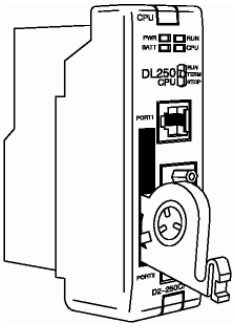
ПРИМЕЧАНИЕ: *Перед установкой или заменой батареи процессорного модуля сохраните данные в V-памяти и системные параметры. С помощью DirectSOFT вы можете сохранить программу, данные из V-памяти, системные параметры на жестком или гибком диске персонального компьютера*.

Для установки батареи D2-BAT в процессорные модули DL230 или DL240:

1. Осторожно вставьте разъем батареи в штепсельный разъем монтажной платы.
2. Вставьте батарею в фиксатор. Не прикладывайте чрезмерного усилия. Вы можете поломать фиксатор
3. Отметьте дату установки батареи.



DL230 или DL240



DL250-1 или DL260

Для установки батареи D2-BAT-1 в процессоры DL250-1, DL-260:

1. Прижмите фиксатор на батарейной дверце и поворачивайте, пока батарейная дверца не откроется.
2. Поместите батарею в прорезь наружу плюсом
3. Закройте батарейную дверцу, убедитесь, что она надежно зафиксирована
4. Отметьте дату установки батареи.

ПРЕДУПРЕЖДЕНИЕ: *Не пытайтесь перезарядить батарею или положить батарею вблизи огня. Батарея может взорваться или выделить опасные материалы*.

Диагностика

Диагностика

Ваша система DL205 выполняет несколько программ диагностики при каждом сканировании процессора. Диагностика разработана таким образом, чтобы обнаруживать сбои различного вида в ПЛК. Выделяют два вида ошибок: критические и некритические (неисправимые и исправимые).

Критические ошибки

Критические ошибки (Fatal Error) — это ошибки, которые связаны с риском того, что система не будет правильно или безопасно функционировать. Если процессорный модуль находится в Рабочем режиме и возникает критическая ошибка, то он переключается в Программный режим. (Напоминаем, что в Программном режиме все выходы отключаются). Если критическая ошибка обнаружена, когда процессор находится в Программном режиме, то он не войдет в Рабочий режим до тех пор, пока ошибка не будет исправлена.

Несколько примеров критических ошибок:

- Отказ источника питания каркаса.
- Ошибка четности или сбой процессора.
- Ошибки в конфигурации ввода/вывода.
- Некоторые программные ошибки.

Некритические ошибки

Некритические ошибки (Non Fatal Error) — это такие ошибки, которые требуют внимания, но не приведут к некорректным действиям. Они не могут вызывают переход из Рабочего режима в Программный режим, также как не препятствуют переходу процессора в Рабочий режим. Имеются специальные реле, которые позволяют прикладной программе определить, имела ли место некритическая ошибка. Из прикладной программы можно затем правильно отключить систему или при необходимости переключить процессор в Программный режим.

Несколько примеров исправимых ошибок:

- Низкое напряжение резервной батареи.
- Ошибки всех модулей ввода/вывода.
- Некоторые программные ошибки.
-

Получение диагностической информации

Диагностическую информацию можно найти в разных местах с различной детализацией сообщений.

- Процессорный модуль автоматически заносит коды ошибок и все сообщения об ОШИБКАХ в две отдельные таблицы, которые можно просмотреть с помощью ручного программатора или *Direct*SOFT.
- На Ручной Программатор выводятся номера ошибок и их краткое описание.
- С помощью *Direct*SOFT выводятся номера ошибок и сообщения об ошибках.
- В приложении B данного руководства приведен полный перечень сообщений об ошибках, упорядоченных по их номеру.

Во многих из этих сообщений имеется указание на дополнительные ячейки памяти, в которых находится дополнительная информация об ошибках. Такой памятью является V-память и специальные реле (SP).

Ячейки V-памяти, соответствующие кодам ошибок

В следующих двух таблицах указаны конкретные ячейки памяти, соответствующие определенному типу сообщений об ошибках.

Класс ошибок	Категория ошибок	V-память
Напряжение батареи (только для DL240)	Показывает напряжение батареи x 10 (32 - это 3.2 В)	V7746
Определяемый пользователем	Код ошибки, используемый с командой FAULT	V7751
Конфигурация ввода/ вывода	Правильный идентификационный код модуля	V7752
	Неправильный идентификационный код модуля	V7753
	Номер каркаса и слота, где возникла ошибка	V7754
Системные ошибки	Код критической ошибки	V7755
	Код существенной ошибки	V7756
	Код менее существенной ошибки	V7757
Диагностика модуля	Номер каркаса и слота, где возникла ошибка	V7760
	Всегда поддерживает "0"	V7761
	Код ошибки	V7762
Грамматические	Адрес, в котором имеется синтаксическая ошибка	V7763
	Код ошибки, обнаруженный при синтаксической проверке	V7764
Цикл работы процессора	Число циклов сканирования после последнего перехода из Программного режима в Рабочий	V7765
	Текущее время цикла сканирования (мс)	V7775
	Минимальное время цикла сканирования (мс)	V7776
	Максимальное время цикла сканирования (мс)	V7777

Специальные реле (SP), соответствующие кодам ошибок

Таблица также содержит индикаторы состояния, которые определяют ошибки. За более полной информацией о каждом специальном реле обращайтесь к Приложению D.

Пусковое реле и реле реального времени	
SP0	ВКЛЮЧЕНО только на первом скани-
SP1	Всегда ВКЛЮЧЕНО
SP3	1 минута на часах
SP4	1 секунда на часах
SP5	100 миллисекунд на часах
SP6	50 миллисекунд на часах
SP7	ВКЛЮЧЕНО при чередующемся циклах сканирования
Реле состояния процессора	
SP11	Принудительный рабочий режим (только на DL240)
SP12	Терминальный рабочий режим
SP13	Тестовый рабочий режим (только на DL240)
SP15	Тестовый программный режим (толь-
SP16	Терминальный программный режим
SP20	Команда STOP выполнена
SP22	Прерывание разрешено
Реле текущего контроля системы	
SP40	Критическая ошибка
SP41	Некритическая ошибка
SP42	Низкое напряжение батареи
SP44	Ошибка программной памяти
SP45	Ошибка Ввода/Вывода
SP46	Ошибка связи
SP47	Ошибка в конфигурации Вво-
SP50	Команда FAULT выполнена
SP51	Сторожевой таймер цикла
SP52	Синтаксическая ошибка
SP53	Невозможно разрешить логику
SP54	Ошибка связи интеллектуального мо-

Реле состояния аккумулятора	
SP60	Аккумулятор меньше заданного зна-
SP61	Аккумулятор равен заданному значе-
SP62	Аккумулятор больше заданного значе-
SP63	Результат суммирования в аккумуля-
SP64	Произошел заем части при вычитании
SP65	Произошло заимствование при вычи-
SP66	Произошел перенос части
SP67	Произошел перенос
SP70	Результат отрицательный (по знаку)
SP71	Ошибка ссылки на указатель
SP73	Переполнение
SP75	Данные не в формате BCD
SP76	Загрузка нуля
Реле текущего контроля коммуникаций	
SP116 DL230/DL240	Процессор соединен с другим устройством
SP116 DL250-1/260	Порт 2 соединен с другим устройством
SP1 17	Ошибка соединения порта 2 (только на DL250-1 и
SP120	Модуль занят, слот 0
SP121	Ошибка соединения, слот 0
SP122	Модуль занят, слот 1
SP123	Ошибка соединения, слот 1
SP124	Модуль занят, слот 2
SP125	Ошибка соединения, Слот 2
SP126	Модуль занят, слот 3
SP127	Ошибка соединения, слот 3
SP130	Модуль занят, слот 4
SP131	Ошибка соединения, слот 4
SP132	Модуль занят, слот 5
SP133	Ошибка соединения, слот 5
SP134	Модуль занят, слот 6
SP135	Ошибка соединения, слот 6
SP136	Модуль занят, слот 7
SP137	Ошибка соединения, слот 7

Коды модулей ввода/вывода

Каждый компонент системы имеет кодовый идентификатор ввода/вывода. Этот кодовый идентификатор используется в некоторых сообщениях об ошибках, связанных с модулями ввода/вывода. В следующих таблицах приведены эти коды.

Код (Hex)	Тип компонента
04	Процессорный модуль
03	Каркас ввода/вывода
20	Выход на 8 точек
21	Вход на 8 точек
24	Комбинированный – 4 входа/выхода
28	Выход на12 точек, вход на 16 точек
3F	Вход на 32 точки
30	Выход на 32 точки
52	H2-ERM
51	H2-CTRIO

Код (Hex)	Тип компонента
36	Аналоговый вход
2B	Вход на 16 точек
37	Аналоговый выход
3D	Комбинированный - аналоговый
4A	Интерфейс счетчика
7F	Ненормальный
FF	Модули не обнаружены
EE	D2-DCM H2-ECOM F2-CP128
BE	D2-RMSM

На следующей схеме показано, как используются коды модулей ввода/вывода.

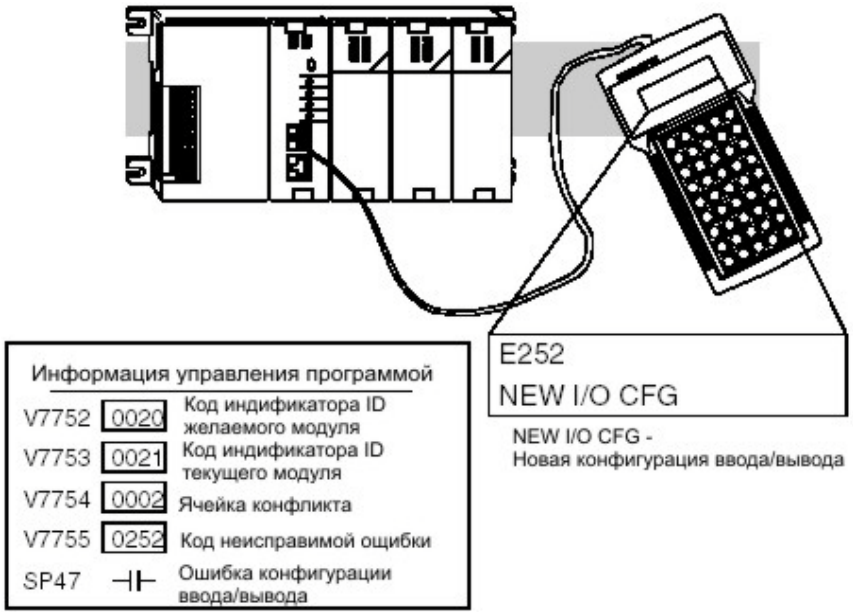


Таблица сообщений об ошибках

Процессорный модуль DL240 автоматически регистрирует коды любых системных ошибок и любые специальные сообщения, которые вы создадите в вашей прикладной программе с использованием команд FAULT. Процессорный модуль регистрирует код ошибки, дату и время, когда произошла ошибка. Имеются две отдельные таблицы, в которых хранится эта информация.

- Таблица кодов ошибок — система регистрирует в этой таблице до 32 ошибок. Когда возникает ошибка, то все уже записанные в таблицу ошибки сдвигаются вниз, а последняя ошибка загружается в верхнюю ячейку. Если таблица полностью заполнена при появлении ошибок, то самая старая ошибка убирается (стирается) из таблицы.
- Таблица сообщений — в этой таблице система регистрирует до 16 сообщений. Когда сообщение инициируется, то все уже записанные в таблицу сообщения сдвигаются вниз, а последнее сообщение загружается в верхнюю ячейку. Если таблица полностью заполнена при появлении сообщения, то самое старое сообщение убирается (стирается) из таблицы.

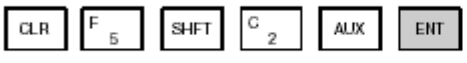
На следующей схеме показан пример таблицы ошибок с сообщениями.

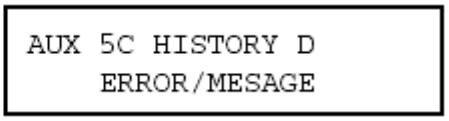
Дата	Время	Сообщение
1993-05-26	08:41:51:11	*Конвейер-2 остановлен
1993-04-30	17:01:11:56	*Конвейер-1 остановлен
1993-04-30	17:01:11:12	*Габарит SW1 нарушен
1993-04-28	03:25:14:31	*Заклинивание пилы

Вы можете получить доступ к таблице кодов ошибок и к таблице сообщений об ошибках с помощью подменю диагностики ПЛК в *Direct*SOFT или через ручной программатор. Более подробная информация по доступу к этим данным можно получить в руководстве пользователя *Direct*SOFT.

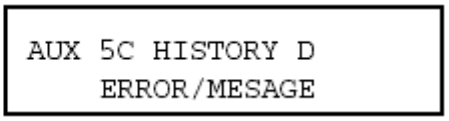
На следующих примерах показывается, как использовать ручной программатор и функцию AUX 5C, чтобы просмотреть коды ошибок. Самый последний код ошибки или сообщение всегда выводятся. Вы можете использовать клавиши PREV или NXT, чтобы просмотреть путем прокрутки эти сообщения.

Использование функции AUX 5C для просмотра таблиц

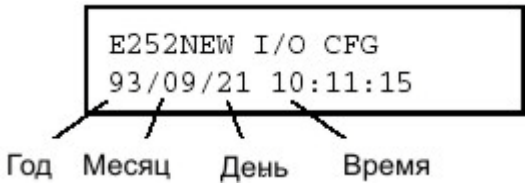




Использование клавиш со стрелками для выбора Ошибок или Сообщений



Пример представления ошибки на экране



Коды системных ошибок

Журнал регистрации системных ошибок содержит 32 последние обнаруженные ошибки. Ошибки, заносимые в журнал ошибок, составляют только часть всех сообщений об ошибках, которые вырабатываются системами DL205. Эти ошибки могут формироваться процессорным модулем или ручным программатором в зависимости от фактической ошибки. В приложении B приводится более полное описание кодов ошибок.

Ошибки могут выявиться в любой момент. Однако, большинство ошибок выявляется при включении питания, при входе в рабочий режим или когда последовательность набранных на ручном программаторе клавиш приводит к ошибке или непредусмотренному запросу.

Код ошибки	Описание
E003	Программный тайм-аут
E004	Неправильная команда (ошибка четности ОЗУ в процессоре)
E041	Низкое напряжение на батарее процессора
E043	Низкое напряжение на батарее картриджа памяти
E099	Превышена программная память
E101	Отсутствие картриджа памяти
E104	Сбой при записи
E151	Неправильная команда
E155	Сбой оперативной памяти (ОЗУ)
E201	Отсутствие контакта с блоком
E202	Отсутствие модуля ввода/вывода
E203	Перегорание предохранителя
E206	Отказ источника питания на 24 В пользователя
E210	Отказ питания
E250	Ошибка связи в цепи ввода/вывода
E251	Ошибка четности ввода/вывода
E252	Новая конфигурация ввода/вывода
E262	Ввод/вывод вне диапазона
E312	Ошибка связи 2
E313	Ошибка связи 3
E316	Ошибка связи 6
E320	Тайм-аут
E321	Ошибка связи
E499	Неправильная запись текста в команде печати
E501	Неверный ввод
E502	Неверный адрес
E503	Неверный оператор
E504	Неверная ссылка/значение
E505	Неправильная команда

Код ошибки	Описание
E506	Неправильная операция
E520	Неверная операция - процессор в Рабочем режиме
E521	Неверная операция - процессор в режиме работы Теста
E523	Неверная операция – процессор в режиме в Тест программы
E524	Неверная операция – процессор в режиме Программирования
E525	Переключатель режима не в положении TERM
E526	Блок отключен
E527	Блок в режиме On-Line
E528	Режим процессора
E540	Процессор блокирован
E541	Ошибочный пароль
E542	Сброс пароля
E601	Память заполнена
E602	Отсутствие команды
E604	Отсутствие ссылки
E610	Неверный тип ввода/вывода
E611	Неверный идентификатор связи
E620	Вне пределов памяти
E621	Память ЭППЗУ не пуста
E622	Нет ЭППЗУ в ручном программаторе
E624	Только V-память
E625	Только программы
E627	Неверная операция записи
E628	Ошибка в типе памяти (должен быть ЭППЗУ)
E640	Несравнимое
E650	Системная ошибка на Ручном Программаторе
E651	Ошибка ПЗУ в Ручном Программаторе
E652	Ошибка ОЗУ в Ручном Программаторе

Коды программных ошибок

В следующей таблице перечисляются коды ошибок рабочего цикла и синтаксические, которые могут присутствовать в программах. Эти ошибки выявляются при переводе Процессора в Программный режим или при использовании вспомогательной функции AUX 21 — Проверка Программ. Процессор также включит SP52 и сохранит ошибку в ячейке V7755. В приложении B приводится более полное описание кодов ошибок.

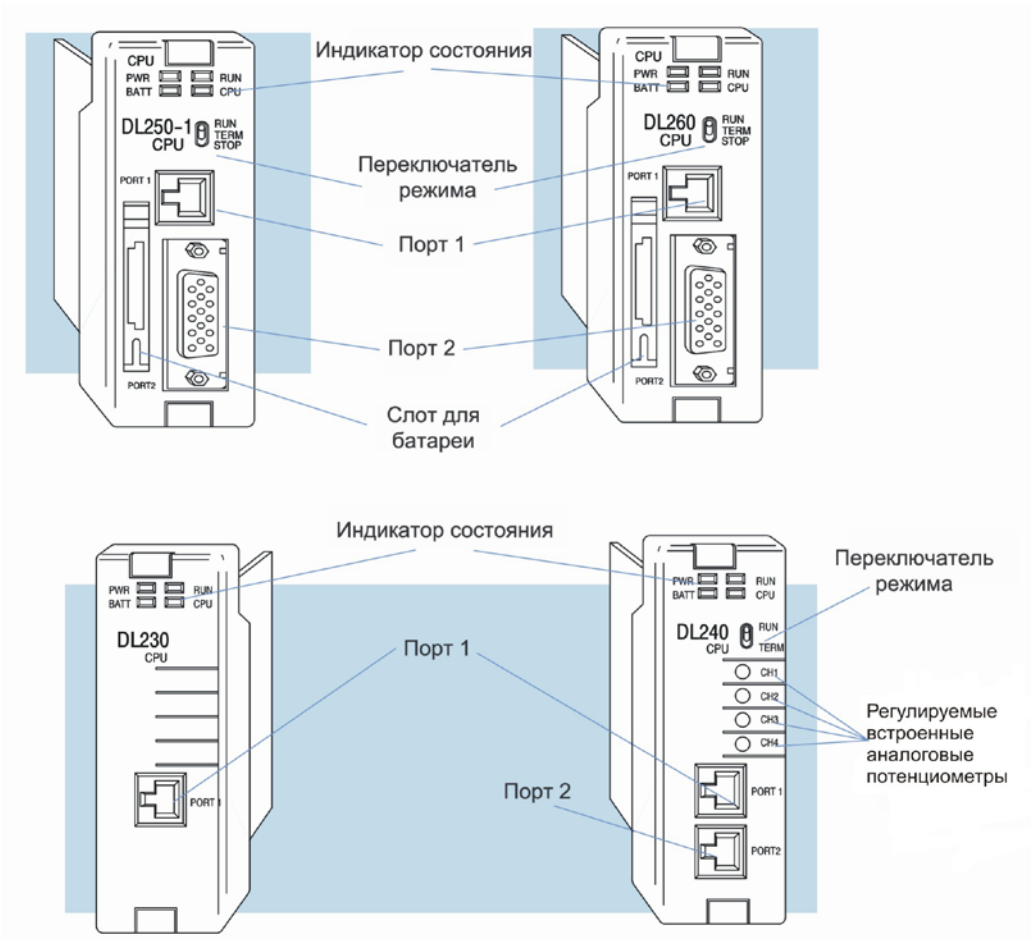
Код ошибки	Описание
E4**	Нет программ в процессоре
E401	Отсутствие оператора END
E402	Отсутствие оператора LBL
E403	Отсутствие оператора RET
E404	Отсутствие оператора FOR
E405	Отсутствие оператора NEXT
E406	Отсутствие оператора IRT
E412	SBR/LBL > 64
E413	FOR/NEXT> 64
E421	Двойная ссылка на стадию
E422	Двойная ссылка на SBR/LBL
E423	Вложенные циклы
E431	Неправильный адрес ISG/SG
E432	Неправильный адрес перехода (GOTO)
E433	Неправильный адрес SBR
E434	Неправильный адрес RTC
E435	Неправильный адрес RT
E436	Неправильный адрес INT
E437	Неправильный адрес IRTC
E438	Неправильный адрес IRT
E440	Неправильный адрес данных
E441	ACON/NCON
E451	Неверное MLC/MLR
E452	Вход X использован как выход
E453	Отсутствие Таймера/Счетчика
E454	Неверный TMRA
E455	Неверный CNT
E456	Неверный SR

Код ошибки	Описание
E461	Переполнение стека
E462	Потеря значимости стека
E463	Логическая ошибка
E464	Цепь LL не закончена
E471	Двойная ссылка на обмотку
E472	Двойная ссылка на TMR
E473	Двойная ссылка на CNT
E480	Ошибка разрядности CV
E481	Нет соединения CV
E482	Превышен CV
E483	Ошибка размещения CVJMP
E484	Нет CV
E485	Нет CVJMP
E486	Ошибка размещения BCALL
E487	Не определен блок
E488	Ошибка размещения блока
E489	Ошибка в идентификаторе блока CR
E490	Нет стадии блока
E4Э1	Ошибка размещения ISG
E492	Ошибка размещения BEND
E493	Ошибка I BEND
E494	Нет BEND

Индикаторы состояния процессорного модуля

На передней панели процессорных модулей контроллера DL205 расположены светодиодные индикаторы, с помощью которых вы можете диагностировать неисправности в системе. В приведенной ниже таблице приводится краткий перечень возможных неисправностей, соответствующих каждому состоянию индикатора. В последующей таблице дается подробный анализ каждой неисправности, отражаемой индикатором.

Состояние индикатора	Возможная неисправность
PWR (выключен)	1. Неправильное напряжение в системе 2. Неисправен источник питания / процессорный модуль 3. Закорочен источник питания другого компонента (Например, модуля ввода/вывода 4. Превышена мощность в используемом каркасе
RUN (не включается)	1. Ошибка в программе процессорного модуля 2. Переключатель находится в положении TERM 3. Переключатель находится в положении STOP (только для DL250-1 и DL260)
RUN (моргает)	ЦПУ в режиме обновления Фирменного ПО
CPU (включен)	1. Действие электрических помех 2. Неисправен процессорный модуль
BATT (включен)	1. Низкое напряжение батареи процессорного модуля 2. Отсутствие батареи процессора или она отсоединена



Индикатор PWR (Питание)

Существуют четыре основных причины, при которых светодиод состояния питания процессора (PWR) находится в состоянии ВЫКЛЮЧЕН:

1. Питание каркаса неправильно или неприложено.
2. Источник питания каркаса поврежден.
3. Отключен источник питания другого компонента(ов).
4. Превышена мощность каркаса.

Неправильное питание каркаса

Если напряжение, подаваемое на источник питания неправильное, то процессорный модуль и/или весь каркас не может нормально функционировать или вообще не сможет работать. Применяйте следующие указания при устранении неисправности.

ПРЕДУПРЕЖДЕНИЕ. *Для минимизации риска электрического удара всегда отключайте питание системы перед любой проверкой физических соединений*.

1. Сначала отключите питание системы и проверьте наличие обрыва в подводящей проводке.
2. Если вы используете отдельную плату для внешнего подключения, проверьте эти подключения, чтобы убедиться, что электропроводка соединена с соответствующими точками.
3. Если соединения правильны, включите питание системы и измерьте напряжение на клеммной колодке каркаса, чтобы убедиться, что оно удовлетворяет техническим требованиям. Если напряжение не удовлетворяет требованиям, отключите систему и устраните неисправность.
4. Если все соединения правильны и питание на входе соответствует требуемым техническим условиям, то источник питания каркаса необходимо отдать в ремонт.

Повреждение процессорного модуля

Лучший способ проверки Процессора это замена его другим исправным процессором, чтобы убедиться в наличии неисправности. Если были большие броски напряжения, то возможно Процессор или источник питания повреждены. Если Вы предполагаете, что броски напряжения являются причиной повреждения источника питания, то в будущем необходимо использовать устройство защиты питания от разрушающих пиков напряжения.

Устройство или модуль, вызывающие отключение источника питания

Существует вероятность того, что некоторый неисправный модуль или внешнее устройство, использующее системное питание 5В, может отключить источник питания. Это питание 5В может поступать с каркаса или с коммуникационных портов процессорного модуля.

Чтобы проверить устройство, вызвавшее эту неисправность:

1. Отключите питание процессорного модуля.
2. Отсоедините все внешние устройства (то есть соедирительные кабели) от процессорного модуля.
3. Снова подайте питание в систему.

Если источник питания работает нормально, то у вас короткое замыкание либо в устройстве, либо в кабеле. Если источник питания работает ненормально, то определение модуля, вызвавшего эту неисправность, проводится в соответствии с указанной ниже процедурой:

Если светодиод индикатора PWR не включается, то неисправность может быть в одном из модулей. Для того чтобы найти этот модуль, отключите питание системы и удаляйте по очереди по одному модулю, пока индикатор PWR не включится. Эту процедуру выполняйте следующим образом:

- Отключите питание каркаса.
- Выньте модуль из каркаса.
- Снова включите питание каркаса

Причиной такой неисправности может быть согнутый контакт разъема каркаса. Проверка разъема не составляет проблемы.

Превышение мощности источника питания каркаса

Если устройство работало нормально в течение длительного времени до момента выключения этого индикатора, то с большой вероятностью превышение мощности не является причиной неисправности. Проблемы превышения мощности обычно возникают при наладке системы, когда только ПЛК начинает работать, и входы/выходы требуют большего тока, чем может дать источник питания каркаса.

ПРЕДУПРЕЖДЕНИЕ: *Если мощность превышена, то ПЛК может произвести сброс и перезагрузку системы. Если у вас есть какие-либо сомнения в возможности превышения мощности, то проверьте нагрузку в этот момент. Превышение мощности может привести к непредсказуемым последствиям, к повреждению оборудования и нанесению вреда персоналу. Проверьте потребляемую модулями мощность в выбранном вами каркасе. Соответствующие таблицы вы можете найти в главе 4, в разделе «Каркасы и конфигурация ввода/вывода».*

Индикатор RUN (Работа)

Если процессор не переводится в Рабочий режим (индикатор RUN выключен), то обычно есть критическая ошибка в прикладной программе, если только сам процессор исправен. Однако, при неисправности процессора индикатор ЦПУ должен гореть. (Вы можете использовать программирующее устройство для определения причины ошибки).

Если вы применяете DL240, DL250-1 или DL260 и хотите изменить режим с помощью программирующего устройства, то убедитесь, что переключатель режимов находится в положении TERM.

Оба программирующих устройства, ручной программатор и *Direct*SOFT, возвращают сообщение с описанием возникшей неисправности. Может также существовать AUX функция, зависящая от вида ошибки, которая поможет вам диагностировать неисправность. Наиболее распространенной ошибкой в программах является «Отсутствие оператора END». Все прикладные программы требуют оператор END для своего корректного завершения. Полный перечень кодов ошибок можно найти в приложении B.

Индикатор CPU

Если индикатор CPU включен, то в процессорном модуле возникла неисправимая ошибка. В общем случае это не программная ошибка, а фактический отказ технических средств. Вы можете повторно запустить систему, чтобы устранить ошибку. Если ошибка устранена, то вам необходимо проконтролировать систему и определить, что явилось причиной неисправности. Иногда такие неисправности вызываются высокочастотными электрическими помехами, вносимыми в процессорный модуль из внешнего источника. Проверьте заземление вашей системы и установите фильтры электрических помех, если сомневаетесь в заземлении. Если при повторном запуске системы ошибка не исчезает, либо, если такая неисправность возникает вновь, то вам необходимо заменить процессор.

Индикатор BATT

Если индикатор BATT включается, то либо батарея процессорного модуля отсоединена, либо она нуждаются в замене. Когда питание подано в систему, напряжение на батарее отслеживается непрерывно.

Неисправности в коммуникациях

Если вы не можете установить связь с процессором, проверьте следующие возможные причины:

- Отсоединен кабель.
- В кабеле обрыв провода или он неправильно подсоединен.
- Кабель плохо заделан или заземлен.
- Подсоединенное устройство не работает при установленной скорости передачи (9600 бод для верхнего порта. Используйте AUX 56 для выбора скорости передачи для нижнего порта DL240, DL250-1 и DL260).
- Устройство, подсоединенное к порту, посылает данные неправильно.
- Существует разность потенциалов заземления двух соединенных устройств.
- Электрические помехи вызывают периодические ошибки.
- В процессорном модуле неисправный коммуникационный порт, процессор должен быть заменен.

При появлении ошибки индикатор включается и остается включенным пока связь не будет успешно установлена.

Поиск неисправностей в модулях ввода/вывода

Возможные причины появления неисправности

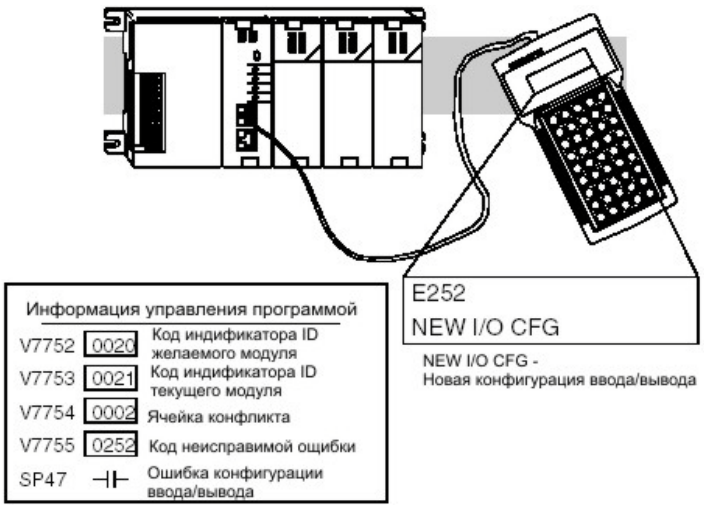
Если вы предполагаете наличие ошибки в работе подсистемы ввода/вывода, то необходимо проверить следующие возможные причины появления неисправности:

- Перегоревший предохранитель
- Плохой контакт модуля с каркасом.
- Отказ источника питания 24В постоянного тока
- Модуль вышел из строя
- При проверке конфигурации ввода/вывода обнаружено изменение в конфигурации.

Диагностика ввода/вывода

Если неизвестно, какой модуль неисправен, запустите функцию AUX 42 с ручного программатора или диагностику ввода/вывода в *Direct*SOFT. В обоих случаях вы получите номер каркаса, номер слота и неисправность в модуле. Как только неисправность будет устранена, индикатор вернется в исходное состояние.

Ошибка в подсистеме ввода/вывода не приводит к переключению процессорного модуля из Рабочего режима в Программный режим. Однако, она фиксируется в специальных реле (SP) процессора, которые дают возможность прочитать эту ошибку программой. Прикладная программа далее может выполнить требуемые действия, например, перевести процессор в Программный режим или инициировать корректное отключение системы. На рисунке ниже показан пример указателей отказов.



Несколько быстрых шагов

При поиске неисправности в модулях ввода/вывода серий DL имеется несколько обстоятельств, которые вам следует учитывать. Они помогут быстро найти и ликвидировать неисправности в подсистеме ввода/вывода.

- В выходных модулях нельзя автоматически обнаружить выходные точки с коротким замыканием или с обрывом. Если вы полагаете, что одна или несколько точек выходного модуля неисправны, то измерьте перепад напряжения между общим проводом и предполагаемой точкой. Напоминаем, что при использовании цифрового вольтметра необходимо учитывать ток утечки из выходного устройства, такого как тиристор или транзистор. Отключенная точка может дать такой же ток, как включенная точка при отключенной нагрузке.
- Индикаторы состояния точек ввода/вывода в модулях являются логическими индикаторами. Это значит, что светодиод, указывающий состояние «включено» или «выключено», отражает это состояние по отношению к процессору. Индикаторы состояния на выходном модуле могут нормально функционировать, в то время как реальное выходное устройство (например, тиристор или транзистор и др.) может быть неисправно. Что касается входных модулей, то, если светодиод индикатора включен, входные цепи должны работать правильно. При проверке правильного функционирования, убедитесь, что светодиод выключается, когда входной сигнал снимается.
- Ток утечки может стать проблемой при подсоединении полевых устройств к модулям ввода/вывода. Могут генерироваться ложные входные сигналы, когда ток утечки выходного устройства большой, достаточный для включения подсоединенного входного устройства. Для того чтобы это устранить, установите резистор параллельно входу или выходу цепи. Величина этого резистора будет зависеть от тока утечки и приложенного напряжения, но обычно достаточно применить резистор сопротивлением 10-20 кОм. Проверьте, что мощность, рассеиваемая этим резистором, достаточна для вашего приложения.
- Самый легкий способ определения неисправного модуля состоит в замене его, если вы имеете резерв. Однако, если другое устройство вызвало неисправность в этом модуле, то оно может привести к такой же неисправности в замененном модуле. Поэтому в качестве меры предосторожности проверьте устройства и источники питания, подключенные к неисправному модулю, перед тем, как заменить его на резервный модуль.

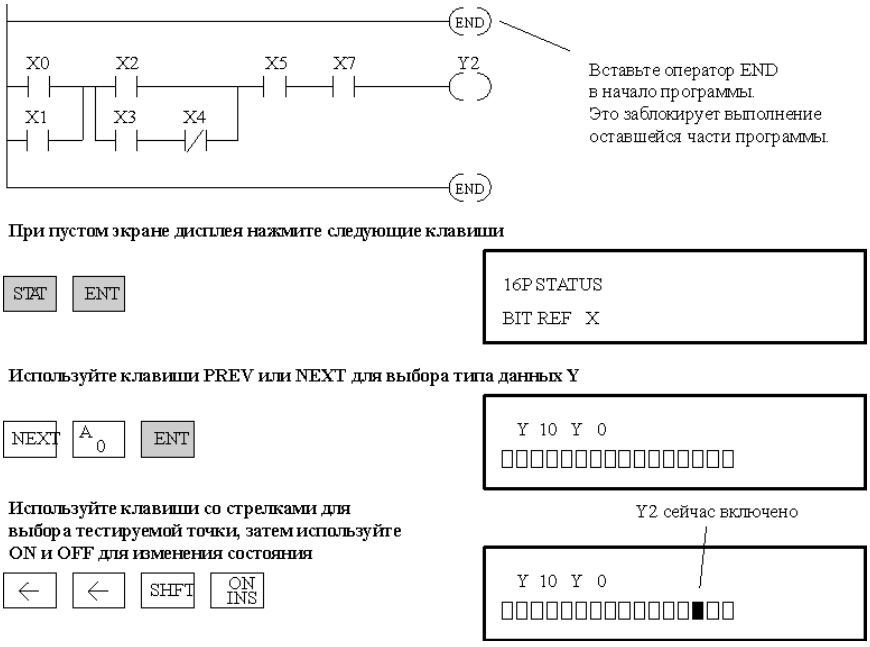
Тестирование выходных точек

В процессорных модулях серии DL205 выходные точки могут находиться в состоянии «включено» или «выключено». В DL240 и DL250-1 вы можете использовать функцию AUX 59, форсирование бита, чтобы искусственно воздействовать на точку даже при работе программы. Однако такой метод тестирования выходных точек не рекомендуется. Если вы хотите провести независимую от прикладной программы проверку ввода/вывода для DL230, DL240, DL250-1 или DL260, используйте следующую процедуру:

Шаг	Действие
1	Используйте ручной программатор или *Direct*SOFT для оперативной связи с ПЛК.
2	Измените режим на Программный.
3	Перейдите в адрес 0.
4	Поставьте в адресе 0 оператор END (При этом программа будет выполняться только до адреса 0 и не сможет включать или выключать точки Ввода/Вывода)
5	Измените режим на Рабочий.
6	Используйте программирующее устройство для включения или выключения точек, которые вы хотите протестировать.
7	После завершения тестирования точек ввода/вывода удалите оператор END в адресе 0.

ПРЕДУПРЕЖДЕНИЕ: *В зависимости от вашего приложения искусственное воздействие на точки ввода/вывода может привести к непредсказуемой работе технических средств, что, в свою очередь, может вызвать риск нанесения вреда персоналу или повреждения оборудования. Убедитесь в том, что перед тестированием точек ввода/вывода приняты все меры предосторожности, обеспечивающие безопасность.*

Работа с клавишами Ручного Программатора при тестировании выходной точки



Поиск и устранение помех

Проблемы электрических помех

Помехи являются одной из самых трудных проблем при диагностике. Электрические помехи могут вводится в систему разными способами, их можно разделить на две категории, введенные и наведенные. Иногда трудно определить, как помехи попадают в систему, однако действия по их устранению сходны.

- Введенные помехи — это электрические помехи, которые поступают в систему через подсоединенные провода, подсоединения платы и др. Они также могут вводиться через модули ввода/вывода, подключение источника питания, заземление коммуникаций или заземление блоков.
- Наведенные помехи — это электрические помехи, поступающие в систему без непосредственного электрического контакта, подобно радиоволнам.

Уменьшение электрических помех

Электрические помехи нельзя полностью исключить, поэтому они должны быть уменьшены до уровня, который не отражается на работе системы.

- Большинство проблем с помехами возникает из-за неправильного заземления системы. Хорошее заземление может оказаться единственным эффективным способом устранения помех. Если заземление недостаточно, установите заземляющий стержень по возможности ближе к системе. Обеспечьте, чтобы все провода заземления подсоединялись к одной точке «земля», чтобы не было «гирляндных» цепей — из одного устройства в другое. Заземлите металлическое ограждение системы. Свободный провод — это большая антенна для приема помех в систему, поэтому вы должны затянуть все соединения в вашей системе. Свободный провод заземления более чувствительный к помехам, чем другие провода вашей системы. По вопросам заземления вашей системы обращайтесь к главе 2 «Установка, монтаж и спецификации».
- Электрические помехи могут попасть в систему через источник питания процессорного модуля или подсистемы ввода/вывода. Установка развязывающего трансформатора для всех источников переменного тока может решить эту проблему. Источники постоянного тока должны быть высокого качества и хорошо заземлен. Импульсные источники питания постоянного тока обычно генерируют больше помех, чем линейные (трансформаторные) источники.
- Отделите входные провода от выходных. Никогда не прокладывайте провода подсистемы ввода/вывода близко к проводам высокого напряжения.

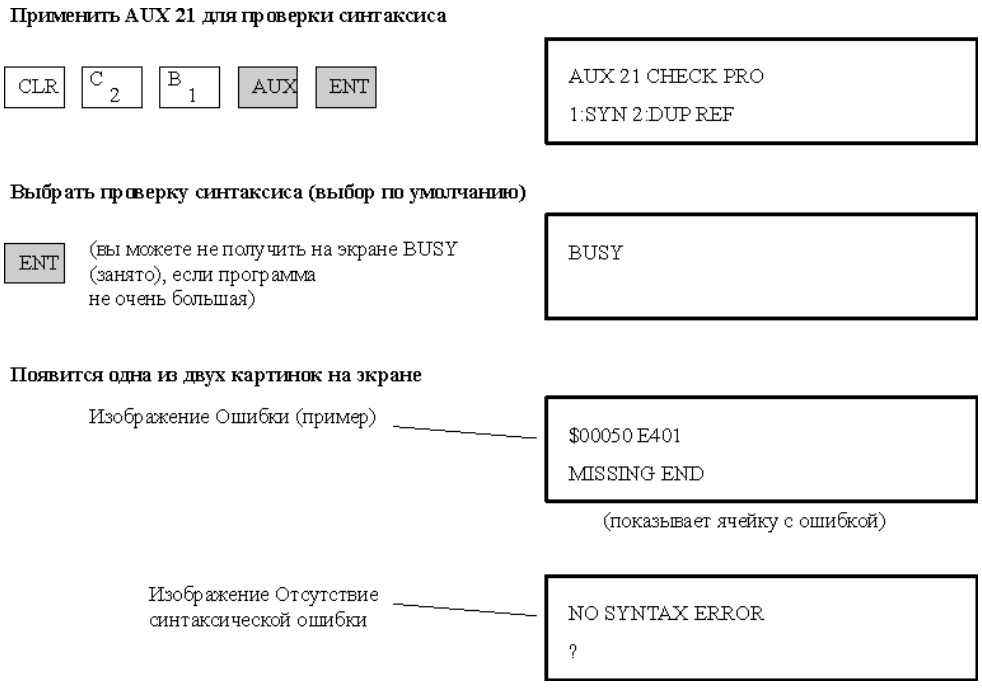
Запуск машин и поиск ошибок в программах

Процессорные модули контроллеров DL205 предоставляют несколько возможностей для отладки вашей программы до и после запуска производственных машин. В данном разделе рассматриваются следующие полезные для вас темы.

- Проверка синтаксиса программ
- Проверка дублированных ссылок
- Режимы тестирования
- Специальные команды
- Редактирование в рабочем режиме (RUN TIME EDIT)
- Форсирование точек ввода/вывода

Проверка синтаксиса

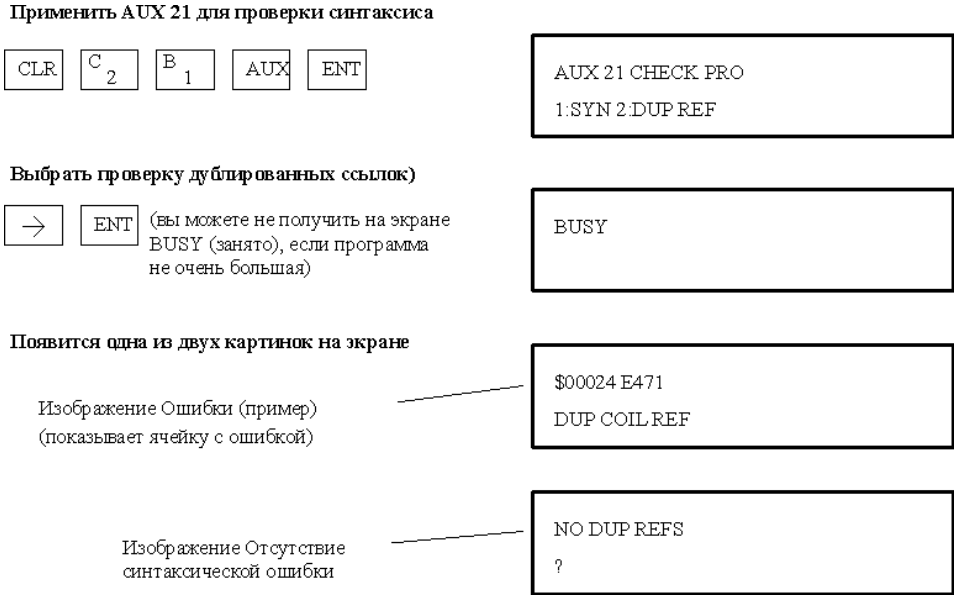
Хотя Ручной Программатор и *Direct*SOFT обеспечивают проверку ошибок при вводе программы, вы можете отдельно проверять измененные программы. Оба этих устройства для программирования предусматривают возможность проверки синтаксиса программ. Например, вы можете использовать вспомогательную функцию AUX 21, CHECK PROGRAMM, для проверки синтаксиса программы с Ручного Программатора, либо вы можете воспользоваться опцией меню диагностика ПЛК в *Direct*SOFT. Эта проверка позволяет найти разнообразные программные ошибки. На следующем примере показывается, как применять проверку синтаксиса с помощью ручного программатора.



Обратитесь к разделу кодов ошибок с полным списком кодов программных ошибок. Если вы получили ошибку, нажмите CLR, и на Ручном Программаторе отобразится команда с ошибкой. Исправьте ошибку и продолжайте проверку синтаксиса, пока не появится сообщение NO SYNTAX ERROR (синтаксических ошибок нет).

Проверка дублированных ссылок

Вы можете проверить также многократное использование одной и той же выходной обмотки. Оба устройства для программирования предоставляют возможность проверки этого условия. Например, вы можете использовать вспомогательную функцию AUX 21, CHECK PROGRAMM, для проверки дублированных ссылок с Ручного Программатора, либо вы можете воспользоваться опцией меню диагностика ПЛК в *Direct*SOFT. На следующем примере показывается, как выполнить проверку дублированных ссылок с Ручного Программатора.



Если вы получили ошибку, нажмите CLR, и на Ручном Программаторе отобразится команда с ошибкой. Исправьте ошибку и продолжайте проверку дублированных ссылок, пока таких ссылок не будет найдено.

ПРИМЕЧАНИЕ: Вы можете использовать одну и ту же обмотку в нескольких местах, особенно в программах на базе стадийных команд и/или команд OROUT. Проверка дублированных ссылок обнаружит эти выходы, хотя их применение допустимо.

Режимы TEST-PGM и TEST-RUN

В режиме тестирования процессор может начать работу в режиме TEST-PGM, перейти в режим TEST-RUN, отработать заданное число циклов сканирования и вернуться в режим TEST-PGM. Вы можете задать от 1 до 65525 циклов сканирования. Тестовый режим позволяет также сохранять выходные состояния при переключениях между режимами тестирования программ (TEST-PGM) и тестированием работы (TEST-RUN). Вы можете выбрать режим тестирования либо с помощью Ручного Программатора (используя клавишу MODE), либо из *Direct*SOFT с помощью опции меню режимы ПЛК.

Основное преимущество использования режима тестирования (TEST) состоит в сохранении выходов и других параметров, когда процессорный модуль переходит обратно в режим тестирования программ. Например, вы можете использовать функцию AUX 58 на ручном программаторе DL205, для конфигурирования конкретных выходов, Специальных реле и др. таким образом, чтобы они сохраняли свои выходные состояния. Также и процессорный модуль сохранит текущие значения таймера и счетчика при переключении в режим TEST-PGM.

ПРИМЕЧАНИЕ: Для задания числа циклов сканирования вы можете использовать только *Direct*SOFT. Эта функция не поддерживается Ручным Программатором. Однако, вы можете использовать Ручной Программатор для переключения между режимами тестирования программ (TEST-PGM) и тестированием работы (TEST-RUN).

Если на Ручном Программаторе вы первый раз выбираете режим тестирования, то фактический режим, в который вы входите, зависит от режима работы процессора в момент запроса. Если процессорный модуль находится в Рабочем режиме, то переход в режим TEST-RUN допустим. Если процессорный модуль — в Программном режиме, то допустим TEST-PGM. Однако, после выбора режима тестирования вы легко можете переключаться между TEST-RUN и TEST-PGM. *Direct*SOFT более гибок при выборе режимов с помощью различных опций меню. В следующем примере показывается, как вы можете пользоваться Ручным Программатором при выборе режимов тестирования.

Используйте клавишу MODE для выбора режимов тестирования (в примере предполагается рабочий (RUN) режим)

MODE NEXT ENT

MODE CHANGE
GO TO T-RUN MODE

Нажмите ENT для подтверждения режима TEST-RUN

ENT (Обратите внимание, что светодиод TEST ручного программатора DL205 показывает, что процессор находится в режиме тестирования (TEST))

MODE CHANGE
CPU T-RUN

Вы можете вернуться в режим RUN, войти в программный режим или в режим TEST-PGM, используя клавишу MODE

CLR MODE NEXT NEXT ENT

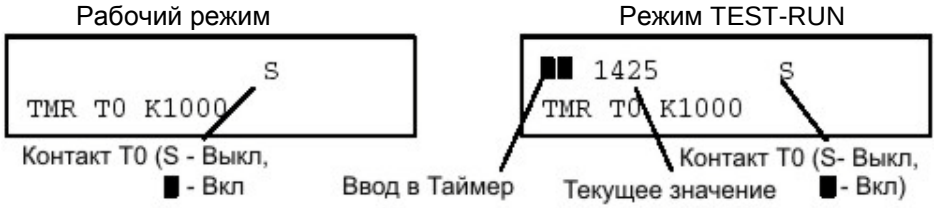
MODE CHANGE
GO TO T-PGM MODE

Нажмите ENT для подтверждения режима TEST-PGM

ENT (Обратите внимание, что светодиод TEST ручного программатора DL205 показывает, что процессор находится в режиме тестирования (TEST))

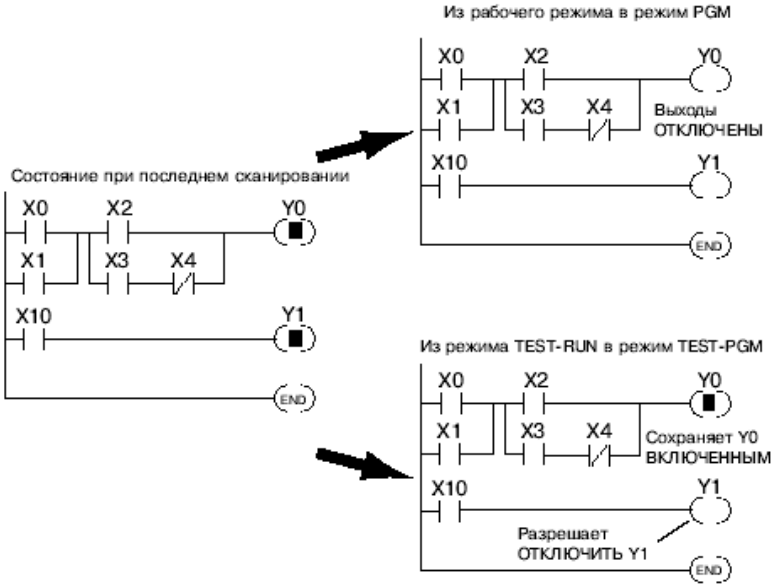
MODE CHANGE
CPU T-PGM

Отображения на экране при тестировании: на Ручном Программаторе в режиме тестирования вы можете получить на экране более подробное отображение данных. При некоторых командах отображение состояний в режиме TEST-RUN более детализированное, чем в Рабочем режиме. На следующей схеме показан пример отображения команды таймера в режиме TEST-RUN.



Сохранение выходных состояний: возможность сохранения выходных состояний весьма полезна, так как позволяет вам сохранить ключевые системные состояния точек ввода/вывода. В некоторых случаях вам может потребоваться изменить программу, но вы не хотите выполнять операции останова. В обычном рабочем режиме выходы отключаются при переходе в Программный режим. В режиме TEST-RUN при переходе в режим TEST-PGM вы можете выбрать для каждого конкретного выхода — либо отключение, либо сохранение его последнего состояния. Можно использовать функцию AUX 58 на Ручном Программаторе для выбора действий по каждому конкретному выходу. Эта функция доступна также через опции меню *Direct*SOFT.

На следующей схеме показано различие между Рабочим режимом и режимом TEST-RUN.



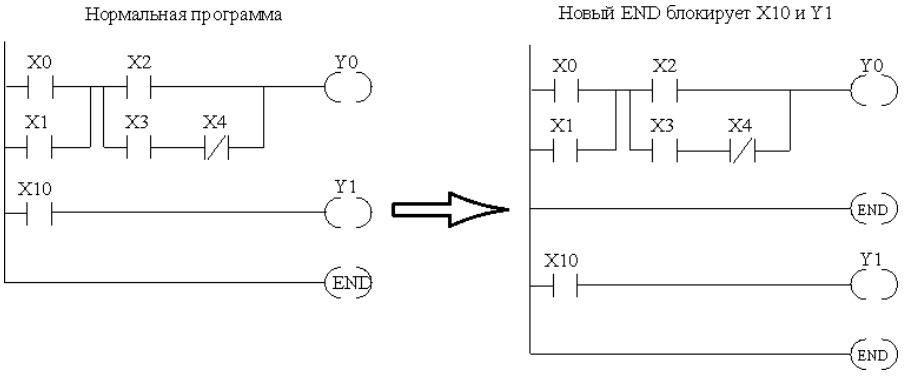
Перед тем как отдать предпочтение режиму тестирования вспомните, что процессорные модули контроллера DL205 также позволяют вам редактировать программы в Рабочем режиме. Основное отличие между режимами тестирования и редактированием в Рабочем режиме (RUN TIME EDIT) состоит в том, что вы не должны конфигурировать каждую конкретную точку ввода/вывода на сохранение ее состояния. Если вы используете редактирование в Рабочем режиме, то процессорный модуль автоматически сохраняет все текущие состояния выходов все время, пока обновляются программы.

Специальные команды

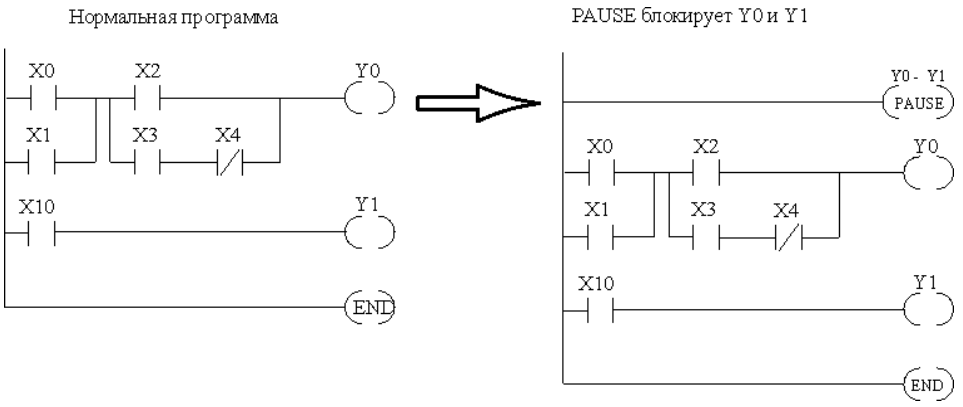
Существует несколько команд, которые могут помочь вам при отладке программы во время операций после запуска машин.

- END
- PAUSE
- STOP

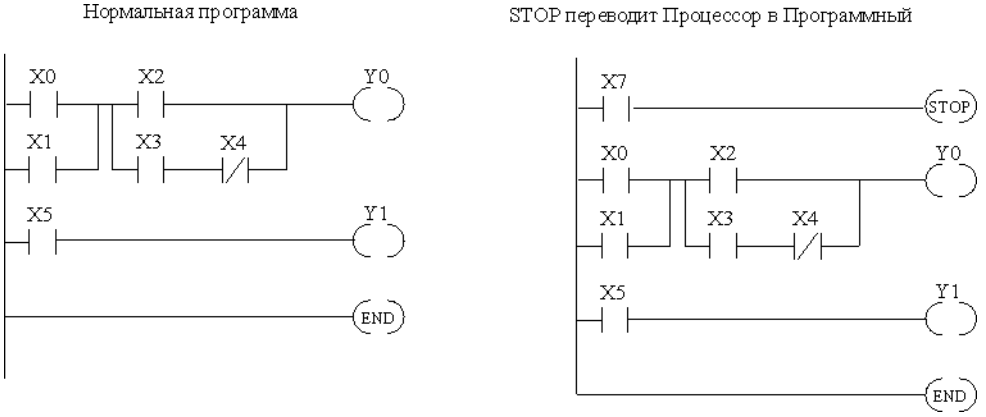
Команда END: если вам необходимо быстро заблокировать часть программы, вставьте оператор END перед той частью, которую нужно заблокировать. Когда процессорный модуль встретит оператор END, он будет считать, что это конец программы. На следующей схеме приведен пример.



Команда PAUSE: эта команда позволяет быстро установить режим, в котором входы (или другие логические переменные) функционируют при блокировании выбранных выходов. При этом регистр отображения выходов обновляется, но выходные состояния не записываются в модули. Например, вы можете сделать эту блокировку условной, добавив входной контакт или Специальное реле, чтобы управлять включением этой команды с помощью переключателя, либо с помощью устройства для программирования. Или вы можете добавить эту команду без всяких условий, при этом выбранные выходы будут блокированы все время.



Команда STOP: иногда после запуска машин вам необходимо быстро отключить все выходы и вернуться в Программный режим. Кроме использования режимов тестирования и функции AUX 58 (для установки каждой конкретной точки) вы можете использовать также команду STOP. При выполнении этой команды процессорный модуль автоматически выйдет из Рабочего режима и перейдет в Программный режим. Напоминаем, что все выходы в Программном режиме отключаются. На следующей схеме приведен пример условия, при котором процессорный модуль возвращается в Программный режим.



В приведенном примере вы можете включить X7, который выполнит команду STOP, процессорный модуль войдет в Программный режим, а все выходы будут отключены.

Редактирование в Рабочем режиме

Процессорные модули контроллера DL205 дают вам возможность делать изменения в прикладной программе в Рабочем режиме. Однако, такое редактирование не является «безударным». Цикл сканирования процессорным модулем мгновенно прерывается (а выходы сохраняются в текущем состоянии), пока изменение в программе не будет завершено. Это означает, что, если выход отключен, то он останется отключенным до тех пор, пока не закончится изменение программы. Если выход включен, он останется включенным.

ПРЕДУПРЕЖДЕНИЕ: *Только квалифицированные лица, знающие в полном объеме все аспекты приложения, могут вносить изменения в программу. Изменения, внесенные в Рабочем режиме, начинают действовать немедленно. Тщательно проанализируйте последствия любых изменений, чтобы минимизировать риск нанесения вреда персоналу или повреждения оборудования.*

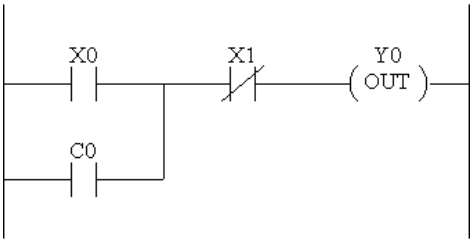
При изменении программ в процессе редактирования в Рабочем режиме следует учитывать следующие важные моменты:
Если новая команда содержит синтаксическую ошибку, то процессор не перейдет в Рабочий режим.
Если вы удалили ссылку на выходную обмотку, а в это время выход был включен, то выход останется включенным, пока вы не отключите его с устройства для программирования.
Изменения входных точек не допускаются при редактировании в Рабочем режиме. Поэтому, когда вы используете высокоскоростную операцию и придет критический входной сигнал, Процессор может не увидеть это изменение.

Не все команды можно редактировать в сеансе редактирования в Рабочем режиме. В следующем списке приведены команды, которые можно редактировать.

Обозначение	Описание
TMR	Таймер
TMRF	Быстрый таймер
TMRA	Аккумулирующий таймер
TMRAF	Аккумулирующий быстрый
CNT	Счетчик
UDC	Суммирующий/вычитающий
SGCNT	Счетчик стадий
STR, STRN	Сохранить, Сохранить НЕ
AND, ANDN	И, И-НЕ
OR, ORN	ИЛИ, ИЛИ-НЕ
STRE, STRNE	Сохранить, если равно; Сохранить, если НЕ-равно
ANDE, ANDNE	И, если равно; И. если НЕ-
ORE, ORNE	ИЛИ, если равно; ИЛИ. если
STR, STRN	Сохранить, если больше или равно; Сохранить, если
AND, ANDN	И, если больше или равно; И, если меньше

Обозначение	Описание
OR, ORN	ИЛИ, если больше или равно; ИЛИ, если меньше
LD	Загрузка данных (константы)
LDD	Загрузка двойная данных
ADDD	Сложение двойное (констант)
SUBD	Вычитание двойное (кон-
MUL	Умножение (констант)
DIV	Деление (констант)
CMPD	Сравнить аккумулятор (кон-
ANDD	И аккумулятора (констант)
ORD	ИЛИ аккумулятора (констант)
XORD	Исключающее ИЛИ аккумулятора (констант)
LDF	Загрузка дискретных точек в
OUTF	Выгрузка аккумулятора в дискретные точки
SHFR	Сдвиг аккумулятора вправо
SHFL	Сдвиг аккумулятора влево
NCON	Числовая константа

Используйте показанную на рисунке программную логику для описания того, как данный процесс работает. В этом примере измените XO на C10. В данном примере предполагается, что процессорный модуль уже находится в Рабочем режиме



Используйте клавишу MODE для выбора RUN TIME EDITS (Редактирования в Рабочем режиме)

MODE | NEXT | NEXT | ENT

```
*MODE CHANGE*
RUN TIME EDIT?
```

Нажмите ENT для подтверждения RUN TIME EDIT

ENT (светодиод RUN Ручного Программатора D2-HPP начинает мигать, указывая, что режим RUN TIME EDIT разрешен)

```
*MODE CHANGE*
RUNTIME EDITS
```

Найдите команду, которую вы хотите изменить (X0)

SHFT | X SET | A 0 | SHFT | FD REF FIND

```
$00000 STR X0
```

Нажмите клавишу со стрелкой для перехода к X. Затем введите новый контакт (C10)

→ | → | SHFT | C 2 | B 1 | A 0 | ENT

```
RUNTIME EDIT?
STR C10
```

Нажмите ENT для подтверждения изменения

ENT (После нажатия отображается следующий адрес)

```
OR C0
```

Форсирование точек ввода/вывода

Существует много случаев, особенно после запуска машин и при поиске неисправностей, когда вам необходимо принудительно включить или выключить точку ввода/вывода. Перед тем как начать использовать устройство программирования для воздействия на какой-либо тип данных, важно понять, как процессорный модуль контроллера DL205 обрабатывает запросы при таком форсировании.

ПРЕДУПРЕЖДЕНИЕ: *Только квалифицированные лица, знающие в полном объеме все аспекты приложения, могут изменять программу. Тщательно проанализируйте последствия любых изменений, чтобы минимизировать риск нанесения вреда персоналу или повреждения оборудования.*

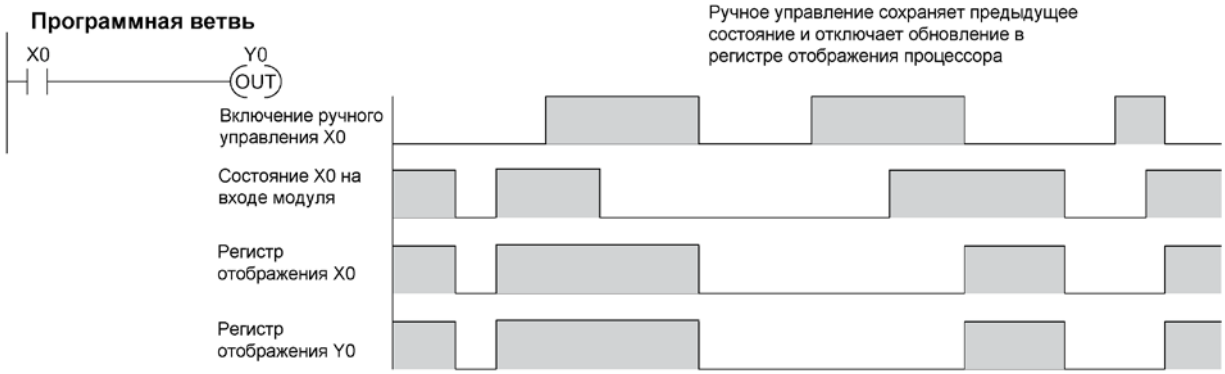
Существует два типа форсирования, доступных для процессорных модулей контроллера DL205. (В главе 3 приводится более подробное описание того, как процессорный модуль обрабатывает каждый тип запроса на форсирование).

- **Нормальное форсирование - Forsing:** этот тип форсирования может временно изменять состояние бита дискретного сигнала. Например, вы желаете включить вход, хотя реально он отключен. Это даст вам возможность изменить состояние точки, которое хранится в регистре отображения. Это значение будет действовать до тех пор, пока ячейка регистра отображения не будет обновлена при следующем цикле сканирования. Это полезно, главным образом, при тестировании, когда вам необходимо включить какой-то бит, чтобы запустить другое событие.

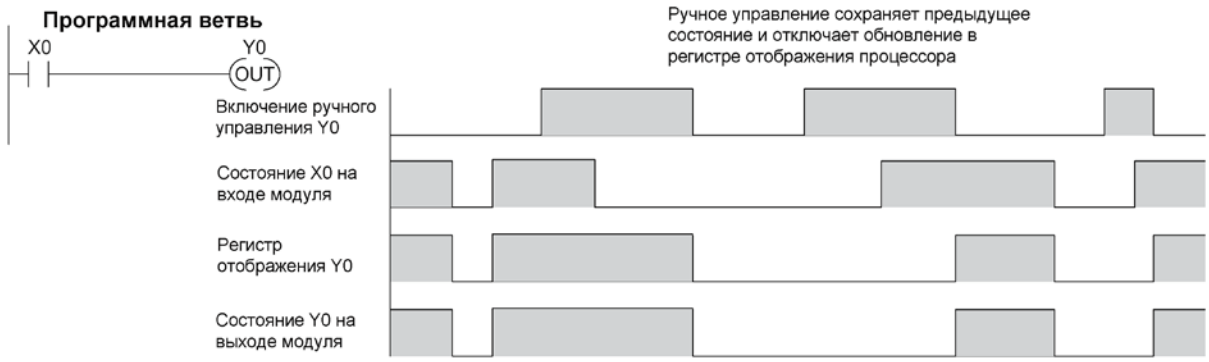
- **Подмена бита- Bit Override (DL240, DL250-1 и DL260):** подмена бита может быть разрешена для конкретных битов с помощью функции AUX 59 с Ручного Программатора или с помощью опции меню *Direct*SOFT. Подмена бита можно применять для типов данных X, Y, C, T, CT и S. Подмена бита отключает любые изменения дискретной точке от процессора. Например, если Вы разрешаете подмену бита X1, и X1 выключен в это время, то процессор не сможет изменить состояние X1. Это означает, что, даже если X1 включается, то процессор не будет подтверждать изменение. Поэтому, если Вы использовали X1 в программе, то этот бит будет всегда «выключен» в этом случае. Если X1 включен, в момент разрешения подмены бита, то X1 будет всегда «включен».

Использование функции подмены бита принесет вам некоторую пользу. Нормальное форсирование не отключается при включении режима подмены бита. Например, если Вы разрешили подмену бита для Y0, и он выключен в это время, то процессор не будет изменять состояние Y0. Однако, Вы можете все еще использовать устройство программирования, чтобы изменить его состояние. Если Вы используете устройство программирования, чтобы включить Y0, то этот бит останется включенным, и процессор не будет изменять состояние Y0. Если Вы затем форсированием выключите Y0, то процессор обработает Y0 как выключенный. Процессор никогда не будет обновлять точку с результатами вычислений прикладной программы или от обновления ввода/вывода, пока не отключена подмена этого битом.

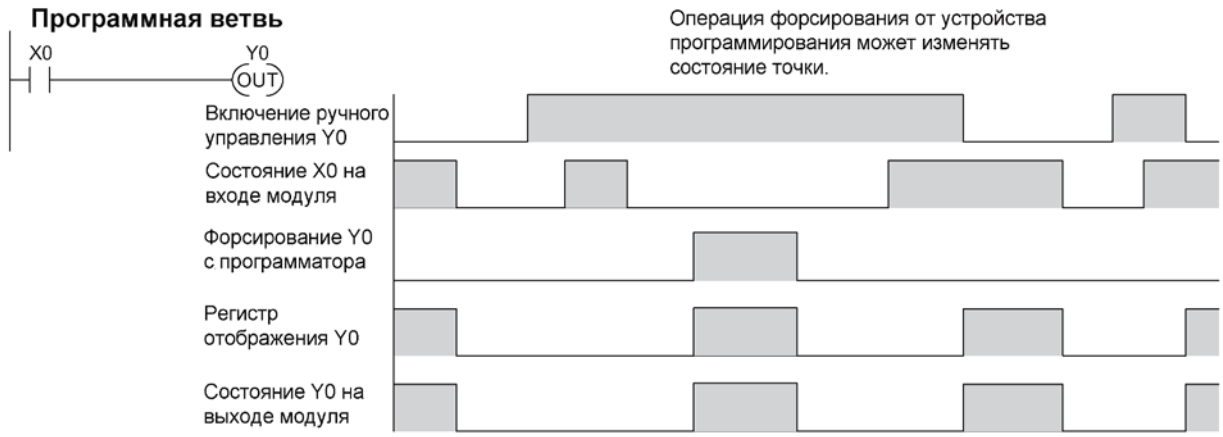
На следующих схемах показано, как работает подмена бита для входных и выходных точек. Пример составлен для простой цепочки, но его принципы справедливы для любого типа битовой памяти.



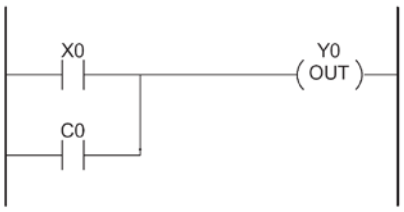
Следующая схема показывает, как работает подмена бита для точки вывода. Заметьте, что ручное управление битом поддерживает выход в текущем состоянии. Если выход включен, когда включается подмена бита, то выход остается включенным. Если он выключен, то выход остается выключенным.



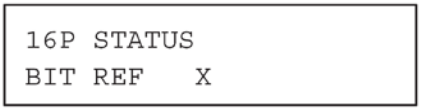
Следующая схема показывает, как вы можете использовать устройство программирования в комбинации с подменой бита, чтобы изменить состояние бита. Помните, подмена бита только отключает изменения от процессора. Вы можете использовать устройство программирования, чтобы форсированием изменить состояние бита. Плюс, так как подмена бита сохраняет текущее состояние, это позволяет выполнить истинное форсирование. Показанный пример использует точку выхода, но Вы можете также использовать другие битовые типы данных.



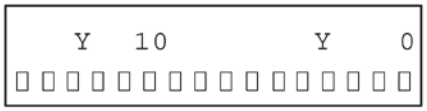
На следующих схемах показан пример того, как вы можете использовать Ручной Программатор для воздействия на точки ввода/вывода. Помните, если Вы используете операцию подмены бита, то процессор сохранит форсированное значение, пока вы не отключите подмену бита или не удалите форсирование. Регистр отображения не будет обновляться по состояниям, получаемым из входных модулей. Решения, полученные в прикладной программе, также не будут использоваться для обновления регистра отображения выходов. В примере предполагается, что вы уже перевели процессорный модуль в Рабочий режим.



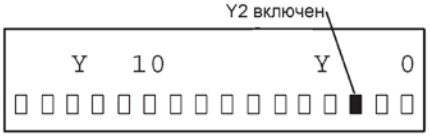
Очистите дисплей и введите следующую последовательность клавиш.



Используйте клавиши PREV или NEXT для выбора типа данных Y. (Как только увидите Y, нажмите 0 для перехода к выбору бита Y0.)

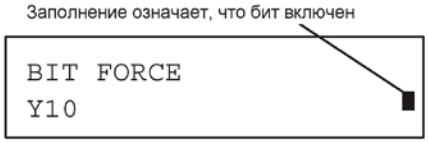


Используйте клавиши со стрелками для выбора точки, затем клавиши ON и OFF для изменения состояния

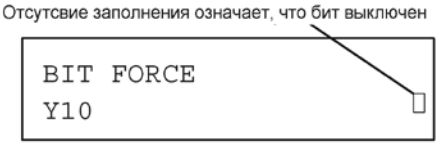


Нормальное форсирование с прямым доступом

Очистите дисплей и введите следующую последовательность клавиш для форсирования включения Y10. Заполнение, указывает, что бит включен.



Очистите дисплей и введите следующую последовательность клавиш для форсирования выключения Y10. Отсутствие заполнения, указывает, что бит выключен.



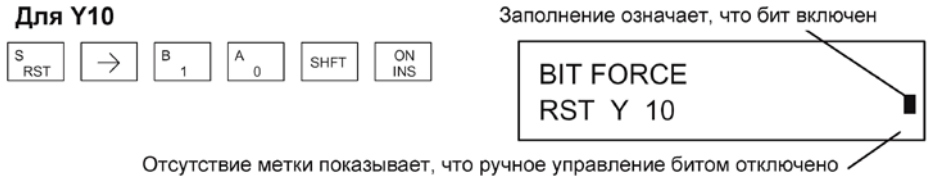
Форсирование бита с подменой

Очистите дисплей и введите следующую последовательность клавиш для форсирования состояния включения Y10.

ПРИМЕЧАНИЕ: Обратите внимание, в этой точке вы можете использовать клавиши PREV и NEXT для перехода к смежным ячейкам памяти, а также использовать клавиши SHFT ON для включения режима подмены бита.

Очистите дисплей и введите следующую последовательность клавиш для выключения режима подмены Y10. Заполнение, указывает, что бит включен.

Для Y10

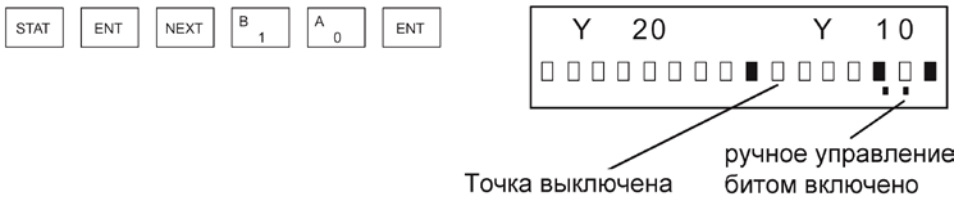


Как и в предыдущем примере, в этой точке вы можете использовать клавиши PREV и NEXT для перехода к смежным ячейкам памяти, а также использовать клавиши SHFT ON для выключения режима подмены бита.

Индикаторы подмены бита

Индикаторы подмены бита (Bit Override) могут отображаться на дисплее состояния Ручного Программатора. Ниже дана последовательность клавиш необходимых для того, чтобы вызвать отображение состояния для выходов Y10 - Y20.

Очистите дисплей и введите следующую последовательность клавиш для отображения состояния выходов Y10 – Y20.



Приложение A. ***Вспомогательные функции***

В этом приложении…

Введение

Что такое вспомогательные функции?

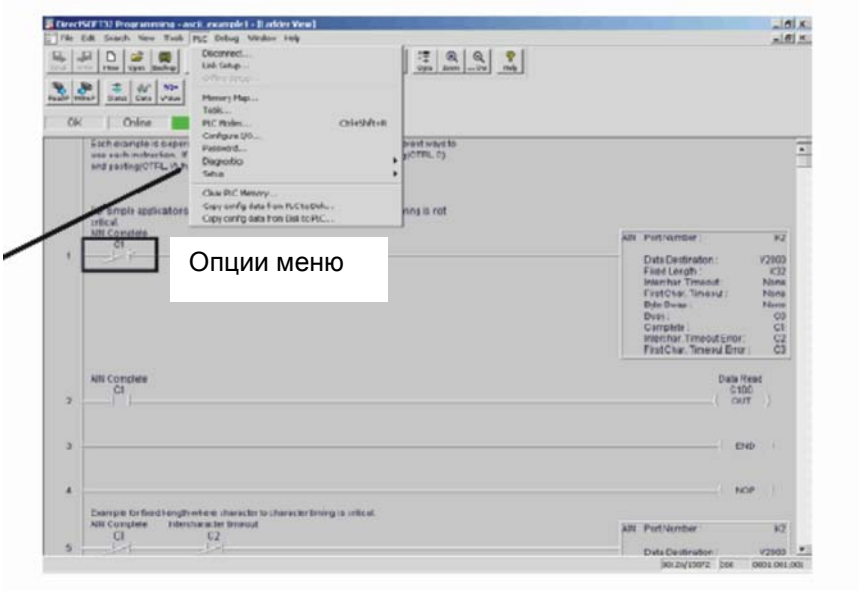
Многие задачи настройки процессорного модуля предполагают использование вспомогательных (AUX) функций. AUX функции выполняют много различных операций, включая очистку программной памяти, отображение времени сканирования, копирование программ в ЭППЗУ HPP и в процессор и др. Они разделены на категории, которые предназначены для различных системных параметров. Вы можете получить доступ к AUX функциям из *Direct*SOFT или с помощью Ручного Программатора, руководства по этим продуктам включают пошаговые процедуры для получения доступа к AUX функциям. Некоторые из этих AUX функций разработаны специально для настройки Ручного Программатора, поэтому они не нужны (или не доступны) для пакета программирования *Direct*SOFT. Несмотря на то, что в настоящем Приложении приводится достаточно много примеров использования AUX функций, вам следует дополнительно использовать документацию по выбранному вами устройству программирования. Следует отметить, что Ручной Программатор может иметь дополнительные AUX функции, которые не поддерживаются процессорными модулями контроллеров DL205.

AUX функции и их описание		230	240	250-1	260	HPP
AUX 2* - Операции RLL						
21	Проверка программы	V	V	V	V	
22	Изменение ссылки	V	V	V	V	
23	Очистка памяти в заданном диапазоне цепей	V	V	V	V	
24	Очистка памяти по всем цепям	V	V	V	V	
AUX 3* — Операции с V-памятью						
31	Очистка V-памяти	V	V	V	V	
AUX 4* — Конфигурация ввода/вывода						
41	Показать конфигурацию ввода/вывода	V	V	V	V	
42	Диагностика ввода/вывода	V	V	V	V	
44	Проверка конфигурации ввода/вывода при включении питания	V	V	V	V	
45	Выбрать конфигурацию	V	V	V	V	
46	Конфигурирование ввода/вывода	X	X	V	V	
AUX 5* — Конфигурация процессорного модуля						
51	Изменить имя программы	V	V	V	V	
52	Показать/изменить календарь	X	V	V	V	
53	Показать время сканирования	V	V	V	V	
54	Инициализировать электронный блокнот (Scratch Pad)	V	V	V	V	
55	Установить сторожевой таймер (Watch Dog Timer)	V	V	V	V	
56	Установить сетевой адрес процессора	X	V	V	V	
57	Установить области сохранения	V	V	V	V	
58	Операции тестирования	V	V	V	V	
59	Переопределить биты	X	V	V	V	
5B	Конфигурировать высокоскоростной ввод/вывод	V	V	V	V	
5C	Показать архив ошибок	X	V	V	V	
AUX 6* — Конфигурация Ручного Программатора						
61	Показать номер версии	V	V	V	V	-
62	Включить/отключить звуковой сигнал	X	X	X	X	V
65	Запустить самодиагностику	X	X	X	X	V
AUX 7* - Операции ЭППЗУ						
71	Копировать память процессора в ЭППЗУ Ручного Программатора	X	X	X	X	V
72	Записать ЭППЗУ Ручного Программатора в процессор	X	X	X	X	V
73	Сравнить процессор с ЭППЗУ Ручного Программатора	X	X	X	X	V
74	Проверить очистку памяти (ЭППЗУ Ручного Программатора)	X	X	X	X	V
75	Стереть ЭППЗУ Ручного Программатора	X	X	X	X	V
76	Показать тип ЭППЗУ (Процессор и Ручной Программатор)	X	X	X	X	V
AUX 8* — Операции с паролем						
81	Изменить пароль (Password)	V	V	V	V	-
82	Разблокировать процессор	V	V	V	V	-
83	Заблокировать процессор	V	V	V	V	-

где, V – поддерживается, X - не поддерживается, - - не применяется

Доступ к AUX функциям с помощью *Direct*SOFT

*Direct*SOFT предоставляет различные опции меню как при диалоговом, так и при автономном режиме программирования. Одни из AUX функций могут использоваться только в диалоговом режиме программирования, другие - только в автономном, а некоторые - и в том, и в другом. На рисунке ниже показан пример меню операции ПЛК в *Direct*SOFT.



Доступ к AUX функциям с Ручного Программатора

Вы также можете использовать Ручной Программатор для доступа к AUX функциям. Напоминаем, что некоторые AUX функции доступны только с Ручного Программатора. Иногда имя и описание AUX функции может выходить за рамки экрана. Для того чтобы увидеть полное описание такой AUX функции, нажимайте клавиши со стрелками для сдвига экрана влево или вправо. Кроме того, в зависимости от текущего отображения на экране вы можете нажимать клавишу CLR несколько раз.

CLR AUX

```
AUX FUNCTION SELECTION
AUX 2* RLL OPERATIONS
```

Используйте клавиши NXT и PREV для просмотра меню

NEXT

```
AUX FUNCTION SELECTION
AUX 3* V OPERATIONS
```

Нажмите клавишу ENT для вызова под-меню

ENT

```
AUX 3* V OPERATIONS
AUX 31 CLR V MEMORY
```

Вы можете также ввести точный номер AUX функции для непосредственного выхода в под-меню.

Введите непосредственно номер AUX функции

CLR D 3 B 1 AUX

```
AUX 3* V OPERATIONS
AUX 31 CLR V MEMORY
```

AUX 2* — Операции в RLL

AUX 21-24

Вы можете использовать четыре AUX функции для различных операций при управлении программой.

- AUX 21 — проверка программы
- AUX 22 — изменение ссылок
- AUX 23 — очистка программной памяти по диапазонам
- AUX 24 — очистка всей программной памяти

AUX 21-проверка программы

Как Ручной Программатор, так и *Direct*SOFT автоматически проверяют ошибки при вводе программы. Но могут возникнуть случаи, когда вам необходимо проверить программу, которая уже введена в процессорный модуль. Доступны два типа проверок:

- проверка синтаксиса
- проверка дублированных ссылок

При синтаксической проверке выявляются разнообразные ошибки в программах, например, отсутствие операторов END, неполные циклы FOT/NEXT и др. Если вы получили ошибку при проведении такой проверки, то обратитесь к Приложению B, в котором приведен полный список кодов ошибок в программах. После исправления ошибки продолжайте синтаксическую проверку, пока не появится сообщение «NO SYNTAX ERROR» (синтаксических ошибок нет).

При проверке дублированных ссылок контролируется, не используете ли вы ссылку на одну и ту же выходную обмотку несколько раз. Следует заметить, что данная AUX функция будет определять одни и те же выходы даже тогда, когда они применяются с командой OROUT, где дублирование допустимо.

Этой AUX функцией можно воспользоваться в *Direct*SOFT с помощью подменю «PLC/Diagnostics».

AUX 22 — изменение ссылок

Иногда вам необходимо изменить ссылку на адрес точки ввода/вывода или ссылку на управляющее реле. Функция AUX 22 позволит вам легко и быстро изменить все случаи использования конкретной команды (в рамках диапазона адресов). Например, вы можете заменить каждое значение X5 на X10.

AUX 23 — очистка программной памяти по диапазонам

Очень часто при решении новых прикладных задач вам необходимо добавить или удалить часть существующих программ. При помощи AUX 23 вы можете выбрать и удалить часть программы. *Direct*SOFT не имеет опции меню для этой AUX функции, но вы можете выбрать соответствующую часть программы и вырезать ее с помощью средств редактирования.

AUX 24 — очистка программной памяти

AUX 24 полностью убирает программу из V-памяти. Перед тем как вы вводите новую программу, вы всегда должны очистить программную память. Данная AUX функция доступна в *Direct*SOFT через подменю «Clear PLC».

AUX 3* — Операции с V-памятью

AUX 31 — очистка V-памяти

AUX 31 удаляет всю информацию из ячеек V-памяти, предназначенной для общего пользования. Данная AUX функция доступна в *Direct*SOFT через подменю «Clear PLC» меню ПЛК

AUX 4* — конфигурирование ввода/вывода

AUX 41-46

Вы можете использовать несколько AUX функций для настройки, просмотра или изменения конфигурации ввода/вывода.

- AUX 41 — вывести конфигурацию ввода/вывода
- AUX 42 — диагностика ввода/вывода
- AUX 43 — не используется в DL205
- AUX 44 — проверка конфигурации при включении питания
- AUX 45 — выбор конфигурации
- AUX 46 — конфигурирование ввода/вывода
-

AUX 41 — вывести конфигурацию ввода/вывода

Эта AUX функция позволяет вам вывести на экран текущую конфигурацию ввода/вывода. На Ручном Программаторе вы можете просмотреть каждый каркас и слот ввода/вывода. В конфигурации указывается тип модуля, установленного в каждом слоте. В *Direct*SOFT выдается та же информация, но гораздо удобнее для просмотра, так как вы можете просмотреть весь каркас на одном экране.

AUX 42 — диагностика ввода/вывода

Это одна из наиболее полезных AUX функций, имеющихся в системе DL205. Она покажет вам точное положение слота и каркаса любого модуля ввода/вывода, в котором возникла ошибка. Эта функция доступна также в *Direct*SOFT, вызывается из меню «PLC/Diagnostics».

AUX 44 — проверка конфигурации при включении питания

При выборе этой функции вы можете быстро обнаружить все изменения, которые могли произойти в тот период, когда питание системы было отключено. Например, если кто-то поместил выходной модуль в слот, в котором до выключения питания находился входной модуль, то при проверке конфигурации это изменение немедленно будет выявлено.

Когда система при включении питания обнаруживает изменение в конфигурации ввода/вывода, то генерируется код ошибки E252 NEW I/O CONFIGURATION (НОВАЯ КОНФИГУРАЦИЯ ВВОДА/ВЫВОДА). Вы можете воспользоваться функцией AUX 42 для определения точного положения каркаса и слота, в котором произошло это изменение.

ПРЕДУПРЕЖДЕНИЕ: В*ы всегда должны исправлять ошибки в конфигурации ввода/вывода перед переходом в Рабочий Режим. Неисправленные ошибки могут вызвать непредсказуемую работу аппаратуры, что может привести к риску нанесения вреда персоналу или к повреждению оборудования.*

Эту функцию можно вызывать также в *Direct*SOFT через подменю «PLC/Setup».

AUX 45 — выбор конфигурации

Хотя процессор и находит автоматически изменения в конфигурации, вы можете установить новую конфигурацию ввода/вывода. Например, вы можете специально изменить модуль ввода/вывода, чтобы поработать с новой программой. Вы можете использовать AUX 45 для выбора новой конфигурации или продолжать работать с прежней конфигурацию, сохраненной в памяти. Эту функцию можно вызывать также в *Direct*SOFT через подменю «PLC/Setup».

ПРЕДУПРЕЖДЕНИЕ: *Убедитесь в том, что выбранная конфигурация правильно работает с программами процессора. Всегда исправляйте ошибки в конфигурации ввода/вывода перед переводом процессорного модуля в Рабочий Режим. Неисправленные ошибки могут вызвать непредсказуемую работу аппаратуры, что может привести к риску нанесения вреда персоналу или к повреждению оборудования.*

AUX 46 — конфигурирование ввода/вывода

Вам, может быть, никогда не пригодится эта функция, тем не менее, с помощью функции AUX 46 вы для процессорных модулей DL-250-1 и DL-260 можете назначить адреса ввода/вывода для любого или всех слотов местного каркаса или каркаса расширения. Обычно гораздо проще выполнить операции по конфигурированию с помощью *Direct*SOFT. Этот пакет программирования предоставляет удобную панель для конфигурирования ввода/вывода, которая доступна через подменю «PLC/Configure I/O».

Эта функция полезна в том случае, когда у вас есть в контроллере стандартная конфигурация, но вам иногда нужно слегка ее изменить, чтобы она соответствовала неким специальным требованиям. Например, вам может потребоваться, чтобы два соседних входных модуля имели адреса, которые начинаются с X10 и X200 соответственно.

При автоматическом конфигурировании адреса назначаются на границе 8-ми точек. При ручной конфигурации предполагается, что все модули имеют по крайней мере 16 точек, поэтому можно только назначать адреса, которые кратны 20 (восьмеричное). Например, X30 и X50 – это не правильные начальные адреса для модуля. X20 и X40 – это правильные примеры начальных адресов для ручной конфигурации. Эти примеры вовсе не означают, что при ручной конфигурации вы можете использовать только 16- и 32-точечные модули. Можно использовать и 8-точечные модули, но назначено будет 16 адресов, и 8 из них не будут использованы.

ПРЕДУПРЕЖДЕНИЕ: *Если вы вручную конфигурируете слот ввода/вывода, то адресация ввода/вывода других модулей тоже изменится. Это связано с тем, что в контроллерах DL205 запрещено назначать одинаковые адреса ввода/вывода. Всегда исправляйте ошибки в конфигурации ввода/вывода перед переводом процессорного модуля в Рабочий Режим. Неисправленные ошибки могут вызвать непредсказуемую работу аппаратуры, что может привести к риску нанесения вреда персоналу или к повреждению оборудования.*

После того, как вы вручную задали адреса для слота ввода/вывода, система автоматически сохранит их и будет пользоваться ими даже после выключения и включения питания. Удалить ручную конфигурацию можно простым выполнением автоматической конфигурации.

AUX 5* — конфигурирование процессорного модуля

AUX 51-5C

Вы можете использовать несколько AUX функций для настройки, просмотра или изменения конфигурации процессорного модуля.

- AUX 51 — изменить имя программы
- AUX 52 — вывести/ изменить календарь
- AUX 53 — вывести время сканирования
- AUX 54 — инициализировать электронный блокнот
- AUX 55 — установить сторожевой таймер
- AUX 56 — сетевой адрес процессорного модуля
- AUX 57 — установить области в памяти для хранения данных
- AUX 58 — операции тестирования
- AUX 59 — форсирование бита
- AUX 5B — конфигурация интерфейса счетчика
- AUX 5C — вывести журнал ошибок/сообщений

AUX 51 — изменить имя программы

В контроллерах DL205 программа процессорного модуля или программа, хранимая в памяти ЭППЗУ Ручного Программатора, может иметь имя. (Следует напомнить, что вы не можете иметь много программ в ЭППЗУ.) Имя программы может быть длиной до восьми символов, в имени можно использовать любой из доступных символов (A - Z, 0 - 9). Функция AUX 51 дает вам возможность ввести имя программы. Вы можете выполнять эту операцию с помощью в *Direct*SOFT через подменю «PLC/Setup» (ПЛК/Настройка). После ввода имени программы вы можете удалить его только с помощью функции AUX 54 при сбросе системной памяти. Перед использованием функцией AUX 54 убедитесь, что вы поняли все возможные последствия этой функции.

AUX 52 — вывести/изменить календарь

Процессорные модули DL240, DL250-1 и DL260 имеют функцию часов и календаря. Если вы пользуетесь ими, то вы можете с помощью Ручного Программатора и функции AUX 52 установить время и дату. Используются следующие форматы:

- Дата — год, месяц, число, день недели (0 - 6, с воскресенья по субботу)
- Время — в формате 24 часа, часы, минуты, секунды.

Вы можете использовать эту AUX функцию, чтобы изменить любой элемент даты и времени. Однако, процессорный модуль не будет автоматически корректировать расхождение между датой и днем недели. Например, если вы изменили дату на 15-ое число месяца и 15 число является четвергом, то вам придется также изменить день недели (если только процессорный модуль уже не показывает дату как четверг).

Вы можете также выполнять эту операцию в *Direct*SOFT через подменю «PLC/Setup».

AUX 53 — вывести время сканирования

Функция AUX 53 отображает текущую, минимальную и максимальную длительность цикла сканирования. Минимальная и максимальная длительности берутся с момента последнего перехода из Программного режима в Рабочий. Вы можете также выполнять эту операцию в *Direct**SOFT*** через подменю «PLC/Diagnostics».

AUX 54 — инициализировать электронный блокнот

Процессорные модули контроллера DL205 поддерживают системные параметры в области памяти, которую часто называют «электронный блокнот». В некоторых случаях вы можете вносить изменения в настройку системы, которая хранится в системной памяти. Например, если вы определяете область управляющих реле (CR) как сохраняемую, то эти изменения запоминаются.

ПРЕДУПРЕЖДЕНИЕ: У вас нет необходимости применять эту функцию до тех пор, пока вы не делаете изменения, которые влияют на системную память. Обычно вам нужно инициализировать системную память только тогда, когда вы изменяете программы, которые требовали специальной настройки системы. Большей частью вы переносите изменения из программы в программу без какой-либо инициализации системной памяти.

Функция AUX 54 возвращает системную память к значениям по умолчанию. Вы можете также выполнять эту операцию в *Direct*SOFT через подменю «PLC/Setup».

AUX 55 — установить сторожевой таймер

У процессорных модулей контроллера DL205 есть «сторожевой» таймер, который применяется для отслеживания времени сканирования. Его значению по умолчанию, установленное в заводских условиях, равно 200 мс. Когда длительность цикла сканирования превосходит установленный временной предел, процессорный модуль автоматически переходитт из Рабочего режима в Программный. При этом на ручной программатор выдается следующее сообщение: E003 S/W TIMEOUT.

Используйте функцию AUX 55 для уменьшения или увеличения значения «сторожевого» таймера. Вы можете также выполнять эту операцию в *Direct*SOFT через подменю «PLC/Setup».

AUX 56 — сетевой адрес процессора

Поскольку процессорные модули DL240, DL250-1 и DL260 имеют дополнительный коммуникационный порт, вы можете использовать Ручной Программатор для установки сетевого адреса для порта и параметров связи порта. Параметрами настройки по умолчанию являются:

Адрес станции «1»
Шестнадцатеричный режим (HEX mode)
Контроль по нечетности (Odd parity)
Вы можете использовать этот порт для подключения Ручного Программатора, для *Direct*SOFT или как коммуникационный порт *Direct*NET. В Руководстве по *Direct*NET приводится дополнительная информация по настройкам линий связи при работе в сети.

ПРИМЕЧАНИЕ: Данная процедура может потребоваться вам только в том случае, если порт 2 подсоединен к сети. В других случаях нормальная работа обеспечивается параметрами настройки, установленными по умолчанию.

Используйте функцию AUX 56 для установки сетевого адреса и параметров связи порта. Вы можете также выполнять эту операцию в *Direct*SOFT через подменю «PLC/Setup».

AUX 57 — установить области в памяти для хранения данных

Процессорные модули контроллера DL205 определяют по умолчанию определенные области в памяти для хранения различных данных. Установленные по умолчанию области пригодны для многих приложений, но вы можете изменить их, если ваше приложение требует дополнительных областей памяти или вообще никаких областей. Областями по умолчанию являются:

Области памяти	DL230		DL240		DL250-1		DL260	
	Область по умолчанию	Доступная область	Область по умолчанию	Доступная область	Область по умолчанию	Доступная область	Область по умолчанию	Доступная область
Управляющие реле	C300 - C377	C0 - C377	C300 - C377	C0 - C377	C1000 - C1777	C0 - C1777	C1000 - C1777	C0 – C3777
V-память	V2000 - V7777	V0 - V7777	V2000 - V7777	V0 - V7777	V1400 - V3777	V0-V17777	V1400 - V3777	V0-V37777
Таймеры	Нет	T0-T77	Нет	TO -T177	Нет	T0 - T377	Нет	T0 - T377
Счетчики	CT0 - CT77	CT0 - CT77	CTO - CT177	CT0 - CT177	CT0 - CT177	CT0 - CT177	CT0 - CT177	CT0 - CT377
Стадии	Нет	SO - S377	Нет	S0 - S777	Нет	SO - S1777	Нет	S0 - S1777

Используйте функцию AUX 57 с целью изменения областей памяти для хранения этих данных. Вы можете также выполнять эту операцию в *Direct*SOFT через подменю «PLC/Setup».

ПРЕДУПРЕЖДЕНИЕ: *Процессоры DL205 не поставляются с батареями. Конденсатор большой емкости при потере питания поддержит записанные в память значения, но не более 1 недели. При некоторых условиях продолжительность сохранения может быть еще меньше. Если области хранения данных важны для вашего приложения, то приобретите батареи, поставляемые по отдельному заказу*

AUX 58 — операции тестирования

В Рабочем режиме при переходе в Программный режим все выходы сбрасываются. В режиме TEST-RUN вы можете установить каждый конкретный выход либо на сброс, либо на сохранение последнего состояния выхода при переходе в режим TEST-PRG. Способность сохранения состояний выходов чрезвычайно полезна, так как дает возможность сохранить для контроля ключевые системные параметры точек ввода/вывода.

Используйте функцию AUX 58 для конфигурирования каждого отдельного выхода. Вы можете также выполнять эту операцию в *Direct*SOFT через подменю «PLC/Setup».

AUX 59 — форсирование бита

Форсирование бита может быть включено для каждой точки с помощью функции AUX 59 с Ручного Программатора или с помощью опций меню *Direct*SOFT. По существу при форсировании бита процессорный модуль не может делать какие-либо изменения в точке дискретного сигнала. Например, если вы включили подавление бита для X1, и X1 был в это время в отключенном состоянии, то процессорный модуль не сможет изменить состояние X1. Это означает, что даже когда X1 включается, процессор не допустит его изменения. Поэтому в данном случае, если X1 используется в программе, то его всегда следует рассматривать как «отключенный». Само собой разумеется, если X1 был включен при установлении подавления бита, то его всегда следует рассматривать как «включенный».

ПРИМЕЧАНИЕ: *Протокол обмена данными DirectNet не поддерживает операции записи отдельного бита.*

Использование функции форсирования бита иногда очень полезно. Например, если вы включили форсирование бита для Y0, а он был в это время отключен, то процессорный модуль не сможет изменить состояние Y0. Однако, вы можете все же, используя программирующее устройство, изменить это состояние. Если вы теперь воспользуетесь программирующим устройством для принудительного включения Y0, то этот выход включится, а процессорный модуль менять его состояние не будет. Далее, если вы принудительно отключите Y0, то процессор будет поддерживать Y0 как «отключенный». Процессор никогда не обновит эту точку по результатам работы прикладной программы или по обновлению ввода/вывода, пока подавление бита не будет снято.

На следующей схеме дается общее представление функции форсирования бита. Следует заметить, что процессорный модуль не обновляет регистр отображения при включенной функции форсирования бита.

AUX 5B — конфигурация интерфейса счетчика

Функция AUX 5B применяется к модулю интерфейса счетчика DL205 при выборе его конфигурации. Вы можете выбрать тип счетчика, установить параметры счетчика и т. д. См. руководство по модулю интерфейса счетчика DL205 с полным описанием инструкций по выбору характеристик счетчика.

AUX 5C — вывести журнал ошибок

Процессорные модули DL240, DL250-1 и DL260 автоматически регистрируют коды любых системных ошибок и созданные вами с помощью команд FAULT сообщения. Процессорный модуль регистрирует код ошибки, дату и время, когда произошла ошибка. Имеются две отдельные таблицы, в которых хранится эта информация.

- **Таблица кодов ошибок** — система регистрирует в этой таблице до 32 ошибок. Когда возникает ошибка, то все уже записанные в таблицу ошибки сдвигаются вниз, а последняя ошибка загружается в верхнюю ячейку. Если таблица полностью заполнена при появлении ошибок, то самая ранняя ошибка убирается (стирается) из таблицы.

- **Таблица сообщений** — в этой таблице система регистрирует до 16 сообщений. Когда сообщение инициируется, то все уже записанные в таблицу сообщения сдвигаются вниз, а последнее сообщение загружается в верхнюю ячейку. Если таблица полностью заполнена при появлении сообщения, то самое раннее сообщение убирается (стирается) из таблицы.

Далее показан пример таблицы сообщений по ошибкам.

Дата	Время	Сообщение
1993-05-26	08:41:51:11	*Конвейер-2 остановлен
1993-04-30	17:01:11:56	*Конвейер-1 остановлен
1993-04-30	17:01:11:12	*Габарит SW1 нарушен
1993-04-28	03:25:14:31	*Заклинивание пилы

Вы можете использовать функцию AUX 5C для просмотра кодов и сообщений об ошибках. Вы можете также просмотреть эти ошибки и сообщения в *Direct*SOFT через подменю «PLC/Diagnostics».

AUX 6* — конфигурирование Ручного Программатора

AUX 61-65

Вы можете использовать несколько AUX функций для настройки, просмотра или изменения конфигурации Ручного Программатора.

- AUX 61 — показать номер версии
- AUX 62 — включить/отключить звуковой сигнал
- AUX 65 — запустить самодиагностику
-

AUX 61 — показать номер версии

Большинство продуктов, предназначенных для производственного управления, как правило, дорабатываются — появляются дополнительные функции, улучшаются старые. Иногда такие новые функции имеются только в определенных версиях программно-аппаратных средств. Используя функцию AUX 61, вы можете быстро просмотреть номера версий программно-аппаратных средств процессорного модуля и Ручного Программатора. Эта информация (по процессорному модулю) доступна также в *Direct*SOFT через подменю «PLC/Diagnostics».

AUX 62 — включить/отключить звуковой сигнал

Ручной Программатор имеет устройство звуковой сигнализации для подтверждения нажатия клавиш. Вы можете использовать вспомогательную функцию AUX 62 для отключения звуковой сигнализации.

AUX 65 — запустить самодиагностику

Если вы не уверены, что Ручной Программатор работает правильно, то можете воспользоваться функцией AUX 65, чтобы запустить программу самодиагностики. Вы можете проверить:

- клавиатуру
- дисплей
- светодиоды и лампы подсветки
- электрически перепрограммируемое ПЗУ (ЭППЗУ) Ручного Программатора.

AUX 7* — операции с ЭППЗУ

AUX 71-76

Вы можете использовать несколько AUX функций для перемещения программ из памяти процессорного модуля в ЭППЗУ Ручного Программатора (HHP) и обратно.

- AUX 71 — считать из памяти процессорного модуля в ЭППЗУ HPP
- AUX 72 — записать ЭППЗУ HPP в процессорный модуль
- AUX 73 — сравнить память процессорного модуля и ЭППЗУ HPP
- AUX 74 — проверить очистку памяти (ЭППЗУ HPP)
- AUX 75 — стереть ЭППЗУ HPP
- AUX 76 — показать тип ЭППЗУ (процессорного модуля и HPP)

Переносимые области памяти

Многие из указанных AUX функций позволяют вам копировать различные области памяти из процессора в Ручной Программатор и наоборот. В таблице ниже приведены области памяти, которые могут копироваться.

Опция и тип памяти	DL240 Область памяти по умолчанию	DL230 Область памяти по умолчанию
1: PGM - Программная	$00000 - $02559	$00000 - $02047
2: V - V-память	$00000 - $4777	$00000 - $04777
3: SYS - Системная	Невыбираемые копии системных параметров	
4: etc(ALL) - Программная, Системная и неразрушаемая V-	Не выбирается	

AUX 71 — считать из памяти процессорного модуля в ЭППЗУ HPP

Функция AUX 71 копирует информацию из памяти процессора в ЭППЗУ Ручного Программатора.

Вы можете копировать различные части памяти процессора в память ЭППЗУ (HPP) в соответствии с приведенной таблицей. Объем данных, которые вы можете скопировать, зависит от процессорного модуля.

AUX 72 — записать ЭППЗУ HPP в процессорный модуль

Функция AUX 72 копирует информацию из ЭППЗУ Ручного Программатора в процессорный модуль. Вы можете копировать различные типы информации из памяти ЭППЗУ Ручного Программатора в соответствии с приведенной таблицей.

AUX 73 — сравнить ЭППЗУ процессора и HPP

Функция AUX 73 сравнивает программы на Ручном Программаторе (в ЭППЗУ) с программамив процессорном модуле. Вы можете сравнивать различные типы информации в соответствии с приведенной таблицей. Опция, названная в таблице «etc(ALL)», позволяет вам последовательно проверить все области без перезапуска AUX функций каждый раз.

AUX 74 — проверить очистку памяти (ЭППЗУ HPP)

Функция AUX 74 дает вам возможность проверить ЭППЗУ, установленное в Ручном Программаторе, чтобы убедиться, что она чистая. Эту функцию желательно использовать всякий раз, когда вы начинаете целиком копировать программы в ЭППЗУ Ручного Программатора.

AUX 75 — стереть ЭППЗУ HPP

Функция AUX 75 позволяет вам удалить все данные из ЭППЗУ Ручного Программатора. Вы должны применять эту AUX функцию перед копированием программ из процессора.

AUX 76 — показать тип ЭППЗУ (процессорного модуля и HPP)

Функцию AUX 76 можно использовать для быстрого определения объема ЭППЗУ, установленного в процессорном модуле и в Ручном Программаторе. В DL230 и DL240 использованы ЭППЗУ различного объема. Для дополнительной информации обратитесь к главе 3.

AUX 8* — операции с паролем

AUX 81-83

Вы можете использовать несколько AUX функций для изменения пароля процессорного модуля или для включения парольной защиты. Вы можете применять эти функции при диалоговом общении с процессорным модулем или применять их в автономном режиме при работе с ЭППЗУ, установленном в Ручном Программаторе. С помощью Ручного Программатора вы можете разработать программу и включить защиту с помощью пароля.

AUX 81 — изменить пароль

AUX 82 — разблокировать процессорный модуль

AUX 83 — заблокировать процессорный модуль

AUX 81 — изменить пароль

Функция AUX 81 позволяет вам ввести особую меру защиты посредством введения пароля, с помощью которого предупреждаются несанкционированные машинные операции. Паролем должен быть 8-значный цифровой (0 -9) код. Пароль может быть удален введением вместо него всех нулей (00000000). Это значение по умолчанию, устанавливается в заводских условиях.

После введения пароля вы можете блокировать доступ к процессорному модулю. Блокировать процессорный модуль с Ручного Программатора можно двумя способами:

- Процессорный модуль блокируется каждый раз после цикла включения питания (если пароль введен).
- Вы используете функции AUX 83 и AUX 82 для блокировки и разблокировки процессорного модуля.

Вы можете также вводить и изменять пароль в *Direct*SOFT через подменю «PLC/Password».

В *Direct*SOFT эта функция работает несколько иначе. Если пароль введен, то процессорный модуль автоматически блокируется, когда вы выходите из программного пакета. Он блокируется также при включении питания.

ПРЕДУПРЕЖДЕНИЕ: *Перед тем, как вы заблокируете процессорный модуль, убедитесь, что вы запомнили пароль. После блокировки процессорного модуля вы не сможете ни посмотреть, ни изменить, ни стереть этот пароль. Если вы забыли пароль, то процессорный модуль придется возвратить на завод для снятия защиты.*

ПРИМЕЧАНИЕ: *Процессорные модули D2-240, D2-250-1 и D2-260 поддерживают защиту пользовательских программ многоуровневым паролем без блокировки порта связи с интерфейсом оператора. Многоуровневый пароль включает прописную «A» и далее семь цифровых символов (например, A1234567).*

AUX 82 — разблокировать процессорный модуль

Функция AUX 82 может использоваться для разблокировки процессора, защищенного паролем. *Direct*SOFT автоматически попросит вас ввести пароль, если вы попытаетесь взаимодействовать с процессорным модулем, содержащим пароль.

AUX 83 — заблокировать процессорный модуль

Функция AUX 83 может использоваться для блокировки процессорного модуля, который содержит пароль. Если процессорный модуль блокирован, вам необходимо ввести пароль, чтобы получить доступ к нему. Напоминаем, что в *Direct*SOFT эта функция не является необходимой, так как в этом случае процессорный модуль автоматически блокируется, как только вы выходите из пакета программирования.

Приложение B. ***Коды ошибок DL205***

В этом приложении…

- Таблица кодов ошибок

Таблица кодов ошибок

Код ошибки	Описание
E003 SOFTWARE TIME-OUT	Эта ошибка возникнет, если длительность цикла сканирования программы превысит время, заданное в сторожевом таймере. SP51 включится, а код ошибки будет храниться в V7755. Чтобы исправить эту ошибку, добавьте команды RSTWT в циклы FOR/NEXT и подпрограммы, или используйте функцию AUX 55 для увеличения контрольного времени в сторожевом таймере.
E041 CPU BATTERY LOW	Батарея процессорного модуля разряжена и должна быть заменена. Включится SP43, а код ошибки будет сохранен в V7757.
E099 PROGRAM MEMORY EXCEEDED	Эта ошибка возникнет, если размер скомпилированной программы превысит размер ОЗУ процессорного модуля. Включится SP52, а код ошибки будет сохранен в V7755. Сократите размер прикладной программы.
E104 WRITE FAILED	Запись в процессорный модуль оказалась неудачной. Отключите питание, вытащите процессорный модуль из каркаса и убедитесь, что не установлена защита от записи. Если защита от записи не установлена, убедитесь, что EEPROM установлена правильно. Если оба условия выполнены, замените процессорный модуль.
E151 BAD COMMAND	В прикладной программе возникла ошибка четности. Включится SP44, и код ошибки будет сохранен в V7755. Эта проблема может возникнуть из-за электрической помехи. Очистите память и загрузите программу снова. Исправьте любые проблемы с заземлением. Если ошибка возникла вновь, замените EEPROM или процессорный модуль целиком.
E155 RAM FAILURE	В системном ОЗУ возникла ошибка контрольной суммы. Включится SP44, а код ошибки будет сохранен в V7755. Эта проблема может возникнуть из-за низкого напряжения батареи, электрические помехи или неисправности ОЗУ процессорного модуля. Очистите память и загрузите программу снова. Исправьте любые проблемы с заземлением. Если ошибка возникла вновь, замените процессорный модуль.
E202 MISSING I/O MODULE	Модуль ввода/вывода не сумел связаться с процессорным модулем или он отсутствует в разъеме. Включится SP45, а код ошибки будет сохранен в V7756. Запустите функцию AUX42, чтобы определить слот и каркас, в которых расположен модуль, сообщающий об ошибке.
E210 POWER FAULT	Возникло краткое пропадание напряжения в сети питания контроллера
E250 COMMUNICATION FAILURE IN THE I/O CHAIN	Произошел сбой в локальной системе ввода/вывода. Проблема может быть в шине ввода/вывода или в источнике питания каркаса. Включится SP45, а код ошибки будет сохранен в V7755. Запустите функцию AUX42, чтобы определить каркас, сообщающий об ошибке.
E252 NEW I/O CFG	Эта ошибка возникает, когда в процессорном модуле включена автоматическая проверка конфигурации, а текущая конфигурация ввода/вывода изменилась из-за перемещения модулей в каркасе или из-за изменения типа модулей. Вы можете вернуть модули в первоначальное положение или запустить функцию AUX45, чтобы принять новую конфигурацию. Включится SP47, а код ошибки будет сохранен в V7755.
E262 I/O OUT OF RANGE	В прикладной программе встретился адрес, выходящий за границы диапазона ввода/вывода. Исправьте неправильный адрес в программе. Включится SP45, а код ошибки будет сохранен в V7755.
E312 HP COMM ERROR 2	Произошла ошибка в данных при обмене с процессорным модулем. Очистите ошибку и повторите запрос. Если ошибка повторяется, проверьте кабельное соединение между двумя устройствами, замените Ручной Программатор, затем, если необходимо, замените процессорный модуль. Включится SP46, а код ошибки будет сохранен в V7756.

Код ошибки	Описание
E313 HP COMM ERROR 3	Произошла ошибка в данных при обмене с процессорным модулем. Очистите ошибку и повторите запрос. Если ошибка повторяется, проверьте кабельное соединение между двумя устройствами, замените Ручной Программатор, затем, если необходимо, замените процессорный модуль. Включится SP46, а код ошибки будет сохранен в V7756.
E316 HP COMM ERROR 6	Произошла ошибка режима в течение связи с процессорным модулем. Очистите ошибку и повторите запрос. Если ошибка повторяется, замените Ручной Программатор, затем, если необходимо, замените процессорный модуль. Включится SP46, а код ошибки будет сохранен в V7756.
E320 HP COMM TIME-OUT	Процессорный модуль не ответил на запрос связи Ручного Программатора. Проверьте кабель, убедитесь, что он исправлен и правильно подсоединен. Повторите цикл включения питания системы и, если ошибка продолжает повторяться, сначала замените процессор, затем, если это будет необходимо, Ручной Программатор.
E321 COMM ERROR	Произошла ошибка в данных при обмене с процессорным модулем. Проверьте кабель, убедитесь, что он исправлен и правильно подсоединен. Повторите цикл включения питания системы и, если ошибка продолжает повторяться, сначала замените процессор, затем, если это будет необходимо, Ручной Программатор.
E4 NO PROGRAM**	В прикладной программе существует синтаксическая ошибка. Чаще всего встречается ошибка — отсутствие оператора END. Запустите функцию AUX21, чтобы определить, какая из ошибок серии E4** произошла. Включится SP52, а код ошибки будет сохранен в V7755.
E401 MISSING END STATEMENT	Все прикладные программы должны заканчиваться оператором END. Введите оператор END в соответствующее место в вашей программе. Включится SP52, а код ошибки будет сохранен в V7755.
E402 MISSING LBL	Команды GOTO, GTS, MOVMC или LDLBL были использованы без соответствующей метки. Обратитесь к руководству по программированию для уточнения требований к этим командам. Включится SP52, а код ошибки будет сохранен в V7755.
E403 MISSING RET (DL240 ONLY)	Подпрограмма в программе не заканчивается командой RET. Включится SP52, а код ошибки будет сохранен в V7755.
E404 MISSING FOR (DL250-1, DL260)	Команда NEXT не имеет соответствующей команды FOR. Включится SP52, а код ошибки будет сохранен в V7755.
E405 MISSING NEXT (DL240/DL250-1/260)	Команда FOR не имеет соответствующей команды NEXT. Включится SP52, а код ошибки будет сохранен в V7755.
E406 MISSING IRT	Подпрограмма прерывания в программе не заканчивается командой IRT. Включится SP52, а код ошибки будет сохранен в V7755.
E412 SBR/LBL>64 (DL240/250-1/260)	В программе больше 64 команд SBR, LBL или DLBL. Эта ошибка также возникает, если в программе используется больше 128 команд GTS или GOTO. Включится SP52, а код ошибки будет сохранен в V7755.
E413 FOR/NEXT>64 (DL24/.DL250-1/260)	В прикладной программе больше 64 циклов FOR/NEXT. Включится SP52, а код ошибки будет сохранен в V7755.
E421 DUPLICATE STAGE REFERENCE	В прикладной программе существуют две или более метки SG или ISG с одинаковым номером. У каждой стадии и начальной стадии должен быть свой уникальный номер. Включится SP52, а код ошибки будет сохранен в V7755.
E422 DUPLICATE SBR/LBL REFERENCE	В прикладной программе существуют две или более команды SBR или LBL с одинаковым номером. У каждой подпрограммы и метки должен быть свой уникальный номер. Включится SP52, а код ошибки будет сохранен в V7755.

Код ошибки	Описание
E423 NESTED LOOPS (DL240/DL250-1/260)	Вложенные циклы (программирование одного цикла FOR/NEXT внутри другого) не допускаются в процессорных модулях DL240/250-1/260. Включится SP52, а код ошибки будет сохранен в V7755.
E431 INVALID ISG/SG ADDRESS	ISG или SG не должны быть запрограммированы после конечного оператора END, как это делается в случае с подпрограммой. Включится SP52, а код ошибки будет сохранен в V7755.
E432 INVALID JUMP (GOTO) ADDRESS (DL240/DL250-1/260	Метка LBL, соответствующая команде GOTO не должна при программировании ставиться после оператора END, например, в подпрограмме. SP52 включается, а код ошибки записывается в V7755.
E433 INVALID SBR ADDRESS (DL240/DL250-1/260)	SBR должна при программировании ставиться после оператора END, но не в главной программе и не в программе прерывания. SP52 включается, а код ошибки записывается в V7755.
E435 INVALID RT ADDRESS (DL240/DL250-1/260)	RT должна при программировании ставиться после оператора END, но не в главной программе и не в программе прерывания. SP52 включается, а код ошибки записывается в V7755.
E436 INVALID INT ADDRESS	INT должна при программировании ставиться после оператора END, но не в главной программе. SP52 включается, а код ошибки записывается в V7755.
E438 INVALID IRT ADDRESS	IRT должна при программировании ставиться после оператора END, но не в главной программе. SP52 включается, а код ошибки записывается в V7755..
E440 INVALID DATA ADDRESS	Или команда DLBL была запрограммирована в главном теле программы (не после оператора END), или команда DLBL находится в звене, содержащем входные контакты.
E441 ACON/NCON (DL240/DL250-1/260)	ACON или NCON должны при программировании ставиться после оператора END, не в главной программе. SP52 включается, а код ошибки записывается в V7755.
E451 BAD MLS/MLR	Команды MLS должны нумероваться в возрастающем порядке сверху вниз.
E452 X AS COIL	Тип данных X (вход) используется как выход.
E453 MISSING T/C	Использован контакт таймера или счетчика, хотя соответствующего таймера или счетчика не существует.
E454 BAD TMRA	В команде TMRA пропущен один из контактов.
E455 BAD CNT	В командах CNT или UDC пропущен один из контактов.
E456 BAD SR	В команде SR пропущен один из контактов.
E461 STACK OVERFLOW	В стеке записано более девяти логических уровней. Проверьте использование команд OR STR И AND STR.
E462 STACK UNDERFLOW	В стеке находится несогласованное число логических уровней. Убедитесь, что число команд OR STR И AND STR соответствует числу команд STR.
E463 LOGIC ERROR	Команда STR/STRN отсутствует в начале цепи релейной логики.
E464 MISSING CKT	Цепь релейной логики не завершена надлежащим образом.

Код ошибки	Описание
E471 DUPLICATE COIL REFERENCE	Две или большее число команд OUT имеют ссылки на одну и ту же точку Ввода/Вывода.
E472 DUPLICATE TMR REFERENCE	Две или большее число команд TMR имеют ссылки на один и тот же номер.
E473 DUPLICATE CNT REFERENCE	Две или большее число команд CNT имеют ссылки на один и тот же номер.
E480 INVALID CV ADDRESS	Команда CV используется в подпрограмме или в программе обработки прерываний. Команда CV может использоваться только в области главной программы (перед оператором END).
E481 CONFLICTING INSTRUCTIONS	Существует команда между сходящимися стадиями.
E482 MAX. CV INSTRUCTIONS EXCEEDED	Число команд CV превышает 17.
E483 INVALID CVJMP ADDRESS	Команда CVJMP использовалась в подпрограмме или в программе обработке прерываний.
E484 MISSING CV INSTRUCTION	Перед командой CVJMP нет соответствующей команды CV. CVJMP должен следовать сразу за командой CV.
E485 NO CVJMP	Команда CVJMP не размещена между командами CV и SG, ISG, BLK, BEND, END.
E486 INVALID BCALL ADDRESS	Команда BCALL используется в подпрограмме или программе прерывания. Команда BCALL может быть использована только в главном теле программы (перед оператором END).
E487 MISSING BLK INSTRUCTION	Команда BCALL не следует за командой BLK.
E488 INVALID BLK ADDRESS	Команда BLK используется в подпрограмме или программе прерывания. Другая команда BLK используется между командами BCALL и BEND.
E489 DUPLICATED CR REFERENCE	Управляющее реле, используемое для команды BLK, было использовано как выход еще в другом месте.
E490 MISSING SG INSTRUCTION	Команда BLK не следует непосредственно за командой SG.
E491 INVALID ISG INSTRUCTION ADDRESS	Между командами BLK и BEND находится команда ISG.
E492 INVALID BEND ADDRESS	Команда END BLK используется в подпрограмме или программе обработки прерываний. Команда END BLK инструкция не имеет предшествующей команды ST BLK.
E493 MISSING REQUIRED INSTRUCTION	Команда [CV, SG, ISG, BLK, BEND] должна непосредственно следовать за командой BEND.
E494 MISSING BEND INSTRUCTION	За командой BLK не следует команда BEND.

Код ошибки	Описание
E499 PRINT INSTRUC-TION	Неправильное использование команды PRINT. Кавычки и/или пробелы не были введены или введены неправильно.
E501 BAD ENTRY	В Ручной Программатор были введены неправильные последовательности клавиш.
E502 BAD ADDRESS	В Ручной Программатор был введен неправильный или выходящий за границы диапазона адрес.
E503 BAD COMMAND	В Ручной Программатор была введена неправильная команда.
E504 BAD REF/VAL	Вместе с командой были введены неправильное значение или номер ссылки.
E505 INVALID INSTRUCTION	В Ручной Программатор была введена недопустимая команда.
E506 INVALID OPERA-TION	На Ручном Программаторе сделана попытка ввести неправильную операцию..
E520 BAD OP-RUN	На Ручном Программаторе сделана попытка ввести операцию, которая неверна в Рабочем режиме(RUN).
E521 BAD OP-TRUN	На Ручном Программаторе сделана попытка ввести операцию, которая неверна в режиме Запуска теста (TEST RUN).
E523 BAD OP-TPGM	На Ручном Программаторе сделана попытка ввести операцию, которая неверна в режиме Тестирования программ (TEST PROGRAM).
E524 BAD OP-PGM	На Ручном Программаторе сделана попытка ввести операцию, которая неверна в Программном режиме (PGM).
E525 MODE SWITCH (DL240/DL250-1/260)	На Ручном Программаторе сделана попытка выполнить операцию, когда переключатель режимов находится в положении, отличном от TERM.
E526 OFF LINE	Ручной Программатор находится в АВТОНОМНОМ (OFF LINE) режиме. Для изменения режима на ОПЕРАТИВНЫЙ (ON LINE) используйте клавишу MODE.
E527 ON LINE	Ручной Программатор находится в ОПЕРАТИВНОМ (ON LINE) режиме. Для изменения режима на АВТОНОМНЫЙ (OFF LINE) используйте клавишу MODE.
E528 CPU MODE	Операция, которую пытались запустить, не допустима в режиме Run Time Edit (Редактирование в Рабочем режиме).
E540 CPU LOCKED	Процессор блокирован паролем. Для разблокирования Процессора используйте функцию AUX 82, позволяющую работать с паролем.
E541 WRONG PASS-WORD	Неверный пароль, который используется для разблокирования Процессора с помощью AUX 82.
E542 PASSWORD RE-SET	Процессор включен с неправильным паролем и установил его значение в 00000000. Пароль может быть введен повторно с помощью AUX 81
E601 MEMORY FULL	Выдается при попытке ввести команду, которая требует больше памяти, чем память, имеющаяся в Процессоре.
E602 INSTRUCTION MISSING	Функция поиска не нашла данную команду.
E604 REFERENCE MISSING	Была выполнена функция поиска, и ссылка не была найдена.
E610 BAD I/O TYPE	Прикладная программа отметила модуль ввода/вывода как неправильный тип модуля.
E620	Была предпринята попытка передать больше данных между процессорным

Код ошибки	Описание
OUT OF MEMORY	модулем и Ручным Программатором, чем принимающее устройство может вместить.
E621 EEPROM NOT BLANK	Была сделана попытка записи в непустой ЭППЗУ Ручного Программатора. Очистите ЭППЗУ и затем повторите запись.
E622 NO HPP EEPROM	Была сделана попытка передачи данных в отсутствующее ЭППЗУ Ручного Программатора (или в его неисправное ЭППЗУ).
E623 SYSTEM EEPROM	Была запрошена функция с ЭППЗУ Ручного Программатора, которая содержит только системную информацию
E624 V-MEMORY ONLY	Была запрошена функция с ЭППЗУ Ручного Программатора, которая содержит только данные V-памяти.
E625 PROGRAM ONLY	Была запрошена функция с ЭППЗУ Ручного Программатора, которая содержит только данные программ.
E627 BAD WRITE	Была сделана попытка записи в неисправную ЭППЗУ Ручного Программатора. Проверьте и, если необходимо, замените ЭППЗУ.
E628 EEPROM TYPE ERROR	ЭППЗУ с несоответствующим объемом памяти.
E640 COMPARE ERROR	При сравнении ЭППЗУ и Процессора была обнаружена ошибка.
E650 HPP SYSTEM ERROR	Ошибка была обнаружена при переносе данных в Картридж памяти программатора. Проверьте соединения и повторите операцию.
E651 HPP ROM ERROR	В Ручном Программаторе имеет место ошибка ПЗУ. Повторно включите Ручной Программатор. Если ошибка появится снова, замените Ручной Программатор
E652 HPP RAM ERROR	В Ручном Программаторе имеет место ошибка ОЗУ. Повторно включите Ручной Программатор. Если ошибка появится снова, замените Ручной Программатор

Приложение C. ***Длительность выполнения команд***

В этом приложении…

Введение

В данном приложении приводятся таблицы времени выполнения команд для процессорных модулей контроллера DL205. Вы можете заметить, что многие из приведенных времен зависят от типа данных, используемого в команде. Например, некоторые команды, использующие ячейки V-памяти, определяются далее следующими элементами:

- Регистрами данных (слово)
- Битовыми регистрами

Регистры данных V-памяти

Некоторые ячейки V-памяти рассматриваются как регистры данных. Например, ячейки V-памяти, в которых хранятся текущие значения счетчика или таймера, или ячейки V-памяти постоянного пользователя, можно рассматривать как регистры данных. Не подумайте, что нельзя загрузить битовую комбинацию в регистры этого типа, - можно. Но основное их назначение — регистры данных. Следующие ячейки рассматриваются как регистры данных.

Регистры данных	DL230	DL240	DL250-1	DL260
Текущие Значения Таймера	V0-V77	V0-V177	V0 - V377	V0 - V377
Текущие Значения Счетчика	V1000-V1077	V1000-V1177	V1000-V1177	V1000-V1377
Слова пользовательских данных	V2000 - V2377 V4000-V4177	V2000 - V3777 V4000 - V4377	V1400 – V7377 V10000 – V17777	V400 – V777 V1400 – V7377 V10000 – V35777

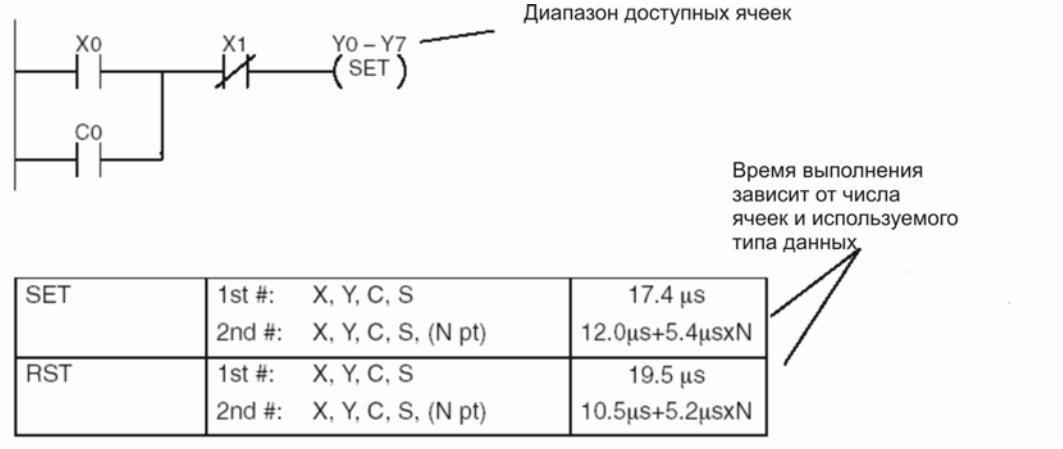
Битовые регистры V-памяти

Некоторые из дискретных точек, например, X, Y, C и др., автоматически отображаются в V-памяти. Следующие ячейки, содержащие такие данные, рассматриваются как регистры битов.

Регистры битов	DL230	DL240	DL250-1	DL260
Входные точки (X)	V40400 - V 40407	V40400 - V 40407	V40400 - V 40437	V40400 - V 40477
Выходные точки (Y)	V40500-V40507	V40500-V40507	V40500-V40537	V40500-V40577
Управляющие Реле (C)	V40600-V40617	V40600-V40617	V40600-V40677	V40600-V40777
Биты состояния Таймера	V41100-V41103	V41100-V41107	V41100-V41117	V41100-V41117
Биты состояния Счетчика	V41040-V41143	V41040-V41147	V41040-V41147	V41140-V41157
Стадии	V41000-V41017	V41000-V41037	V41000-V41077	V41000-V41077

Как читать таблицы

Некоторые команды могут иметь несколько параметров, поэтому в таблице приведены времена выполнения в зависимости от числа и типа параметров. Например, команду SET можно применить для установки одной точки или группы точек. Если вы просмотрите таблицу, вы найдете возможные типы данных и времена выполнения для обоих этих случаев. Следующая схема иллюстрирует этот пример.



SET	1st #: X, Y, C, S	17.4 µs
	2nd #: X, Y, C, S, (N pt)	12.0µs+5.4µsxN
RST	1st #: X, Y, C, S	19.5 µs
	2nd #: X, Y, C, S, (N pt)	10.5µs+5.2µsxN

Булевы команды

Булевы команды		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
STR	X, Y, C, T, CT,S, SP	3.3 µs	3.3 µs	1.4 µs	1.4 µs	.67 µs	0 µs	.67 µs	0 µs
STRN	X, Y, C, T, CT,S, SP	3.9 µs	3.9 µs	1.6 µs	1.6 µs	.67 µs	0 µs	.67 µs	0 µs
OR	X, Y, C, T, CT, S, SP	2.7 µs	2.7 µs	1.0 µs	1.0 µs	.51 µs	.51 µs	.51 µs	.51 µs
ORN	X, Y, C, T, CT,S, SP	3.3 µs	3.3 µs	1.4 µs	1.4 µs	.55 µs	.55 µs	.55 µs	.55 µs
AND	X, Y, C, T, CT, S, SP	2.1 µs	2.1 µs	0.8 µs	0.8 µs	.42 µs	.42 µs	.42 µs	.42 µs
ANDN	X, Y, C, T, CT, S, SP	2.7 µs	2.7 µs	1.2 µs	1.2 µs	.51 µs	.51 µs	.51 µs	.51 µs
ANDSTR	None	1.2 µs	1.2 µs	0.7 µs	0.7 µs	.37 µs	.37 µs	.37 µs	.37 µs
ORSTR	None	1.2 µs	1.2 µs	0.7 µs	0.7 µs	.37 µs	.37 µs	.37 µs	.37 µs
OUT	X, Y, C	3.4 µs	3.4 µs	7.95 µs	7.65 µs	1.82 µs	1.82 µs	1.82 µs	1.82 µs
OROUT	X, Y, C	8.6 µs	8.6 µs	8.25 µs	8.4 µs	2.09 µs	2.09 µs	2.09 µs	2.09 µs
NOT		–	–	–	–	1.04 µs	1.04 µs	1.04 µs	1.04 µs
SET	1st #: X, Y, C, S	17.4 µs	6.8 µs	11.4 µs	8.4 µs	9.2 µs	1.0 µs	9.2 µs	1.0 µs
	2nd #: X, Y, C, S (N pt)	12.0µs+ 5.4µsxN	6.8 µs	11.0µs+ 7.0µsxN	8.4 µs	9.6 µs+ 0.9µsxN	1.1 µs	9.6 µs+ 0.9µsxN	1.1 µs
RST	1st #: X, Y, C, S	17.7 µs	6.8 µs	11.4 µs	8.4 µs	9.2 µs	1.0 µs	9.2 µs	1.0 µs
	2nd #: X, Y, C, S (N pt)	10.5µs+ 5.2µsxN	6.8 µs	11.0µs+ 7.0µsxN	8.4 µs	9.6 µs+ 0.9µsxN	1.1 µs	9.6 µs+ 0.9µsxN	1.1 µs
	1st #: T, CT	31.6 µs	6.8 µs	29.0 µs	8.4 µs	25.7 µs	1.1 µs	25.7 µs	1.1 µs
	2nd #: T, CT (N pt)	17µs+ 14.6µsx N	6.8 µs	24.3µs+ 4.7µsxN	8.4 µs	16.8µs+ 2.7µsxN	1.4 µs	16.8µs+ 2.7µsxN	1.4 µs
PAUSE	1wd: Y	19.0 µs	19.0 µs	13.0 µs	13.0 µs	5.6 µs	5.4 µs	5.6 µs	5.4 µs
	2wd: Y (N points)	15µs+ 4µs x N	15µs+4µs x N	11µs+3µs x N	11µs+3µs x N	9.2 µs+ 0.3µsxN	4.8 µs	9.2 µs+ 0.3µsxN	4.8 µs

Булевы команды сравнения

Булевы команды сравнения			DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных		Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
STRE	1st	2nd								
	V: Data Reg.	V:Data Reg.	77 µs	13.8 µs	46 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	158 µs	13.8 µs	135 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	57 µs	13.8 µs	46 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	141 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	235 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	158 µs	13.8 µs	135 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	240 µs	13.8 µs	225 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	139 µs	13.8 µs	135 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	231 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	324 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.7 µs	27.7 µs	27.7 µs	27.7 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.7 µs	27.7 µs	27.7 µs	27.7 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
STRNE	1st	2nd								
	V: Data Reg.	V:Data Reg.	77 µs	13.8 µs	46 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	158 µs	13.8 µs	136 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	57 µs	13.8 µs	46 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	141 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	235 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	158 µs	13.8 µs	135 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	240 µs	13.8 µs	225 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	139 µs	13.8 µs	135 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	231 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	324 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg.	—	—	—	—	30.3 µs	30.3 µs	30.3 µs	30.3 µs
		V:Bit Reg.	—	—	—	—	30.3 µs	30.3 µs	30.3 µs	30.3 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	—	—	30.3 µs	30.3 µs	30.3 µs	30.3 µs
		V:Bit Reg.	—	—	—	—	30.3 µs	30.3 µs	30.3 µs	30.3 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Булевы команды сравнения (продолжение)			**DL230**		**DL240**		**DL250-1**		**DL260**	
Команда	Разрешенные типы данных		Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
ORE	1st	2nd								
	V: Data Reg.	V:Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	140 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	234 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	239 µs	12.0 µs	223 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	137 µs	12.0 µs	133 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	230 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	324 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg.	—	—	—	—	30.3 µs	30.3 µs	30.3 µs	30.3 µs
		V:Bit Reg.	—	—	—	—	30.3 µs	30.3 µs	30.3 µs	30.3 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	50.4 µs	50.4 µs	50.4 µs	50.4 µs
		P:Indir. (Bit)	—	—	—	—	50.4 µs	50.4 µs	50.4 µs	50.4 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	—	—	30.3 µs	30.3 µs	30.3 µs	30.3 µs
		V:Bit Reg.	—	—	—	—	30.3 µs	30.3 µs	30.3 µs	30.3 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	50.4 µs	50.4 µs	50.4 µs	50.4 µs
		P:Indir. (Bit)	—	—	—	—	50.4 µs	50.4 µs	50.4 µs	50.4 µs
ORNE	1st	2nd								
	V: Data Reg.	V:Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	141 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	234 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	239 µs	12.0 µs	223 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	137 µs	12.0 µs	133 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	230 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	323 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Булевы команды сравнения (продолжение)			**DL230**		**DL240**		**DL250-1**		**DL260**	
Ко-ман-да	Разрешенные типы данных		Ис-пол-няе-мые	Неис-пол-няе-мые	Ис-пол-няе-мые	Неис-пол-няе-мые	Ис-пол-няе-мые	Неис-пол-няе-мые	Ис-пол-няе-мые	Неис-пол-няе-мые
ANDE	1st	2nd								
	V: Data Reg.	V:Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	139 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	233 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	239 µs	12.0 µs	223 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	137 µs	12.0 µs	133 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	229 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	322 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
ANDNE	1st	2nd								
	V: Data Reg.	V:Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	158 µs	12.0 µs	133 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	139 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	233 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	239 µs	12.0 µs	223 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	137 µs	12.0 µs	133 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	229 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	323 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Булевы команды сравнения (продолжение)			**DL230**		**DL240**		**DL250-1**		**DL260**	
Команда	Разрешенные типы данных		Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
STR	1st	2nd								
	T, CT	V:Data Reg.	78 µs	13.8 µs	46 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	158 µs	13.8 µs	135 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	57 µs	13.8 µs	46 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	141 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	235 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	1st	2nd								
	V: Data Reg.	V:Data Reg.	78 µs	13.8 µs	46 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	159 µs	13.8 µs	135 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	57 µs	13.8 µs	46 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	141 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	235 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	159 µs	13.8 µs	135 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	241 µs	13.8 µs	225 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	139 µs	13.8 µs	135 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	231 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	324 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Булевы команды сравнения (продолжение)			**DL230**		**DL240**		**DL250-1**		**DL260**	
Команда	Разрешенные типы данных		Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
STRN	1st	2nd								
	T, CT	V:Data Reg.	78 µs	13.8 µs	46 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	158 µs	13.8 µs	136 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	57 µs	13.8 µs	46 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	141 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	235 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	1st	2nd								
	V: Data Reg.	V:Data Reg.	78 µs	13.8 µs	46 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	159 µs	13.8 µs	135 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	57 µs	13.8 µs	46 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	141 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	235 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	159 µs	13.8 µs	136 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	241 µs	13.8 µs	225 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	139 µs	13.8 µs	135 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	231 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	324 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Булевы команды сравнения (продолжение)			**DL230**		**DL240**		**DL250-1**		**DL260**	
Команда	Разрешенные типы данных		Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
OR	1st	2nd								
	T, CT	V:Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	140 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	234 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	1st	2nd								
	V: Data Reg.	V:Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	140 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	234 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	240 µs	12.0 µs	223 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	137 µs	12.0 µs	133 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	230 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	323 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Булевы команды сравнения (продолжение)			**DL230**		**DL240**		**DL250-1**		**DL260**	
Команда	Разрешенные типы данных		Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
ORN	1st	2nd								
	T, CT	V:Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	140 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	234 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	1st	2nd								
	V: Data Reg.	V:Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	141 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	234 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	240 µs	12.0 µs	223 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	137 µs	12.0 µs	133 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	230 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	324 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Булевы команды сравнения (продолжение)			**DL230**		**DL240**		**DL250-1**		**DL260**	
Команда	Разрешенные типы данных		Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
AND	1st	2nd								
	T, CT	V:Data Reg.	76 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	139 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	233 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	1st	2nd								
	V: Data Reg.	V:Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	140 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	233 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	240 µs	12.0 µs	223 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	137 µs	12.0 µs	133 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	229 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	323 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
ANDN	1st	2nd								
	T, CT	V:Data Reg.	76 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	139 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	233 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	1st	2nd								
	V: Data Reg.	V:Data Reg.	76 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	139 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	233 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V:Bit Reg.	240 µs	12.0 µs	223 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K:Constant	137 µs	12.0 µs	133 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P:Indir. (Data)	—	—	229 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	322 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V:Bit Reg.	—	—	—	—	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K:Constant	—	—	—	—	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P:Indir. (Data)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P:Indir. (Bit)	—	—	—	—	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Булевы команды с битом из слова

Булевы команды бит из слова		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
STRB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — —	— — —	— — —	— — —	3.1 µs 3.1 µs 30.0 µs 30.0 µs	3.1 µs 3.1 µs 30.0 µs 30.0 µs	3.1 µs 3.1 µs 30.0 µs 30.0 µs	3.1 µs 3.1 µs 30.0 µs 30.0 µs
STRNB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — —	— — —	— — —	— — —	3.0 µs 3.0 µs 29.8 µs 29.8 µs	3.0 µs 3.0 µs 29.8 µs 29.8 µs	3.0 µs 3.0 µs 29.8 µs 29.8 µs	3.0 µs 3.0 µs 29.8 µs 29.8 µs
ORB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — —	— — —	— — —	— — —	2.9 µs 2.9 µs 29.9 µs 29.9 µs	2.9 µs 2.9 µs 29.9 µs 29.9 µs	2.9 µs 2.9 µs 29.9 µs 29.9 µs	2.9 µs 2.9 µs 29.9 µs 29.9 µs
ORNB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — —	— — —	— — —	— — —	2.8 µs 2.8 µs 29.6 µs 29.6 µs	2.8 µs 2.8 µs 29.6 µs 29.6 µs	2.8 µs 2.8 µs 29.6 µs 29.6 µs	2.8 µs 2.8 µs 29.6 µs 29.6 µs
ANDB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — —	— — —	— — —	— — —	2.8 µs 2.8 µs 29.6 µs 29.6 µs	2.8 µs 2.8 µs 29.6 µs 29.6 µs	2.8 µs 2.8 µs 29.6 µs 29.6 µs	2.8 µs 2.8 µs 29.6 µs 29.6 µs
ANDNB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — —	— — —	— — —	— — —	2.7 µs 2.7 µs 29.6 µs 29.6 µs	2.7 µs 2.7 µs 29.6 µs 29.6 µs	2.7 µs 2.7 µs 29.6 µs 29.6 µs	2.7 µs 2.7 µs 29.6 µs 29.6 µs
OUTB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — —	— — —	— — —	— — —	3.1 µs 3.1 µs 30.3 µs 30.3 µs	3.4 µs 3.4 µs 30.7 µs 30.7 µs	3.1 µs 3.1 µs 30.3 µs 30.3 µs	3.4 µs 3.4 µs 30.7 µs 30.7 µs
SETB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — —	— — —	— — —	— — —	13.4 µs 13.4 µs 41.1 µs 41.1 µs	3.4 µs 3.4 µs 29.1 µs 29.1 µs	13.4 µs 13.4 µs 41.1 µs 41.1 µs	3.4 µs 3.4 µs 29.1 µs 29.1 µs
RSTB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — —	— — —	— — —	— — —	13.5 µs 13.5 µs 41.3 µs 41.3 µs	1.4 µs 1.4 µs 29.1 µs 29.1 µs	13.5 µs 13.5 µs 41.3 µs 41.3 µs	1.4 µs 1.4 µs 29.1 µs 29.1 µs

Команды немедленного действия

Команды немедленного действия		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
LDI	V	—	—	—	—	—	—	20.6 µs	1.1 µs
LDIF	1st#: X 2nd#: K:Constant	—	—	—	—	—	—	26.6µs+ 0.9µs x N	1.4 µs
STRI	X	27 µs	9.8 µs	29 µs	10.7 µs	19.3 µs	19.3 µs	19.3 µs	19.3 µs
STRNI	X	26 µs	8.6 µs	29 µs	10.7 µs	19.4 µs	19.4 µs	19.4 µs	19.4 µs
ORI	X	27 µs	9.8 µs	29 µs	8.4 µs	19.1 µs	18.7 µs	19.1 µs	18.7 µs
ORNI	X	26 µs	8.6 µs	29 µs	8.4 µs	19.2 µs	18.9 µs	19.2 µs	18.9 µs
ANDI	X	25 µs	8.0 µs	27 µs	8.4 µs	18.7 µs	18.7 µs	18.7 µs	18.7 µs
ANDNI	X	24 µs	6.8 µs	28 µs	8.4 µs	18.8 µs	18.8 µs	18.8 µs	18.8 µs
OROUTI	Y	45 µs	45 µs	39 µs	40 µs	27.5 µs	27.5 µs	27.5 µs	27.5 µs
OUTI	Y	45 µs	45 µs	39 µs	40 µs	25.5 µs	25.5 µs	25.5 µs	25.5 µs
OUTIF	1st#: Y 2nd#: K:Constant	—	—	—	—	—	—	66.1µs+ 0.9µs x N	1.4 µs
SETI	1st #: Y 2nd #: Y (N pt)	25.5 µs 5.5µs+20 xN	6.8 µs 6.8 µs	39.0 µs 44µs+25 xN	8.4 µs 8.4 µs	23.1 µs 22.8µs+ 1.4xN	0.9 µs 0.9 µs	23.1 µs 22.8µs+ 1.4xN	0.9 µs 0.9 µs
RSTI	1st #: Y 2nd #: Y (N pt)	25.5 µs 5µs+20.5 xN	6.8 µs 6.8 µs	37 µs 45µs+22 xN	8.4 µs 8.4 µs	23.2 µs 22.8µs+ 1.4xN	0.9 µs 0.9 µs	23.2 µs 22.8µs+ 1.4xN	0.9 µs 0.9 µs

Таймеры, счетчики и регистры сдвига

Таймеры, счетчики и регистры сдвига			DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных		Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
TMR	1st	2nd								
	T	V:Data Reg.	75 µs	31 µs	61 µs	23.5 µs	26.8 µs	7.3 µs	26.8 µs	7.3 µs
		V:Bit Reg.	158 µs	31 µs	158 µs	23.5 µs	26.8 µs	7.3 µs	26.8 µs	7.3 µs
		K:Constant	66 µs	31 µs	70 µs	23.5 µs	20.0 µs	4.8 µs	20.0 µs	4.8 µs
		P:Indir. (Data)	—	—	177 µs	131.0 µs	45.6 µs	30.2 µs	45.6 µs	30.2 µs
		P:Indir. (Bit)	—	—	271 µs	136.0 µs	45.6 µs	30.2 µs	45.6 µs	30.2 µs
TMRF	1st	2nd								
	T	V:Data Reg.	75 µs	31 µs	61 µs	23.5 µs	51.4 µs	7.3 µs	51.4 µs	7.3 µs
		V:Bit Reg.	158 µs	31 µs	158 µs	23.5 µs	51.4 µs	7.3 µs	51.4 µs	7.3 µs
		K:Constant	66 µs	31 µs	70 µs	23.5 µs	48.4 µs	4.6 µs	48.4 µs	4.6 µs
		P:Indir. (Data)	—	—	177 µs	131.0 µs	75.9 µs	30.2 µs	75.9 µs	30.2 µs
		P:Indir. (Bit)	—	—	271 µs	136.0 µs	75.9 µs	30.2 µs	75.9 µs	30.2 µs
TMRA	1st	2nd								
	T	V:Data Reg.	94 µs	56 µs	75 µs	41 µs	48.9 µs	7.3 µs	48.9 µs	7.3 µs
		V:Bit Reg.	304 µs	264 µs	253 µs	219 µs	48.9 µs	7.3 µs	48.9 µs	7.3 µs
		K:Constant	95 µs	45 µs	79 µs	49 µs	45.0 µs	4.6 µs	45.0 µs	4.6 µs
		P:Indir. (Data)	—	—	193 µs	159 µs	75.9 µs	30.2 µs	75.9 µs	30.2 µs
		P:Indir. (Bit)	—	—	366 µs	331 µs	75.9 µs	30.2 µs	75.9 µs	30.2 µs
TMRAF	1st	2nd								
	T	V:Data Reg.	98 µs	54 µs	75 µs	42 µs	54.2 µs	7.3 µs	54.2 µs	7.3 µs
		V:Bit Reg.	304 µs	264 µs	253 µs	218 µs	54.2 µs	7.3 µs	54.2 µs	7.3 µs
		K:Constant	95 µs	49 µs	80 µs	50 µs	50.3 µs	4.6 µs	50.3 µs	4.6 µs
		P:Indir. (Data)	—	—	193 µs	159 µs	81.2 µs	30.2 µs	81.2 µs	30.2 µs
		P:Indir. (Bit)	—	—	366 µs	331 µs	81.2 µs	30.2 µs	81.2 µs	30.2 µs
CNT	1st	2nd								
	CT	V:Data Reg.	68 µs	61 µs	59 µs	38 µs	25.8 µs	7.3 µs	25.8 µs	7.3 µs
		V:Bit Reg.	148 µs	141 µs	157 µs	133 µs	25.8 µs	7.3 µs	25.8 µs	7.3 µs
		K:Constant	56 µs	45 µs	59 µs	45 µs	22.2 µs	4.6 µs	22.2 µs	4.6 µs
		P:Indir. (Data)	—	—	176 µs	152 µs	53.5 µs	30.2 µs	53.5 µs	30.2 µs
		P:Indir. (Bit)	—	—	270 µs	245 µs	53.5 µs	30.2 µs	53.5 µs	30.2 µs
SGCNT	1st	2nd								
	CT	V:Data Reg.	57 µs	64 µs	58 µs	38 µs	27.3 µs	7.3 µs	27.3 µs	7.3 µs
		V:Bit Reg.	140 µs	148 µs	155 µs	133 µs	27.3 µs	7.3 µs	27.3 µs	7.3 µs
		K:Constant	46 µs	53 µs	67 µs	45 µs	23.5 µs	4.6 µs	23.5 µs	4.6 µs
		P:Indir. (Data)	—	—	175 µs	152 µs	54.9 µs	30.2 µs	54.9 µs	30.2 µs
		P:Indir. (Bit)	—	—	268 µs	245 µs	54.9 µs	30.2 µs	54.9 µs	30.2 µs
UDC	1st	2nd								
	CT	V:Data Reg.	103 µs	74 µs	80.0 µs	56 µs	39.8 µs	7.3 µs	39.8 µs	7.3 µs
		V:Bit Reg.	310 µs	281 µs	261 µs	224 µs	39.8 µs	7.3 µs	39.8 µs	7.3 µs
		K:Constant	102 µs	70 µs	97 µs	60 µs	35.4 µs	4.6 µs	35.4 µs	4.6 µs
		P:Indir. (Data)	—	—	202 µs	165 µs	67.8 µs	30.2 µs	67.8 µs	30.2 µs
		P:Indir. (Bit)	—	—	374 µs	336 µs	67.8 µs	30.2 µs	67.8 µs	30.2 µs
SR	C (N points to shift)		30µs+ 4.6µsxN	17.2 µs	25µs+ 4µsxN	19.7 µs	17.8µs+ 0.9µsxN	9.8 µs	17.8µs+ 0.9µsxN	9.8 µs

Команды загрузки аккумулятора/стека и вывода данных

Команды загрузки аккумулятора/стека и вывода данных		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
LD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	68 µs 149 µs 62 µs 169 µs 256 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs 8.4 µs	68 µs 143 µs 159 µs 238 µs 62 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs 8.4 µs	11.8 µs 11.8 µs 9.0 µs 33.9 µs 33.9 µs	1.0 µs 1.0 µs 1.0 µs 0.9 µs 0.9 µs	11.8 µs 11.8 µs 9.0 µs 33.9 µs 33.9 µs	1.0 µs 1.0 µs 1.0 µs 0.9 µs 0.9 µs
LDA	O: (Octal constant for address)	58 µs	8.4 µs	56 µs	8.4 µs	10.4 µs	1.0 µs	10.4 µs	1.0 µs
LDD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	72 µs 266 µs 64 µs 172 µs 373 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs 8.4 µs	67 µs 228 µs 69 µs 158 µs 323 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs 8.4 µs	12.2 µs 12.2 µs 9.0 µs 37.8 µs 37.8 µs	1.0 µs 1.0 µs 1.0 µs 0.9 µs 0.9 µs	12.2 µs 12.2 µs 9.0 µs 37.8 µs 37.8 µs	1.0 µs 1.0 µs 1.0 µs 0.9 µs 0.9 µs
LDF	1st 2nd X, Y, C, S K:Constant T, CT, SP	—	—	86µs+ 5µs x N	8.4 µs	20.5µs+ 0.9µs x N	0.9µs	20.5µs+ 0.9µs x N	0.9µs
LDR	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	—	—	—	—	29.5 µs 29.5 µs 25.5 µs 54.9 µs 54.9 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs	29.5 µs 29.5 µs 25.5 µs 54.9 µs 54.9 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs
LDSX	K: Constant	—	—	79µs	8.4 µs	14.6 µs	1.0 µs	14.6 µs	1.0 µs
LDX	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	—	—	—	—	10.8 µs 10.8 µs 45.2 µs 45.2 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs	10.8 µs 10.8 µs 45.2 µs 45.2 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs
OUT	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	60 µs 132 µs 162 µs 239 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs	21 µs 126 µs 112 µs 222 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs	9.3 µs 9.3 µs 35.2 µs 35.2 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs	9.3 µs 9.3 µs 35.2 µs 35.2 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs
OUTD	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	68 µs 276 µs 196 µs 384 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs	26 µs 235 µs 116 µs 331 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs	10.2 µs 10.2 µs 35.8 µs 35.8 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs	10.2 µs 10.2 µs 35.8 µs 35.8 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs

Команды загрузки аккумулятора/стека и вывода данных (продолжение)		**DL230**		**DL240**		**DL250-1**		**DL260**	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
OUTF	1st 2nd X, Y, C K:Constant	—	—	53µs+ 7µs x N	8.4 µs	54µs+ 1.0µs x N	0.9 µs	54µs+ 1.0µs x N	0.9 µs
OUTL	V:Data Reg. V:Bit Reg.	—	—	—	—	—	—	13.5 µs 13.5 µs	1.0 µs 1.0 µs
OUTM	V:Data Reg. V:Bit Reg.	—	—	—	—	—	—	13.7 µs 13.7 µs	1.0 µs 1.0 µs
OUTX	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	—	—	—	—	—	—	17.2 µs 17.2 µs 43.4 µs 43.4 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs
POP	None	55 µs	7.2 µs	50 µs	8.4 µs	8.4 µs	1.0µs	8.4 µs	1.0µs

Логические команды

Логические команды (аккумулятор)		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
AND	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	58 µs 261 µs — —	10.4 µs 10.4 µs — —	54 µs 145 µs 162 µs 241 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs	7.9 µs 7.9 µs 33.4 µs 33.4 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs	7.9 µs 7.9 µs 33.4 µs 33.4 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs
ANDD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — 53 µs — —	— — 8.4 µs — —	— — 60 µs — —	— — 8.4 µs — —	8.9 µs 8.9 µs 5.7 µs 34.4 µs 34.4 µs	1.0 µs 1.0 µs 1.0 µs 0.9 µs 0.9 µs	8.9 µs 8.9 µs 5.7 µs 34.4 µs 34.4 µs	1.0 µs 1.0 µs 1.0 µs 0.9 µs 0.9 µs
ANDF	1st: X, Y, C,S T,CT,SP GX,GY 2nd: K:Constant	—	—	—	—	21.6µs+ 0.9µs x N	1.0 µs	21.6µs+ 0.9µs x N	1.0 µs
ANDS	None	—	—	—	—	—	—	10.0 µs	1.0 µs
OR	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	59 µs 257 µs — —	10.4 µs 10.4 µs — —	54 µs 144 µs 160 µs 239 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs	8.1 µs 8.1 µs 33.8 µs 33.8 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs	8.1 µs 8.1 µs 33.8 µs 33.8 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs
ORD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — 49 µs — —	— — 8.4 µs — —	— — 60 µs — —	— — 8.4 µs — —	9.0 µs 9.0 µs 5.8 µs 34.5 µs 34.5 µs	1.0 µs 1.0 µs 1.0 µs 0.9 µs 0.9 µs	9.0 µs 9.0 µs 5.8 µs 34.5 µs 34.5 µs	1.0 µs 1.0 µs 1.0 µs 0.9 µs 0.9 µs
ORF	1st: X, Y, C,S T,CT,SP GX,GY 2nd: K:Constant	—	—	—	—	20.9µs+ 0.9µs x N	1.0 µs	20.9µs+ 0.9µs x N	1.0 µs
ORS	None	—	—	—	—	—	—	10.2 µs	1.0 µs
XOR	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	60 µs 257 µs — —	10.4 µs 10.4 µs — —	69 µs 144 µs 160 µs 239 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs	8.0 µs 8.0 µs 33.6 µs 33.6 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs	8.0 µs 8.0 µs 33.6 µs 33.6 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs
XORD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — 49 µs — —	— — 8.4 µs — —	— — 62 µs — —	— — 8.4 µs — —	9.0 µs 9.0 µs 5.4 µs 34.4 µs 34.4 µs	1.0 µs 1.0 µs 1.0 µs 0.9 µs 0.9 µs	9.0 µs 9.0 µs 5.4 µs 34.4 µs 34.4 µs	1.0 µs 1.0 µs 1.0 µs 0.9 µs 0.9 µs

Логические команды (аккумулятор)- продолжение		**DL230**		**DL240**		**DL250-1**		**DL260**	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
XORF	1st 2nd X, Y, C,S K:Constant T,CT,SP GX,GY	—	—	—	—	20.9µs+ 0.9µs x N	1.0 µs	20.9µs+ 0.9µs x N	1.0 µs
XORS	None	—	—	—	—	—	—	10.1 µs	1.0 µs
CMP	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	59 µs 259 µs — —	10.4 µs 10.4 µs — —	69 µs 115 µs 130 µs 211 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs	9.4 µs 9.4 µs 34.9 µs 34.9 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs	9.4 µs 9.4 µs 34.9 µs 34.9 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs
CMPD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	63 µs 257 µs 54 µs — —	8.4 µs 8.4 µs 8.4 µs — —	47 µs 206 µs 49 µs 133 µs 303 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs 8.4 µs	9.9 µs 9.9 µs 6.7 µs 35.4 µs 35.4 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs	9.9 µs 9.9 µs 6.7 µs 35.4 µs 35.4 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs
CMPF	1st 2nd X, Y, C,S K:Constant T,CT,SP GX,GY	—	—	—	—	29.2µs+ 1.0µs x N	1.0 µs	29.2µs+ 1.0µs x N	1.0 µs
CMPR	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	—	—	—	—	42.8 µs 42.8µs 38.4 µs 69.0 µs 69.0 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs	42.8 µs 42.8µs 38.4 µs 69.0 µs 69.0 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs
CMPS	None	—	—	—	—	—	—	11.2 µs	1.0 µs

Математические команды

Математические команды (аккумулятор)		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
ADD	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	198 µs 397 µs — —	10.6 µs 10.6 µs — —	291 µs 363 µs 441 µs 520 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs	78.4 µs 78.4 µs 101.2 µs 101.2 µs	0.9 µs 0.9 µs 0.9 µs 0.9 µs	78.4 µs 78.4 µs 101.2 µs 101.2 µs	0.9 µs 0.9 µs 0.9 µs 0.9 µs
ADDD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	198 µs 397 µs 188 µs — —	8.4 µs 8.4 µs 8.4 µs — —	291 µs 512 µs 298 µs 442 µs 608 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs 8.4 µs	83.3 µs 83.3 µs 67.7 µs 101.2 µs 101.2 µs	0.9 µs 0.9 µs 0.9 µs 0.9 µs 0.9 µs	83.3 µs 83.3 µs 67.7 µs 101.2 µs 101.2 µs	0.9 µs 0.9 µs 0.9 µs 0.9 µs 0.9 µs
SUB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	200 µs 397 µs — —	10.6 µs 10.6 µs — —	287 µs 360 µs 434 µs 513 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs	77.4 µs 77.4 µs 95.1 µs 95.1 µs	0.9 µs 0.9 µs 0.9 µs 0.9 µs	77.4 µs 77.4 µs 95.1 µs 95.1 µs	0.9 µs 0.9 µs 0.9 µs 0.9 µs
SUBD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	198 µs 392 µs 190 µs — —	8.4 µs 8.4 µs 8.4 µs — —	288 µs 504 µs 294 µs 434 µs 600 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs 8.4 µs	82.5 µs 82.5 µs 66.0 µs 99.7 µs 99.7 µs	0.9 µs 0.9 µs 0.9 µs 0.9 µs 0.9 µs	82.5 µs 82.5 µs 66.0 µs 99.7 µs 99.7 µs	0.9 µs 0.9 µs 0.9 µs 0.9 µs 0.9 µs
MUL	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	497 µs 483 µs 487 µs — —	10.6 µs 10.6 µs 8.4 µs — —	311 µs 385 µs 334 µs 401 µs 461 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs 8.4 µs	266.1 µs 266.1 µs 286.9 µs 290.0 µs 290.0 µs	0.9 µs 0.9 µs 0.9 µs 0.9 µs 0.9 µs	266.1 µs 266.1 µs 286.9 µs 290.0 µs 290.0 µs	0.9 µs 0.9 µs 0.9 µs 0.9 µs 0.9 µs
MULD	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — —	— — —	— — —	— — —	839.1 µs 839.1 µs 863.1 µs 863.1 µs	0.9 µs 0.9 µs 0.9 µs 0.9 µs	839.1 µs 839.1 µs 863.1 µs 863.1 µs	0.9 µs 0.9 µs 0.9 µs 0.9 µs
DIV	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	909 µs 1108 µs 699 µs — —	10.6 µs 10.6 µs 8.4 µs — —	601 µs 675 µs 573 µs 691 µs 771 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs 8.4 µs	363.9 µs 363.9 µs 384.4 µs 419.8 µs 419.8 µs	0.9 µs 0.9 µs 0.9 µs 0.9 µs 0.9 µs	363.9 µs 363.9 µs 384.4 µs 419.8 µs 419.8 µs	0.9 µs 0.9 µs 0.9 µs 0.9 µs 0.9 µs
DIVD	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — —	— — —	— — —	— — —	398.3 µs 398.3 µs 390.9 µs 390.9 µs	0.9 µs 0.9 µs 0.9 µs 0.9 µs	398.3 µs 398.3 µs 390.9 µs 390.9 µs	0.9 µs 0.9 µs 0.9 µs 0.9 µs
INC	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — —	— — —	— — —	— — —	48.5 µs 48.5 µs 74.7 µs 74.7 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs	48.5 µs 48.5 µs 74.7 µs 74.7 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs
DEC	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — —	— — —	— — —	— — —	47.5 µs 47.5 µs 71.5 µs 71.5 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs	47.5 µs 47.5 µs 71.5 µs 71.5 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs

Математические команды (продолжение)		**DL230**		**DL240**		**DL250-1**		**DL260**	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
INCB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	88 µs 349 µs — —	10.4 µs 10.4 µs — —	35 µs 211 µs 126 µs 307 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs	13.2 µs 13.2 µs 38.6 µs 38.6 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs	13.2 µs 13.2 µs 38.6 µs 38.6 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs
DECB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	82 µs 351 µs — —	10.4 µs 10.4 µs — —	33 µs 210 µs 123 µs 304 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs	13.2 µs 13.2 µs 38.0 µs 38.0 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs	13.2 µs 13.2 µs 38.0 µs 38.0 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs
ADDB	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — —	— — — —	— — — —	— — — —	24.9 µs 24.9 µs 23.5 µs 51.1 µs 51.1 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs	24.9 µs 24.9 µs 23.5 µs 51.1 µs 51.1 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs
ADDBD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — —	— — — —	— — — —	— — — —	— — — —	— — — —	24.4 µs 24.4 µs 20.7 µs 50.7 µs 50.7 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs
SUBB	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — —	— — — —	— — — —	— — — —	24.7 µs 24.7 µs 23.3 µs 50.6 µs 50.6 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs	24.7 µs 24.7 µs 23.3 µs 50.6 µs 50.6 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs
SUBBD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — —	— — — —	— — — —	— — — —	— — — —	— — — —	24.2 µs 24.2 µs 20.2 µs 50.2 µs 50.2 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs
MULB	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — —	— — — —	— — — —	— — — —	10.8 µs 10.8 µs 8.2 µs 37.1 µs 37.1 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs	10.8 µs 10.8 µs 8.2 µs 37.1 µs 37.1 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs
DIVB	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — —	— — — —	— — — —	— — — —	28.7 µs 28.7 µs 26.1 µs 54.9 µs 54.9 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs	28.7 µs 28.7 µs 26.1 µs 54.9 µs 54.9 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs
ADDR	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — —	— — — —	— — — —	— — — —	48.1 µs 48.1 µs 41.7 µs 74.3 µs 74.3 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs	48.1 µs 48.1 µs 41.7 µs 74.3 µs 74.3 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs
SUBR	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — —	— — — —	— — — —	— — — —	50.1 µs 50.1 µs 58.7 µs 76.3 µs 76.3 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs	50.1 µs 50.1 µs 58.7 µs 76.3 µs 76.3 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs

Математические команды (продолжение)		**DL230**		**DL240**		**DL250-1**		**DL260**	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
MULR	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — —	— — — —	— — — —	— — — —	54.2 µs 54.2 µs 42.7 µs 80.4 µs 80.4 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs	54.2 µs 54.2 µs 42.7 µs 80.4 µs 80.4 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs
DIVR	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — —	— — — —	— — — —	— — — —	50.1 µs 50.1 µs 58.7 µs 76.3 µs 76.3 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs	50.1 µs 50.1 µs 58.7 µs 76.3 µs 76.3 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs 1.0 µs
ADDF	1st: X, Y, C,S T,CT,SP GX,GY 2nd: K:Constant	—	—	—	—	—	—	109.3µs+ 0.9µs x N	1.0 µs
SUBF	1st: X, Y, C,S T,CT,SP GX,GY 2nd: K:Constant	—	—	—	—	—	—	107.3µs+ 0.9µs x N	1.0 µs
MULF	1st: X, Y, C,S T,CT,SP GX,GY 2nd: K:Constant	—	—	—	—	—	—	352.5µs+ 0.8µs x N	1.0 µs
DIVF	1st: X, Y, C,S T,CT,SP GX,GY 2nd: K:Constant	—	—	—	—	—	—	477.3µs+ 0.8µs x N	1.0 µs
ADDS	None	—	—	—	—	—	—	99.5 µs	1.0 µs
SUBS	None	—	—	—	—	—	—	97.5 µs	1.0 µs
MULS	None	—	—	—	—	—	—	342.5 µs	1.0 µs
DIVS	None	—	—	—	—	—	—	467.3 µs	1.0 µs
ADDBS	None	—	—	—	—	—	—	24.3 µs	1.0 µs
SUBBS	None	—	—	—	—	—	—	23.7 µs	1.0 µs
MULBS	None	—	—	—	—	—	—	11.7 µs	1.0 µs
DIVBS	None	—	—	—	—	—	—	29.7 µs	1.0 µs

Математические команды (продолжение)		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
SQRTR	None	—	—	—	—	—	—	87.9 µs	1.0 µs
SINR	None	—	—	—	—	—	—	226.8 µs	1.0 µs
COSR	None	—	—	—	—	—	—	213.1 µs	1.0 µs
TANR	None	—	—	—	—	—	—	285.5 µs	1.0 µs
ASINR	None	—	—	—	—	—	—	489.8 µs	1.0 µs
ACOSR	None	—	—	—	—	—	—	508.3 µs	1.0 µs
ATANR	None	—	—	—	—	—	—	317.1 µs	1.0 µs

Дифференциальные (импульсные) команды

Дифференциальные команды		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
PD	X, Y, C	13.5 µs	13.5 µs	15.9 µs	14.6 µs	14.4 µs	14.4 µs	14.4 µs	14.4 µs
STRPD	X, Y, C,S,T,CT	—	—	—	—	5.4 µs	5.4 µs	5.4 µs	5.4 µs
STRND	X, Y, C,S,T,CT	—	—	—	—	7.3 µs	7.3 µs	7.3 µs	7.3 µs
ORPD	X, Y, C,S,T,CT	—	—	—	—	6.8 µs	5.2 µs	6.8 µs	5.2 µs
ORND	X, Y, C,S,T,CT	—	—	—	—	7.1 µs	4.9 µs	7.1 µs	4.9 µs
ANDPD	X, Y, C,S,T,CT	—	—	—	—	6.8 µs	5.2 µs	6.8 µs	5.2 µs
ANDND	X, Y, C,S,T,CT	—	—	—	—	7.1 µs	4.9 µs	7.1 µs	4.9 µs

Битовые команды

Битовые команды (аккумулятор)		DL230		DL240		DL250-1		DL260	
Ко-ман-да	Разрешенные типы данных	Ис-пол-няе-мые	Неис-пол-няе-мые	Ис-пол-няе-мые	Неис-пол-няе-мые	Ис-пол-няе-мые	Неис-пол-няе-мые	Ис-пол-няе-мые	Неис-пол-няе-мые
SUM	None	—	—	—	—	—	—	6.7 µs	1.0 µs
SHFR	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	44µs+14 .6 x N 243µs+1 4.6 x N 34µs+14 .6 x N	10.4 µs 8.4 µs 8.4 µs	35µs+6 x N 110µs+6 x N 35µs+6 x N	8.4 µs 8.4 µs 8.4 µs	12.1µs+ 0.1 x N 8.4µs+ 0.1 x N	0.9 µs	12.1µs+ 0.1 x N 8.4µs+ 0.1 x N	0.9 µs
SHFL	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	44µs+14 .6 x N 243µs+1 4.6 x N 34µs+14 .6 x N	10.4 µs 8.4 µs 8.4 µs	33µs+6 x N 107µs+6 x N 33µs+6 x N	8.4 µs 8.4 µs 8.4 µs	12.1µs+ 0.1 x N 8.4µs+ 0.1 x N	0.9 µs	12.1µs+ 0.1 x N 8.4µs+ 0.1 x N	0.9 µs
ROTR	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	—	—	—	—	—	—	16.4 µs 16.4 µs 12.9 µs	1.0 µs 1.0 µs 1.0 µs
ROTL	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	—	—	—	—	—	—	16.4 µs 16.4 µs 12.7 µs	1.0 µs 1.0 µs 1.0 µs
ENCO	None	62 µs	7.2 µs	98 µs	8.4 µs	33.9 µs	0.9 µs	33.9 µs	0.9 µs
DECO	None	34 µs	7.2 µs	28 µs	8.4 µs	5.7 µs	1.0 µs	5.7 µs	1.0 µs

Команды преобразования чисел

Команды преобразования чисел (аккумулятор)		DL230		DL240		DL250-1		DL260	
Ко-манда	Разрешенные типы данных	Ис-пол-няе-мые	Неис-пол-няе-мые	Ис-пол-няе-мые	Неис-пол-няе-мые	Ис-пол-няе-мые	Неис пол-няе-мые	Ис-пол-няе-мые	Неис-пол-няе-мые
BIN	None	359 µs	7.2 µs	267 µs	8.4 µs	100.2 µs	0.9 µs	100.2 µs	0.9 µs
BCD	None	403 µs	7.2 µs	383 µs	8.4 µs	95.2 µs	0.9 µs	95.2 µs	0.9 µs
INV	None	27 µs	5.0 µs	12.0 µs	8.4 µs	2.5 µs	1.0 µs	2.5 µs	1.0 µs
BCDCPL	None	296 µs	7.2 µs	69 µs	8.4 µs	75.6 µs	1.0 µs	75.6 µs	1.0 µs
ATH	V	—	—	—	—	—	—	25.4 µs	1.0 µs
HTA	V	—	—	—	—	—	—	25.4 µs	1.0 µs
GRAY	None	—	—	227 µs	9.0 µs	110.8 µs	1.0 µs	110.8 µs	1.0 µs
SFLDGT	None	—	—	258 µs	9.0 µs	23.1 µs	1.0 µs	23.1 µs	1.0 µs
BTOR	None	—	—	—	—	18.6 µs	1.0 µs	18.6 µs	1.0 µs
RTOB	None	—	—	—	—	8.6 µs	1.0 µs	8.6 µs	1.0 µs
RADR	None	—	—	—	—	—	—	51.4 µs	1.0 µs
DEGR	None	—	—	—	—	—	—	81.5 µs	1.0 µs

Команды работы с таблицами

Команды работы с таблицами		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
FILL	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— —	— —	— —	— —	— —	— —	29.4µs+ 8.0µs x N 26.2µs+ 8.0µs x N 55.1µs+ 8.0µs x N	1.0 µs 1.0 µs 1.0 µs
FIND	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	—	—	—	—	—	—	66.8 µs 66.8 µs 64.0 µs	1.0 µs 1.0 µs 1.0 µs
FDGT	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	—	—	—	—	—	—	66.1 µs 66.1 µs 55.2 µs	1.0 µs 1.0 µs 1.0 µs
FINDB	V:Data Reg. (N bits) V:Bit Reg. (N bits) P:Indir. (Data) P:Indir. (Bit)	—	—	—	—	—	—	210.8 µs 210.8 µs 237.0 µs 237.0 µs	1.0 µs 1.0 µs 1.0 µs 1.0 µs
MOV	Move V:data reg. to V:data reg. Move V:bit reg. to V:data reg. Move V:data reg to V:bit reg. Move V:bit reg. to V:bit reg. N= #of words	450µs+ 17 x N 430µs+ 244 x N 460µs+ 215 x N 490µs+ 448 x N	6.2µs 6.2µs 6.2µs 6.2µs	586µs+ 8 x N 629µs+ 114.7 xN 569µs+ 94.4 x N 639µs+ 198 x N	8.4µs 8.4µs 8.4µs 8.4µs	60.2µs+ 9.5xN	0.9 µs	60.2µs+ 9.5xN	0.9 µs
TTD	V:Data Reg. V:Bit Reg	—	—	—	—	—	—	66.9 µs 66.9 µs	1.0 µs 1.0 µs
RFB	V:Data Reg. V:Bit Reg	—	—	—	—	—	—	66.8 µs 66.8 µs	1.0 µs 1.0 µs
STT	V:Data Reg. V:Bit Reg K:Constant	—	—	—	—	—	—	67.8 µs 67.8 µs 65.0 µs	1.0 µs 1.0 µs 1.0 µs
RFT	V:Data Reg. V:Bit Reg	—	—	—	—	—	—	51.1 µs 51.1 µs	1.0 µs 1.0 µs
ATT	V:Data Reg. V:Bit Reg K:Constant	—	—	—	—	—	—	53.5 µs 53.5 µs 50.8 µs	1.0 µs 1.0 µs 1.0 µs
TSHFL	V:Data Reg. V:Bit Reg	—	—	—	—	—	—	134.0 µs 134.0 µs	1.0 µs 1.0 µs
TSHFR	V:Data Reg. V:Bit Reg	—	—	—	—	—	—	133.9 µs 133.9 µs	1.0 µs 1.0 µs

Команды работы с таблицами (продолжение)		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
ANDMOV	V:Data Reg. V:Bit Reg	— —	— —	— —	— —	— —	— —	80.2 µs 80.2 µs	1.0 µs 1.0 µs
ORMOV	V:Data Reg. V:Bit Reg	— —	— —	— —	— —	— —	— —	80.4 µs 80.4 µs	1.0 µs 1.0 µs
XORMOV	V:Data Reg. V:Bit Reg	— —	— —	— —	— —	— —	— —	80.4 µs 80.4 µs	1.0 µs 1.0 µs
SWAP	V:Data Reg. V:Bit Reg	— —	— —	— —	— —	— —	— —	84.1 µs 84.1 µs	1.0 µs 1.0 µs
SETBIT	V:Data Reg. (N bits) V:Bit Reg. (N bits)	— —	— —	— —	— —	— —	— —	59.5 µs 59.5 µs	1.0 µs 1.0 µs
RSTBIT	V:Data Reg. (N bits) V:Bit Reg. (N bits)	— —	— —	— —	— —	— —	— —	59.5 µs 59.5 µs	1.0 µs 1.0 µs
MOVMC	Move V:Data Reg. to E^2 Move V:Bit Reg. to E^2 Move from E^2 to V:Data Reg. Move from E^2 to V:Bit Reg. N= #of words	— — 250µs+ 201xN —	— — 6.2µs —	— — 392µs+ 7843xN 520µs+ 181 x N 565µs+ 344 x N	8.4µs 8.4µs 8.4µs 8.4µs	33.5µs+ 10.4xN	0.9µs	33.5µs+ 10.4xN	0.9µs
LDLBL	K	58µs	8.4µs	56 µs	8.4µs	6.4µs	1.3 µs	6.4µs	1.3µs

Команды управления процессором

Команды управления процессором		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
NOP	None	0 µs	0 µs	0 µs	0 µs	0.5 µs	0.5 µs	0.5 µs	0.5 µs
END	None	27 µs	27 µs	16 µs	16 µs	12.8 µs	0 µs	12.8 µs	0 µs
STOP	None	16 µs	5 µs	15 µs	7.4 µs	0 µs	0.9 µs	0 µs	0.9 µs
RSTWT	None	—	—	19 µs	8.4 µs	4.7 µs	0.9 µs	4.7 µs	0.9 µs

Команды управления программой

Команды управления программой		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
GOTO	K	—	—	14 µs	8.4 µs	5.1 µs	4.8 µs	5.1 µs	4.8 µs
LBL	K	—	—	0.6 µs	0.6 µs	5.7 µs	0.0 µs	5.7 µs	0.0 µs
FOR	V, K	—	—	32 µs	16.4 µs	85.8 µs	5.8 µs	85.8 µs	5.8 µs
NEXT	None	—	—	19 µs	0 µs	10.2 µs	0.0 µs	10.2 µs	0.0 µs
GTS	K	—	—	37 µs	11.4 µs	10.9 µs	5.5 µs	10.9 µs	5.5 µs
SBR	K	—	—	0.6 µs	0 µs	0.5 µs	0.0 µs	0.5 µs	0.0 µs
RT	None	—	—	35 µs	0 µs	9.9 µs	0.0 µs	9.9 µs	0.0 µs
RTC	None	—	—	—	—	—	—	11.4 µs	5.9 µs
MLS	K (1–7)	12 µs	12 µs	11.5 µs	11.5 µs	3.7 µs	3.7 µs	3.7 µs	3.7 µs
MLR	K (0–7) N= 1 to 7	13 µs + 2.4 x N	13 µs + 2.4 x N	12.7µs + 2.3 xN	12.7µs + 2.3 xN	3.5 µs	3.5 µs	3.5 µs	3.5 µs

Команды прерывания

Команды прерывания		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
ENI	None	9 µs	5 µs	10.5 µs	8.4 µs	5.0 µs	1.0 µs	5.0 µs	1.0 µs
DISI	None	8 µs	5 µs	11 µs	8.4 µs	5.7 µs	0.9 µs	5.7 µs	0.9 µs
INT	0 (0–7)	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs
IRT	None	1.6 µs	0 µs	8 µs	0 µs	1.3 µs	0 µs	1.3 µs	0 µs
IRTC	None	—	—	—	—	—	—	0.5 µs	0 µs

Сетевые команды

Сетевые команды		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
RX	X, Y, C, T, CT, SP, S V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	—	—	TBD	TBD	251.3 µs 251.3 µs 251.3 µs 270.3 µs 270.3 µs	1.1 µs 1.1 µs 1.1 µs 1.9 µs 1.9 µs	251.3 µs 251.3 µs 251.3 µs 270.3 µs 270.3 µs	1.1 µs 1.1 µs 1.1 µs 1.9 µs 1.9 µs
WX	X, Y, C, T, CT, SP, S V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	—	—	TBD	TBD	252.0 µs 252.0 µs 252.0 µs 271.3 µs 271.3 µs	2.7 µs 2.7 µs 2.7 µs 3.4 µs 3.4 µs	252.0 µs 252.0 µs 252.0 µs 271.3 µs 271.3 µs	2.7 µs 2.7 µs 2.7 µs 3.4 µs 3.4 µs

Команды интеллектуального ввода/вывода

Команды интеллектуального ввода/вывода		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
RD	V:Data Reg. V:Bit Reg.	TBD	TBD	TBD	TBD	385.7 µs 385.7 µs	1.2 µs 1.2 µs	385.7 µs 385.7 µs	1.2 µs 1.2 µs
WT	V:Data Reg. V:Bit Reg.	TBD	TBD	TBD	TBD	385.6 µs 385.6 µs	1.2 µs 1.2 µs	385.6 µs 385.6 µs	1.2 µs 1.2 µs

Команды вывода сообщений

Команды вывода сообщений		DL230		DL240		DL250-1		DL260	
Ко-ман-да	Разрешенные типы данных	Испол-няе-мые	Неис пол-няе-мые	Испол-няе-мые	Неис-пол-няе-мые	Испол-няе-мые	Неис пол-няе-мые	Испол-няе-мые	Неис-пол-няе-мые
FAULT	V:Data Reg. V:Bit Reg. K:Constant	171 µs 253 µs 2798 µs	8.4 µs 8.4 µs 8.4 µs	23176 µs 23206 µs 29108 µs	8.4 µs 8.4 µs 8.4 µs	84.9 µs 84.9 µs 80.8 µs	1.1 µs 1.1 µs 1.2 µs	84.9 µs 84.9 µs 80.8 µs	1.1 µs 1.1 µs 1.2 µs
DLBL	K	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs
NCON	K	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs
ACON	K	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs
PRINT	Text Data	—	—	—	—	36.3 µs	1.1 µs	36.3 µs	1.1 µs

Команды RLL PLUS

Команды RLL PLUS		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
ISG	S	31 µs	32 µs	28 µs	27 µs	20.9 µs	9.2 µs	20.9 µs	9.2 µs
SG	S	31 µs	32 µs	28 µs	27 µs	20.9 µs	9.2 µs	20.9 µs	9.2 µs
JMP	S	14 µs	8 µs	14.3 µs	8.4 µs	20.9 µs	3.7 µs	20.9 µs	3.7 µs
NJMP	S	14 µs	8 µs	13.3 µs	8.4 µs	21.0 µs	4.0 µs	21.0 µs	4.0 µs
CV	S	43 µs	27 µs	20 µs	20 µs	12.1 µs	12.1 µs	12.1 µs	12.1 µs
CVJMP	S (N stages, 1 to 16)	33µs +14.5µs xN	23 µs	22.9µs + 6.1 xN	10 µs	11.0 µs	11.0 µs	11.0 µs	11.0 µs
BCALL	C	18 µs	17 µs	17 µs	18 µs	22.1 µs	22.6 µs	22.1 µs	22.6 µs
BLK	C	32 µs	30 µs	17 µs	13 µs	17.1 µs	14.6 µs	17.1 µs	14.6 µs
BEND	None	17 µs	17 µs	9 µs	9 µs	8.7 µs	0.0 µs	8.7 µs	0.0 µs

Команды барабанного командоаппарата

Команды барабанного командоаппарата		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
DRUM	CT	—	—	—	—	265.2 µs	48.8 µs	265.2 µs	48.8 µs
EDRUM	CT	—	—	—	—	189.5 µs	78.0 µs	189.5 µs	78.0 µs
MDRMD	CT	—	—	—	—	411.3 µs	216.4 µs	411.3 µs	216.4 µs
MDRMW	CT	—	—	—	—	378.6 µs	147.0 µs	378.6 µs	147.0 µs

Команды даты / времени

Команды даты / времени		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
DATE	V:Data Reg. V:Bit Reg.	—	—	—	—	24.0 µs	1.2 µs	24.0 µs	1.2 µs
TIME	V:Data Reg. V:Bit Reg.	—	—	—	—	50.8 µs	1.2 µs	50.8 µs	1.2 µs

Команды MODBUS

Команды MODBUS		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
MRX	Input, Input Register Coil, Holding Register	—	—	—	—	—	—	120.2 µs	1.3 µs
MWX	Input, Input Register Coil, Holding Register	—	—	—	—	—	—	21.3 µs	1.3 µs

Команды ASCII

Команды ASCII		DL230		DL240		DL250-1		DL260	
Команда	Разрешенные типы данных	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые	Исполняемые	Неисполняемые
AIN	V	—	—	—	—	—	—	13.9 µs	12.0 µs
AFIND	V	—	—	—	—	—	—	111.5 µs	1.3 µs
AEX	V	—	—	—	—	—	—	111.7 µs	1.3 µs
CMPV	V	—	—	—	—	—	—	12.2 µs	1.3 µs
SWAPB	V	—	—	—	—	—	—	109.8 µs	1.3 µs
VPRINT	Text Data	—	—	—	—	—	—	161.6 µs	1.3 µs
PRINTV	V	—	—	—	—	—	—	163.3 µs	1.3 µs
ACRB	V	—	—	—	—	—	—	3.9 µs	1.1 µs

Приложение D. ***Специальные реле***

В этом приложении…

- Специальные реле процессорного модуля DL230
- Специальные реле процессорных модулей DL240/ DL250-1/DL260

«Специальные реле» — это контакты, которые используются операционной системой процессора (ЦПУ) для сообщения, что произошло определенное системное событие. Эти контакты можно использовать в своей релейной программе.

Специальные реле процессорного модуля DL230

Реле запуска и реального времени

SP0	Первый цикл сканирования	Включается только в первом цикле сканирования после включении питания или при переходе из Программного режима в Рабочий. Реле отключается на втором цикле сканирования. Полезно для функций, выполняемых только при запуске программы.
SP1	Всегда ВКЛЮЧЕНО	Обеспечивает контакт, гарантирующий выполнение команды в каждом цикле сканирования.
SP2	Всегда ВЫКЛЮЧЕНО	Обеспечивает контакт, который всегда выключен.
SP3	1 мин по таймеру	Включено в течение 30 сек и отключено в течение 30 сек.
SP4	1 сек. по таймеру	Включено в течение 0,5 сек и отключено в течение 0,5 сек.
SP5	100 мс по таймеру	Включено в течение 50 мс и отключено в течение 50 мс.
SP6	50 мс по таймеру	Включено в течение 25 мс и отключено в течение 25 мс.
SP7	Чередующиеся циклы сканирования	Включается через один цикл сканирование.

Реле состояния процессора

SP12	Терминальный Рабочий режим	Включено, когда процессорный модуль находится в Рабочем режиме.
SP16	Терминальный Программный режим	Включено, когда процессорный модуль находится в Программном режиме.
SP20	Реле вынужденного режима останова	Включено, когда выполнена команда STOP.
SP22	Разрешение прерывания	Включено, когда прерывание разрешено с использованием команды ENI.

Реле контроля работы системы

SP40	Критическая ошибка	Включено, когда возникает критическая ошибка, например, потеря связи с точками ввода/вывода.
SP41	Предупреждение	Включено, когда возникает некритическая ошибка, например, низкое напряжение батареи.
SP43	Низкое напряжение батареи	Включено, когда напряжение батареи низкое.
SP44	Ошибка программной памяти	Включено, когда возникает ошибка памяти, например, ошибка четности памяти.
SP45	Ошибка ввда/вывода	Включено, когда возникает ошибка ввода/вывода. Например, модуль ввода/вывода вынут из каркаса или обнаружена ошибка по шине вода/вывода.
SP47	Ошибка в конфигурации ввода/вывода	Включено, если обнаружена ошибка в конфигурации ввода/вывода. Это реле начинает работать после включения питания процессорного модуля в том случае, когда включена проверка конфигурации ввода/вывода
SP50	Неправильная команда	Включается при выполнении неправильной команды.
SP51	Истечение времени сторожевого таймера	Включено, когда время сторожевого таймера процессорного модуля истекло.
SP52	Грамматическая ошибка	Включено, когда обнаружена грамматическая ошибка либо при работе процессорного модуля, либо при проверке синтаксиса. В V7755 хранится точный код ошибки.
SP53	Логическая ошибка	Включено, когда процессорный модуль не может разрешить логику.

Реле состояния аккумулятора

SP60	Значение меньше, чем	Включено, когда значение в аккумуляторе меньше значения в команде.
SP61	Значение равно	Включено, когда значение в аккумуляторе равно значению в команде.
SP62	Больше, чем	Включено, когда значение в аккумуляторе больше значения в команде.
SP63	Нуль	Включено, когда результат выполнения команды равен нулю (в аккумуляторе).
SP64	Заимствование половинной длины	Включено, когда команда 16-битового вычитания приводит к заимствованию.
SP65	Заимствование	Включено, когда команда 32-битового вычитания приводит к заимствованию.
SP66	Перенос половинной длины	Включено, когда команда 16-битового сложения приводит к переносу.
SP67	Перенос	Включено, когда команда 32-битового сложения приводит к переносу.
SP70	Знак	Включается всякий раз, когда значение в Аккумуляторе становится отрицательным.
SP71	Неправильное восьмеричное число	Включается, когда вводится неправильное восьмеричное число. Это происходит также тогда, когда V-память неправильно определена указателем (P).
SP73	Переполнение	Включается, если происходит переполнение аккумулятора, когда сложение со знаком или вычитание приводит к неправильному биту знака.
SP75	Ошибка в данных	Включается, если число должно быть в формате BCD, а оно встретилось не в формате BCD.
SP76	Загрузка нуля	Включается, когда любая команда загружает в аккумулятор значение нуль.

Реле входа высокоскоростного ввода/ вывода

SP100	Состояние X0=ON	Включено, когда X0 включен.

Реле равенства высокоскоростного ввода/вывода в режиме 10 Счетчика 1
(используется с модулем D2-CTRINT)

SP540	Включается, когда текущее значение Счетчика равно значению в V3630.
SP541	Включается, когда текущее значение Счетчика равно значению в V3632.
SP542	Включается, когда текущее значение Счетчика равно значению в V3634
SP543	Включается, когда текущее значение Счетчика равно значению в V3636.
SP544	Включается, когда текущее значение Счетчика равно значению в V3640.
SP545	Включается, когда текущее значение Счетчика равно значению в V3642.
SP546	Включается, когда текущее значение Счетчика равно значению в V3644.
SP547	Включается, когда текущее значение Счетчика равно значению в V3646.
SP550	Включается, когда текущее значение Счетчика равно значению в V3650.
SP551	Включается, когда текущее значение Счетчика равно значению в V3652.
SP552	Включается, когда текущее значение Счетчика равно значению в V3654.
SP553	Включается, когда текущее значение Счетчика равно значению в V3656.
SP554	Включается, когда текущее значение Счетчика равно значению в V3660.
SP555	Включается, когда текущее значение Счетчика равно значению в V3662.
SP556	Включается, когда текущее значение Счетчика равно значению в V3664.
SP557	Включается, когда текущее значение Счетчика равно значению в V3666.
SP560	Включается, когда текущее значение Счетчика равно значению в V3670.
SP561	Включается, когда текущее значение Счетчика равно значению в V3672.
SP562	Включается, когда текущее значение Счетчика равно значению в V3674.
SP563	Включается, когда текущее значение Счетчика равно значению в V3676.
SP564	Включается, когда текущее значение Счетчика равно значению в V3700.
SP565	Включается, когда текущее значение Счетчика равно значению в V3702.
SP566	Включается, когда текущее значение Счетчика равно значению в V3704.
SP567	Включается, когда текущее значение Счетчика равно значению в V3706.

Специальные реле процессорных модулей DL240/DL250-1/DL260

Реле запуска и реального времени

SPO	Первый цикл сканирования	Включается только в первом цикле сканирования после включении питания или при переходе из программного режима в рабочий. Реле отключается на втором цикле сканирования. Полезно для функций, выполняемых только при запуске программы.
SP1	Всегда ВКЛЮЧЕНО	Обеспечивает контакт, гарантирующий выполнение команды в каждом цикле сканирования.
SP2	Всегда ВЫКЛЮЧЕНО	Обеспечивает контакт, который всегда отключен
SP3	1 мин по таймеру	Включено в течение 30 сек и отключено в течение 30 сек.
SP4	1 сек. по таймеру	Включено в течение 0,5 сек и отключено в течение 0,5 сек.
SP5	100 мс по таймеру	Включено в течение 50 мс и отключено в течение 50 мс.
SP6	50 мс по таймеру	Включено в течение 25 мс и отключено в течение 25 мс.
SP7	Чередующиеся циклы сканирования	Включается через один цикл сканирования.

Реле состояния процессора

SP11	Принудительный Рабочий режим	Включено всегда, когда переключатель процессорного модуля находится в положении RUN.
SP12	Терминальный Рабочий режим	Включено, когда переключатель процессорного модуля находится в положении TERM, а процессорный модуль находится в Рабочем (RUN) режиме.
SP13	Режим работы Тестирования	Включено, когда переключатель процессорного модуля находится в положении TERM, а процессорный модуль находится в режиме тестирования (Test RUN).
SP14	Реле размыкания 1 (только DL250-1/260)	Включено, когда выполняется команды BREAK. Выключено, когда процессорный модуль находится в любом другом режиме.
SP15	Режим тестирования программы	Включено, когда переключатель процессорного модуля находится в положении TERM, а процессорный модуль находится в РЕЖИМЕ ТЕСТИРОВАНИЯ ПРОГРАММЫ (TEST PROGRAM MODE).
SP16	Терминальный Программный режим	Включено, когда переключатель процессорного модуля находится в положении TERM, а процессор находится в ПРОГРАММНОМ РЕЖИМЕ (PROGRAM MODE).
SP17	Реле вынужденного режима останова (только	Включено всегда, когда переключатель процессорного модуля находится в положении STOP.
SP20	Режим вынужденного останова	Включено, когда выполнена команда STOP.
SP21	Реле размыкания 2 (только DL250-1/260)	Включено, когда выполняется команды BREAK. Выключено, когда режим процессора изменяется на РАБОЧИЙ (RUN).
SP22	Прерывание разрешено	Включено, когда прерывание разрешено с использованием команды ENI.
SP25	Блокирующее реле батареи процессорного модуля (DL250-1/260)	Включается, когда батарея процессорного модуля блокируются специальной V-памятью.

Реле контроля работы системы

SP40	Критическая ошибка	Включено, когда возникает критическая ошибка, например, потеря связи с точками ввода/вывода.
SP41	Предупреждение	Включено, когда возникает некритическая ошибка, например, низкое напряжение батареи.
SP43	Низкое напряжение батареи	Включено, когда напряжение батареи низкое или отсутствует. Замечание: Батарея должна быть установлена в процессорный модуль.
SP44	Ошибка программной памяти	Включено, когда возникает ошибка памяти, например, ошибка четности памяти.
SP45	Ошибка ввода/вывода	Включено, когда возникает ошибка ввода/вывода. Например, модуль ввода/вывода вынут из каркаса или обнаружена ошибка по шине каркаса ввода/вывода.
SP46	Ошибка связи	Включено, когда возникает ошибка связи в любом порту процессорного модуля.
SP47	Ошибка в конфигурации ввода/вывода	Включено, если обнаружена ошибка в конфигурации ввода/вывода. Это реле начинает работать после включения питания процессорного модуля в том случае, когда включена проверка конфигурации ввода/вывода.
SP50	Неправильная команда	Включается при выполнении неправильной команды.
SP51	Истечение времени сторожевого таймера	Включено, когда время сторожевого таймера процессорного модуля истекло.
SP52	Грамматическая ошибка	Включено, когда обнаружена грамматическая ошибка либо при работе процессорного модуля, либо при проверке синтаксиса. В V7755 хранится точный код ошибки.
SP53	Логическая ошибка	Включено, когда процессор не может разрешить логику.
SP54	Ошибка интеллектуального ввода/вывода	Включено, когда устанавливается связь с интеллектуальным модулем.

Реле состояния аккумулятора

SP60	Значение меньше, чем	Включено, когда значение в аккумуляторе меньше значения в команде.
SP61	Значение равно	Включено, когда значение в аккумуляторе равно значению в команде.
SP62	Больше, чем	Включено, когда значение в аккумуляторе больше значения в команде.
SP63	Нуль	Включено, когда результат выполнения команды равно нулю (в аккумуляторе).
SP64	Заимствование половинной длины	Включено, когда команда 16-битового вычитания приводит к заимствованию.
SP65	Заимствование	Включено, когда команда 32-битового вычитания приводит к заимствованию.
SP66	Перенос половины	Включено, когда команда 16-битового сложения приводит к переносу.
SP67	Перенос	Включено, когда команда 32-битового сложения приводит к переносу.
SP70	Знак	Включается, когда значение в аккумуляторе становится отрицательным.
SP71	Неправильное восьмеричное число	Включается, когда вводится неправильное восьмеричное число. Это происходит также, когда V-память неправильно определена указателем (P).
SP72	Число с плавающей запятой	Включается всякий раз, когда в аккумуляторе неправильное число с плавающей запятой.
SP73	Переполнение	Включается, если происходит переполнение аккумулятора, когда сложение со знаком или вычитание приводит к неправильному биту знака.
SP74	Антипереполнение	Включается, когда операция с плавающей запятой приводит к ошибке обнуления.
SP75	Ошибка в данных	Включается, если число должно быть в формате BCD, а оно встретилось не в формате BCD.
SP76	Загрузка нуля	Включается, когда любая команда загружает в аккумулятор значение нуль.

Реле модуля H2-CTRIO

SP100	X0 включен	X0 - включается, когда соответствующий вход включен.
SP101	X1 включен	X1- включается, когда соответствующий вход включен.
SP102	X2 включен	X2 - включается, когда соответствующий вход включен.
SP103	X3 включен	X3 - включается, когда соответствующий вход включен.

Реле контроля связи

SP116	Процессор DL240 занят	Включено, когда процессорный модуль связан с другим устройством.
SP116	Порт 2 процессора занят (DL250-1/260)	Включено, когда Порт 2 связан с другим устройством.
SP117	Ошибка связи Порта 2 (DL250-1/260)	Включается, когда в Порту 2 обнаружена ошибка связи.
SP120	Модуль занят, слот 0	Включается, когда коммуникационный модуль в слоте 0 занят приемом или передачей данных. Вы должны использовать это реле с командами RX или WX, чтобы исключить выполнение этих команд, когда модуль занят.
SP121	Ошибка соединения, слот 0	Включается, когда в коммуникационном модуле в слоте 0 локального каркаса обнаружена ошибка связи.
SP122	Модуль занят, слот 1	Включается, когда коммуникационный модуль в слоте 1 локального каркаса занят приемом или передачей данных. Вы должны использовать это реле с командами RX или WX, чтобы исключить выполнение этих команд, пока модуль занят.
SP123	Ошибка соединения, слот 1	Включается, когда в коммуникационном модуле в слоте 1 локального каркаса обнаружена ошибка связи.
SP124	Модуль занят, слот 2	Включается, когда коммуникационный модуль в слоте 2 локального каркаса занят приемом или передачей данных. Вы должны использовать это реле с командами RX или WX, чтобы исключить выполнение этих команд, пока модуль занят.
SP125	Ошибка соединения, слот 2	Включается, когда в коммуникационном модуле в слоте 2 локального каркаса обнаружена ошибка связи.
SP126	Модуль занят, слот3	Включается, когда коммуникационный модуль в слоте 3 локального каркаса занят приемом или передачей данных. Используйте это реле с командами RX или WX, чтобы исключить выполнение этих команд, пока модуль занят.
SP127	Ошибка соединения, слот3	Включается, когда в коммуникационном модуле в слоте 3 локального каркаса обнаружена ошибка связи.
SP130	Модуль занят, слот 4	Включается, когда коммуникационный модуль в слоте 4 локального каркаса занят приемом или передачей данных. Используйте это реле с командами RX или WX, чтобы исключить выполнение этих команд, пока модуль занят.
SP131	Ошибка соединения, слот 4	Включается, когда в коммуникационном модуле в слоте 4 локального каркаса обнаружена ошибка связи.
SP132	Модуль занят, слот 5	Включается, когда коммуникационный модуль в слоте 5 локального каркаса занят приемом или передачей данных. Используйте это реле с командами RX или WX, чтобы исключить выполнение этих команд, пока модуль занят.
SP133	Ошибка соединения, слот 5	Включается, когда в коммуникационном модуле в слоте 5 локального каркаса обнаружена ошибка связи.
SP134	Модуль занят, слот 6	Включается, когда коммуникационный модуль в слоте 6 локального каркаса занят приемом или передачей данных. Используйте это реле с командами RX или WX, чтобы исключить выполнение этих команд, пока модуль занят.
SP135	Ошибка соединения, Слот 6	Включается, когда в коммуникационном модуле в слоте 6 локального каркаса обнаружена ошибка связи.
SP136	Модуль занят, слот 7	Включается, когда коммуникационный модуль в слоте 7 локального каркаса занят приемом или передачей данных. Используйте это реле с командами RX или WX, чтобы исключить выполнение этих команд, пока модуль занят.
SP137	Ошибка соединения, слот 7	Включается, когда в коммуникационном модуле в слоте 7 локального каркаса обнаружена ошибка связи.

Реле равенства уставкам высокоскоростного счетчика 1 (для работы с H2-CTRIO)

SP540	Включается, когда текущее значение Счетчика равно значению в V3630.
SP541	Включается, когда текущее значение Счетчика равно значению в V3632.
SP542	Включается, когда текущее значение Счетчика равно значению в V3634
SP543	Включается, когда текущее значение Счетчика равно значению в V3636.
SP544	Включается, когда текущее значение Счетчика равно значению в V3640.
SP545	Включается, когда текущее значение Счетчика равно значению в V3642.
SP546	Включается, когда текущее значение Счетчика равно значению в V3644.
SP547	Включается, когда текущее значение Счетчика равно значению в V3646.
SP550	Включается, когда текущее значение Счетчика равно значению в V3650.
SP551	Включается, когда текущее значение Счетчика равно значению в V3652.
SP552	Включается, когда текущее значение Счетчика равно значению в V3654.
SP553	Включается, когда текущее значение Счетчика равно значению в V3656.
SP554	Включается, когда текущее значение Счетчика равно значению в V3660.
SP555	Включается, когда текущее значение Счетчика равно значению в V3662.
SP556	Включается, когда текущее значение Счетчика равно значению в V3664.
SP557	Включается, когда текущее значение Счетчика равно значению в V3666.
SP560	Включается, когда текущее значение Счетчика равно значению в V3670.
SP561	Включается, когда текущее значение Счетчика равно значению в V3672.
SP562	Включается, когда текущее значение Счетчика равно значению в V3674.
SP563	Включается, когда текущее значение Счетчика равно значению в V3676.
SP564	Включается, когда текущее значение Счетчика равно значению в V3700.
SP565	Включается, когда текущее значение Счетчика равно значению в V3702.
SP566	Включается, когда текущее значение Счетчика равно значению в V3704.
SP567	Включается, когда текущее значение Счетчика равно значению в V3706.

Реле равенства уставкам высокоскоростного счетчика 2 (для работы с H2-CTRIO)

SP570	Включается, когда текущее значение Счетчика равно значению в V3710.
SP571	Включается, когда текущее значение Счетчика равно значению в V3712.
SP572	Включается, когда текущее значение Счетчика равно значению в V3714.
SP573	Включается, когда текущее значение Счетчика равно значению в V3716.
SP574	Включается, когда текущее значение Счетчика равно значению в V3720.
SP575	Включается, когда текущее значение Счетчика равно значению в V3722.
SP576	Включается, когда текущее значение Счетчика равно значению в V3724.
SP577	Включается, когда текущее значение Счетчика равно значению в V3726.
SP600	Включается, когда текущее значение Счетчика равно значению в V3730.
SP601	Включается, когда текущее значение Счетчика равно значению в V3732.
SP602	Включается, когда текущее значение Счетчика равно значению в V3734.
SP603	Включается, когда текущее значение Счетчика равно значению в V3736.
SP604	Включается, когда текущее значение Счетчика равно значению в V3740.
SP605	Включается, когда текущее значение Счетчика равно значению в V3742.
SP606	Включается, когда текущее значение Счетчика равно значению в V3744.
SP607	Включается, когда текущее значение Счетчика равно значению в V3746.
SP610	Включается, когда текущее значение Счетчика равно значению в V3750.
SP611	Включается, когда текущее значение Счетчика равно значению в V3752.
SP612	Включается, когда текущее значение Счетчика равно значению в V3754.
SP613	Включается, когда текущее значение Счетчика равно значению в V3756.
SP614	Включается, когда текущее значение Счетчика равно значению в V3760.
SP615	Включается, когда текущее значение Счетчика равно значению в V3762.
SP616	Включается, когда текущее значение Счетчика равно значению в V3764.
SP617	Включается, когда текущее значение Счетчика равно значению в V3766.

Приложение E. ***Память ПЛК***

В этом приложении…

Память ПЛК DL205

При проектировании системы на ПЛК очень важно понимать, как устроена память в контроллере. В процессорном модуле контроллера DL205 используется два типа памяти: ОЗУ (RAM) и электрически перепрограммируемое ПЗУ (EEPROM). Вся эта память может быть сконфигурирована пользователем, как память постоянного хранения или временного хранения.

Память постоянного хранения - это память, которая конфигурируется пользователем так, чтобы данные в ней сохранялись при выключении и повторном включении питания или при переходе процессорного модуля из режима Программирования в Рабочий режим. Память временного хранения – это память, которая конфигурируется пользователем так, чтобы данные в ней стирались при выключении и повторном включении питания или при переходе процессорного модуля из режима Программирования в Рабочий режим. Области памяти постоянного хранения могут быть заданы с помощью Ручного Программатора функцией AUX 57 или в *Direct*SOFT (PLC Setup – Конфигурация ПЛК).

В памяти ОЗУ можно данные записать или прочитать бесчисленное количество раз, но для оперативной памяти (ОЗУ) требуется источник питания, который поддерживает сохранение данных в ней. Данные в ОЗУ сохраняются в том случае, когда этот источник питания (5 В постоянного тока) включен (к ПЛК подведено внешнее питание, обычно 220В переменного тока). Если питание ПЛК отключается, то содержимое ОЗУ может быть сохранено с помощью дополнительной батареи. Содержимое ОЗУ теряется, когда выключается внешнее питание, и нет питания со стороны батареи.

В памяти EEPROM (ЭППЗУ) можно данные записать или считать очень большое (но ограниченное) количество раз (типичное число записей – 100000 раз). Данные в ЭППЗУ сохраняются очень долго без источника питания.

Пользовательская V-память ПЛК может находиться как в энергозависимой ОЗУ, так и в энергонезависимой ЭППЗУ. Смотрите карту памяти, относящуюся к конкретному процессорному модулю.

Области памяти контроллеров DL205				
	DL230	DL240	DL250-1	DL260
ОЗУ (энергозависимое)	V2000 - V2377	V2000 - V3777	V1400 - V7377 V10000 - V17777	V400 - V777 V1400 - V7377 V10000 - V35777
ЭППЗУ	V4000 - V4177	V4000 - V4377	нет	нет

Данные, которые необходимо сохранять в течение длительного периода времени, когда ПЛК отключен от сети, следует помещать в V-память ЭППЗУ.

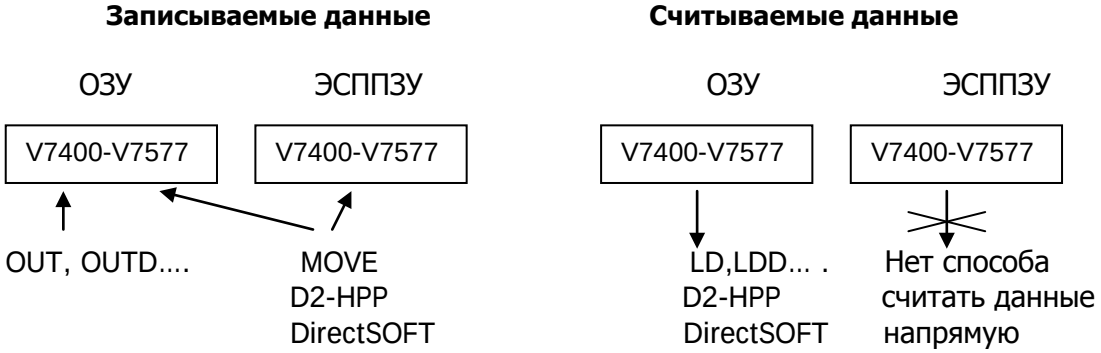
Данные, которые постоянно изменяются или могут быть инициализированы логикой программы, следует хранить в V-памяти ОЗУ

Долговременная V-память в контроллерах D2-230, D2-240 и DL05/06

Два типа памяти выделены для области долговременной V-памяти. Это ОЗУ (RAM) и ЭСППЗУ (flash ROM или EEPROM). Они используют одни и те же адреса V- памяти. Однако, Вы можете использовать **только команду MOVE, ручной программатор и DirectSOFT** , **чтобы записать данные в ЭСППЗУ.** Когда Вы пишите данные в ЭСППЗУ, те же самые данные записываются и в ОЗУ.

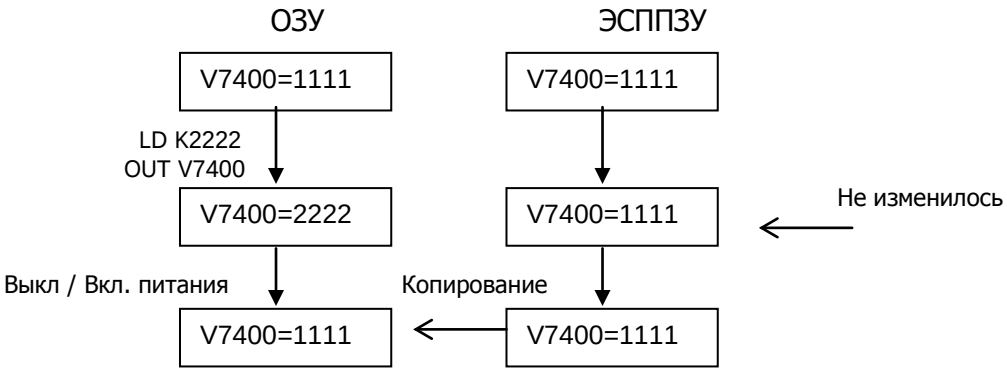
Если Вы используете другие команды, то можете писать только в ОЗУ.

Когда Вы читаете данные из области долговременной V-памяти, то всегда читаете данные из ОЗУ. Объяснение приведенное далее использует в качестве примера ЦПУ DL240.



После цикла включения питания контроллер всегда копирует данные ЭСПЗУ в ОЗУ.

Если Вы использовали другую команду, нежели MOVE для записи в область долговременной V-памяти, Вы только обновили данные в ОЗУ. После выключения/включения питания, контроллер копирует первоначальные данные из ЭСППЗУ в ОЗУ, так что данные, которые Вы ввели, исчезают. Чтобы избежать этого, используйте команду MOVE.



Выглядит так, будто вернулись первоначальные данные.

Приложение F. ***Вес изделий DL205***

В этом приложении…

Таблица весов изделий

ЦПУ	Вес
D2-230	80г.
D2-240	80г
D2-250-1	70г.
D2-260	70г.
Каркасы	
D2-03B-1	350г.
D2-03BDC1-1	322г.
D2-03BDC-2	285г.
D2-04B-1	381г.
D2-04BDC1-1	354г.
D2-04BDC-2	317г.
D2-06B-1	410г.
D2-06BDC1-1	392г.
D2-06BDC2-1	392г.
D2-09B-1	530г.
D2-09BDC1-1	522г.
D2-09BDC2-1	530г.
Входные модули пост.тока	
D2-08ND3	65г.
D2-16ND3-2	65г.
D2-32ND3	60г.
D2-32ND3-2	60г.
Входные модули перем.тока	
D2-08NA-1	70г.
D2-08NA-2	70г.
D2-16NA	68г.
Комбинированные модули	
D2-08CDR 3.5	100г

Выходные модули пост.тока	
D2-04TD1	80г.
D2-08TD1	65г.
D2-08TD2	60г.
D2-16TD1-2	65г.
D2-16TD2-2	80г.
F2-16TD1P	56г.
F2-16TD2P	56г.
D2-32TD1	60г.
D2-32TD2	60г.
Выходные модули перем. тока	
D2-08TA	80г.
F2-08TA	99г.
D2-12TA	80г.
Выходные модули релейные	
D2-04TRS	80г.
D2-08TR	114г.
D2-12TR	130г.
F2-08TR	156г.
F2-08TRS	156г.
Контроллеры каркасов	
H2-EBC	45г.
H2-EBC-F	60г.
F2-SDS-1	80г.
H2-PBC	60г.
F2-DEVNETS	86г.

Аналоговые модули	
F2-04AD-1	86г.
F2-04AD-2	86г.
F2-04AD-1L	86г.
F2-04AD-2L	86г.
F2-08AD-1	86г.
F2-08AD-2	118г.
F2-02DA-1	80г.
F2-02DA-2	80г.
F2-02DA-1L	80г..
F2-02DA-2L	80г.
F2-08DA-1 2	80г.
F2-08DA-2	109г.
F2-02DAS-1	109г.
F2-02DAS-2	109г.
F2-4AD2DA	118г.
F2-8AD4DA-1	118г.
F2-8AD4DA-2	118г.
F2-04RTD	86г.
F2-04THM	86г.
Специальные модули	
H2-CTRIO	65г.
H2-CTRIO2	62г.
D2-CTRINT	65г.
H2-ECOM	45г.
H2-ECOM100	43г.
H2-ECOM-F	156г.
H2-ERM(100)	45г.
H2-ERM-F	156г.
H2-SERIO	43г.
D2-DCM	109г.
F2-CP128	111г.
D2-EM	65г.
D2-CM	50г.
F2-08SIM	70г.

Приложение G. *Таблица ASCII*

В этом приложении…

Таблица преобразования кодов ASCII

DECIMAL TO HEX TO ASCII CONVERTER											
DEC	HEX	ASCII	DEC	HEX	ASCII	DEC	HEX	ASCII	DEC	HEX	ASCII
0	0	NUL	32	20	space	64	40	@	96	60	`
1	1	SOH	33	21	!	65	41	A	97	61	a
2	2	STX	34	22	“	66	42	B	98	62	b
3	3	ETX	35	23	#	67	43	C	99	63	c
4	4	EOT	36	24	$	68	44	D	100	64	d
5	5	ENQ	37	25	%	69	45	E	101	65	e
6	6	ACK	38	26	&	70	46	F	102	66	f
7	7	BEL	39	27	‘	71	47	G	103	67	g
8	8	BS	40	28	(	72	48	H	104	68	h
9	9	TAB	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C			76	4C	L	108	6C
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B			91	5B	[	123	7B
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Приложение H. ***Системы счисления***

В этом приложении…

В этом приложении приводятся краткие объяснения по наиболее часто используемым в ПЛК системам счисления

Двоичная система счисления - Binary Numbering System

Компьютеры и программируемые контроллеры используют систему счисления с основанием «2», которую называют двоичной или булевой (Binary или Boolean). В этой системе используются только две цифры НОЛЬ и ЕДИНИЦА (или ON и OFF).

Для представления больших используются комбинации из нескольких цифр.

Каждая цифра двоичной системе при использовании в компьютере называется – «бит».

Группа из четырех бит образует – тетраду (*nibble*).

Восемь бит или две тетрады – это байт (*byte*).

Шестнадцать бит или два байта – это слово (*word*).

Тридцать два бита или два слова – это двойное слово (*double word*).

Word (**Слово**)															
Byte (**Байт**)								Byte (**Байт**)							
Nibble (Тетрада)				Nibble (Тетрада)				Nibble (Тетрада)				Nibble (Тетрада)			
Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

В таблице показаны двоичные числа и их десятичные эквиваленты.

Например: значение байта 110101012 равно 213 ($1*2^7 + 1*2^6 + 1*2^4 + 1*2^2 + 1*2^0$ или $128_{10} + 64_{10} + 16_{10} + 4_{10} + 1_{10}$)

	Binary / Decimal – **Двоичные / Десятичные числа**															
Номер бита	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Степень	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Десятичное значение числа	32678	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Макс. значение	65535_{10}															

Шестнадцатиричная система счисления - Hexadecimal

Недостаток двоичной системы при ее использовании человеком являются большая длина числа и сложная интерпретация / преобразование числа.

Для устранения этих недостатков используются промежуточные системы. Одна из альтернативных систем счисления – это шестнадцатиричная (Hexadecimal или Hex).

Основание этой системы число 16. Представление чисел осуществляется цифрами от 0 до 10 и буквами от A до F

Десятичная	0	1	2	3	4	5	6	5	8	9	10	11	12	13	14	15	16	17
Шестнадцатиричная	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11

При работе с двоичной системой каждая тетрада (Nibble) может быть легко преобразована в шестнадцатиричный и десятичный эквивалент:

Тетрада 1111_2 эквивалентна 15_{10} или F Hex.

Шестнадцатибитовое слово можно представить в виде шестнадцатиричного эквивалентного числа в диапазоне от 0000 до FFFF Hex.

	Шестнадцатиричное число - Hexadecimal															
Номер бита	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Степень	16^3				16^2				16^1				16^0			
Значение бита	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
Макс. значение	F				F				F				F			

$A6D4_{16}$ эквивалентно 42708_{10} ($10*16^3 + 6*16^2 + 13*16^1 + 4*16^0$ или $40960_{10} + 1536_{10} + 208_{10} + 4_{10}$).

Восьмеричная система счисления - Octal

Восьмеричная система счисления похожа на шестнадцатиричную систему в интерпретации бит.

Отличия: основание – 8 и максимальное число-7. Двоичное слово разбивается на триады

	Восьмеричное число - Octal															
Номер бита	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Степень	8^5	8^3			8^2						8^1			8^0		
Значение бита	1	4	2	1	4	2	1	4	2	1	4	2	1	4	2	1
Максим. значение	1	7			7			7			7			7		

Пример: Число 63_8 эквивалентно 51_{10} ($6*8^1 + 3*8^0$ или 4810 + 310).

Двоично-кодированная десятичная система счисления - BCD

BCD (Binary Coded Decimal – Двоично-кодированная Десятичная) система подобна *Octal* и *Hexa-decimal* системам счисления. Она также основана на двоично-кодированных данных (см. таблицу). В ее основании число 10. Как видите, между BCD и Binary большое различие

	Двоично-кодированное десятичное число - BCD															
Номер бита	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Степень	10^3				10^2				10^1				10^0			
Значение бита	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
Максим. значение	9				7			7	9				9			

Пример: Число 638 эквивалентно 5110 (6*8 1 + 3*8 0 или 4810 + 310).

Реальные числа в формате с плавающей запятой - Real / Floating Point Numbering System

Под терминами **Реальные числа и числа с плавающей запятой** понимаются числа в формате соответствующем IEEE-754. Большинство ПЛК используют 32-х битовое представление чисел с плавающей точкой или реальных (Real).

Формула преобразования следующая:

$N = 1.M \times 2^{(E-127)}$

N: Число, которое надо представить в формате с плавающей запятой

M: Мантисса

E: Экспонента

	Реальные числа с плавающей запятой - Real (Floating-Point 32)															
Номер бита	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Знак	Экспонента								Мантисса						
Номер бита	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Мантисса - продолжение															

BCD / Binary / Decimal / Hex / Octal - Как различить?

Иногда происходят ошибки из-за различия типов данных, используемых в ПЛК. У контроллеров DirectLOGIC встроенный формат – BCD, а нумерация входов/выходов и памяти – октальная. При работе используются двоичный формат и формат реальных чисел. Хотя все данные хранятся в памяти в виде «0» и «1» ПЛК интрепретирует их по разному. В таблицах приведены битовые комбинации соответствующие разным форматам.

	Binary / Decimal – Двоичные / Десятичные числа															
Номер бита	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Степень	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Десятичное значение числа	32678	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Макс. значение	**65535_{10}**															
	Шестнадцатиричное число - Hexadecimal															
Номер бита	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Степень	16^{3}				16^{2}				16^{1}				16^{0}			
Значение бита	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
Макс. значение	**F**				**F**				**F**				**F**			
	Восьмеричное число - Octal															
Номер бита	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Степень	8^{5}	8^{3}			8^{2}						8^{1}			8^{0}		
Значение бита	1	4	2	1	4	2	1	4	2	1	4	2	1	4	2	1
Макс. значение	1	7			7			7			7			7		
	Двоично-кодированное десятичное число - BCD															
Номер бита	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Степень	10^{3}				10^{2}				10^{1}				10^{0}			
Значение бита	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
Макс. значение	**9**				**9**				**9**				**9**			

Путаница с типами данных

Путаница типов является общей проблемой при использовании операторского интерфейса. Так как формат BCD является внутренним форматом ПЛК, многие предполагают, что он взаимозаменяемый с форматом Binary (Unsigned Integer).
В таблице справа показаны сходство и отличия чисел в формате BCD и Binary. Как видно из таблицы, форматы BCD и Binary используют одни и те же наборы бит (bit pattern) для чисел до 10 (десятичного), но далее все меняется резко.
Битовое представление BCD числа 10 соответствует битовому представлению числа 16 в Binary формате. Чем больше число, тем больше разница.

Decimal	0	1	2	3	4	5	6	7	8	9	10	11
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	0001 0000	0001 0001
Binary	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	0000 1010	0000 1011

Целые числа со знаком или без знака – Sign / Unsigned Integers
Теперь рассмотрим, как обстоят дела при работе с отрицательными числами и форматами чисел со знаком. Формат BCD не может быть использован для работы с отрицательными числами.
Для указания того, что число отрицательное мы должны использовать специальный бит.
Обычно это MSB (*most significant bit*) – наиболее значимый бит. Для 16-битовых чисел мы будем иметь диапазон представления чисел от -32767 до 32767

Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MSB															LSB

Такая форма представления отрицательного числа называется *Magnitude plus Sign* (Значение плюс Знак)

Magnitude Plus Sign	
Decimal	Binary
100	0000 0000 0110 0100
-100	1000 0000 0110 0100

Вторая форма представления отрицательных чисел – это Two's Complement (Двоичный дополнительный код). Такой код получают, инвертируя двоичное число и добавляя единицу

Two's Complement	
Decimal	Binary
100	0000 0000 0110 0100
-100	1111 1111 1001 1100

Типы данных изделий AutomationDirect

В семействе контроллеров DirectLogic для работы с модулями ввода/вывода используется восьмеричная система счисления (Octal).
Вся информация в V-памяти хранится в BCD формате, если формат специально не изменен программистом. Это значит, что все математические операции должны производиться при помощи BCD функций. В некоторых моделях ПЛК Вы можете использовать формы:You
Binary, Real или Double-Word BCD. Для изменения типа данных (data types), необходимо использовать Функциональные блоки (function box). Для перевода из BCD в Binary используют функциональный блок BIN. Из BCD в Real Вам сначала надо использовать блок
BIN , а затем блок BTOR. Вы не можете сложить число в формате BCD или Binary с числом в формате Real, или BCD с числом в Binary и получить корректный результат. Типы данных должны совпадать.
Есть несколько моментов связанных с типами данных, на которые надо обращать внимание при работе.
1. Модули аналогового ввода могут быть настроены на выдачу результата измерений в формате **Binary** или *BCD*.
2. Вторая область, где не все в BCD, это ПИД-регулятор (PID). Почти все параметры регулятора хранятся, как двоичные числа (Binary). В ПИД-регуляторе используется представление отрицательных двоичных чисел в виде числа со знаком (*Magnitude Plus Sign*), тогда как в других математических функциях ПЛК используется представление отрицательных чисел в дополнительном коде (*Two's Complement*).
3. Будьте внимательны при работе с окнами просмотра данных (Data view). Различить Вы их можете в режиме просмотра данных программы - Data view . Удостоверьтесь в том, что Вы выбрали соответствующий формат и длину представления данных в ниспадающем меню расположенном в верхней части окна.
И последнее. Вы заметили, что формат BCD в data view обозначается - BCD/Hex формат. Ранее было показано, что эти числа одной длины и различаются только использованием букв A - F, поэтому используется одно окно представления .

Панели оператора C-more/C-more Micro

В панелях оператора C-more/C-more Micro:

- Формат 16-bit BCD обозначается, как BCD int 16.
- Двоичный (Binary) формат, как Unsigned int 16 или Signed int 16 в зависимости от того - может ли значение быть отрицательным.
- Формат чисел с плавающей запятой (Real) обозначается, как Floating PT 32.

Часто используются форматы: "BCD int 32", "Unsigned int 32" и "Signed int 32".

Приложение I. ***Директивы Европейского Союза (CE)***

В этом приложении…

- Директивы Европейского союза (ЕС)
- Основные руководящие указания по электромагнитной совместимости (ЭМС) оборудования

Директивы Европейского Союза (ЕС)

ПРИМЕЧАНИЕ: *Информация, содержащаяся в этом разделе, предлагается в качестве справочного материала, она базируется на нашей интерпретации различных стандартов и требований. Поскольку реальные стандарты издаются другими организациями, в некоторых случаях – Правительственными органами, то эти требования могут периодически изменяться без предварительного предупреждения или извещения. Изменения и дополнения к этим стандартам могут сделать недействительной любую часть информации, приводимой в данном разделе.*

Сертификация и аттестация – абсолютно необходимые сферы деятельности для каждого, кто хочет заняться бизнесом в Европе. Одной из ключевых проблем, с которой столкнулись страны – члены ЕС и Европейского Экономического Союза (ЕЭС), состоит в том, чтобы привести несколько подобных, но пока еще различных стандартов, к одному стандарту, общему для всех стран – членов. Основная цель единого стандарта состоит в том, чтобы облегчить торговлю и транспортировку товаров между различными странами, создать условия для безопасной работы, сохранить окружающую среду. Директивы, которые являются результатом объединения стандартов, в настоящее время представляют собой официальные требования для всех, кто ведет бизнес в Европе. Изделия, удовлетворяющие этим Директивам, должны иметь марку CE, которая подтверждает их соответствие этим Директивам.

Страны-члены ЕС

В настоящее время членами ЕС являются: Австрия, Бельгия, Дания, Финляндия, Франция, Германия, Греция, Ирландия, Италия, Люксембург, Нидерланды, Португалия, Испания, Швеция и Великобритания. Исландия, Лихтенштейн и Норвегия наряду с другими членами ЕС входят в Европейский Экономический Союз (ЕЭС), все эти страны поддерживают данные Директивы.

Применяемые директивы

Существует несколько Директив, которые применимы к нашим изделиям. При необходимости в Директивы могут вноситься поправки, они могут дополняться.

- **Директива по электромагнитной совместимости (ЭМС)** – эта Директива направлена на то, чтобы устройства, аппаратура и системы удовлетворительно функционировали в электромагнитной среде и не оказывали чрезмерного электромагнитного воздействия на что-либо в этой среде.
- **Директива по безопасности машинного оборудования** – эта Директива включает аспекты безопасности аппаратуры, установок и др. Она затрагивает несколько областей, включая стандарты на тестирование, относящиеся как к защите от электрических помех, так и к генерации этих помех.
- **Директива по низкому напряжения** – эта Директива относится к электробезопасности и касается оборудования, которое работает в диапазоне напряжений 50 – 1000 В переменного тока и/или 75 – 1500 В постоянного тока.
- **Директива по батареям** – эта Директива относится к производству батарей, их повторному использованию и утилизации.

Соответствие директивам

Определенные стандарты в рамках каждой Директивы уже требуют обязательного соответствия, например, Директива по электромагнитной совместимости, которая получила наибольшее внимание, и Директива по низкому напряжению.

В конечном счете, мы все ответственны за различные аспекты проблемы. Как изготовители, мы должны проводить испытания наших изделий документировать любые результаты испытаний и/или методов сборки, которые необходимы для соблюдения требований Директив. Как механики, вы ответственны за такую установку оборудования, которая гарантирует его работу в соответствии с Директивами. Вы также ответственны за испытания любых комбинаций оборудования, которые при совместном использовании могут соответствовать требованиям Директив (или не соответствовать).

Конечный пользователь оборудования должен соблюдать любые Директивы, относящиеся к техническому обслуживанию, утилизации и пр. аппаратуры и ее различных компонентов. Хотя мы и прилагаем большие усилия, мы не в состоянии испытать все возможные конфигурации наших изделий по отношению к любой конкретной Директиве. Поэтому, в конечном счете, на вас лежит ответственность за то, чтобы ваше оборудование (и вся установка в целом) соответствовало этим Директивам, и чтобы эти требования поддерживались в процессе эксплуатации. **Соответствие директивам CE считается нарушенным, если установка оборудования не произведена согласно рекомендованному руководству по установке.**

В настоящее время контроллеры производства фирм *Koyo Electronics Industries, FACTS Engineering* или *Host Engineering* при надлежащей установке и использовании соответствуют требованиям Директив по электромагнитной совместимости, низкому напряжению в составе следующих стандартов:

- Стандарты Директивы по электромагнитной совместимости, относящиеся к ПЛК
  - EN50081-1 Общий стандарт по защищенности для бытовой техники, торгового оборудования и легкой промышленности
  - EN5008 1-2 Общий стандарт по излучению для промышленности
  - EN50082-1 Общий стандарт по защищенности для бытовой техники, торгового оборудования и легкой промышленности
  - EN50082-2 Общий стандарт по излучению для промышленности.
- Стандарты Директивы по низкому напряжению относящиеся к ПЛК
  - EN61010-1 Требования к безопасности электрического оборудования для измерений, управления и лабораторного применения
- Специальный стандарт для ПЛК

  EN61131-2 Программируемые контроллеры, требования к оборудванию и испытаниям.

Этот стандарт заменяет указанные выше стандарты по защищенности и безопасности. Однако, общие стандарты по излучению должны пока применяться вместе со следующими стандартами: - EN 61000-3-2 – Колебания (*Harmonics*)

- EN 61000-3-2 – Случайные составляющие (флуктуации)

ПРЕДУПРЕЖДЕНИЕ по электростатическому разряду (ESD). Мы рекомендуем, чтобы персонал соблюдал меры предосторожности связанные с переносом электростатических зарядов внутрь шкафа управления.

ПРЕДУПРЕЖДЕНИЕ по радионаводкам (RFI): Это изделие класса A. В домашних условиях это изделие может создавать радио-помехи и возможно придется принять адекватные меры.

Специальное руководство по установке

Требования к установке, соответствующие требованиям Директив по машинному оборудованию, электромагнитной совместимости и низкому напряжению, немного сложнее обычных требований к установке, принятых в США. Для помощи вам в этом вопросе мы издали специальное руководство, которое вы можете «скачать» с сайта www.AutomationDirect.com.

- **DA-EU-M** – Руководство по установке, которое включает конкретные требования к установке оборудования, удовлетворяющие Директивам ЕС. «Скачайте» это руководство с сайта, и вы получите самую последнюю информацию

Другие источники информации

Хотя Директива по электромагнитной совместимости получила наибольшее внимание, другие базовые Директивы, например, Директивы по машинному оборудованию и низкому напряжению, также накладывают ограничения на построение панели управления. Чтобы вы могли учесть эти дополнительные требования, рекомендуем приобрести и использовать в качестве руководства следующие публикации:

- TH 42073 (публикация Британского института стандартов), 1996 г. включает аспекты безопасности и электрические аспекты Директив по машинному оборудованию.
- EN 60204-1:1992 – Общие электрические требования к машинному оборудованию, включая требования по низкому напряжению и электромагнитной совместимости.
- IEC 1000-5-2: Электромагнитное заземление и технические требования к прокладке кабеля.
- IEC 1000-5-1: Общие положения по электромагнитной совместимости

Вы можете получить эту информацию в местных организациях, однако, официальным источником применяемых Директив и соответствующих стандартов является

The Office for Official Publications of the European Communities (Агенство официальных публикаций Европейского Сообщества),
L-2985 Luxembourg; самый быстрый способ связи через интернет -
http://publications.europa.eu/index_en.htm

Основные руководящие указания по электромагнитной совместимости (ЭМС) оборудования

Шкафы

На следующем рисунке представлен самый простой способ выполнения требований Директив по машинному оборудованию и низкому напряжению. Он состоит в том, чтобы поместить всю аппаратуру управления в стандартный запираемый стальной шкаф, а провода и кабели проложить в металлических коробах.

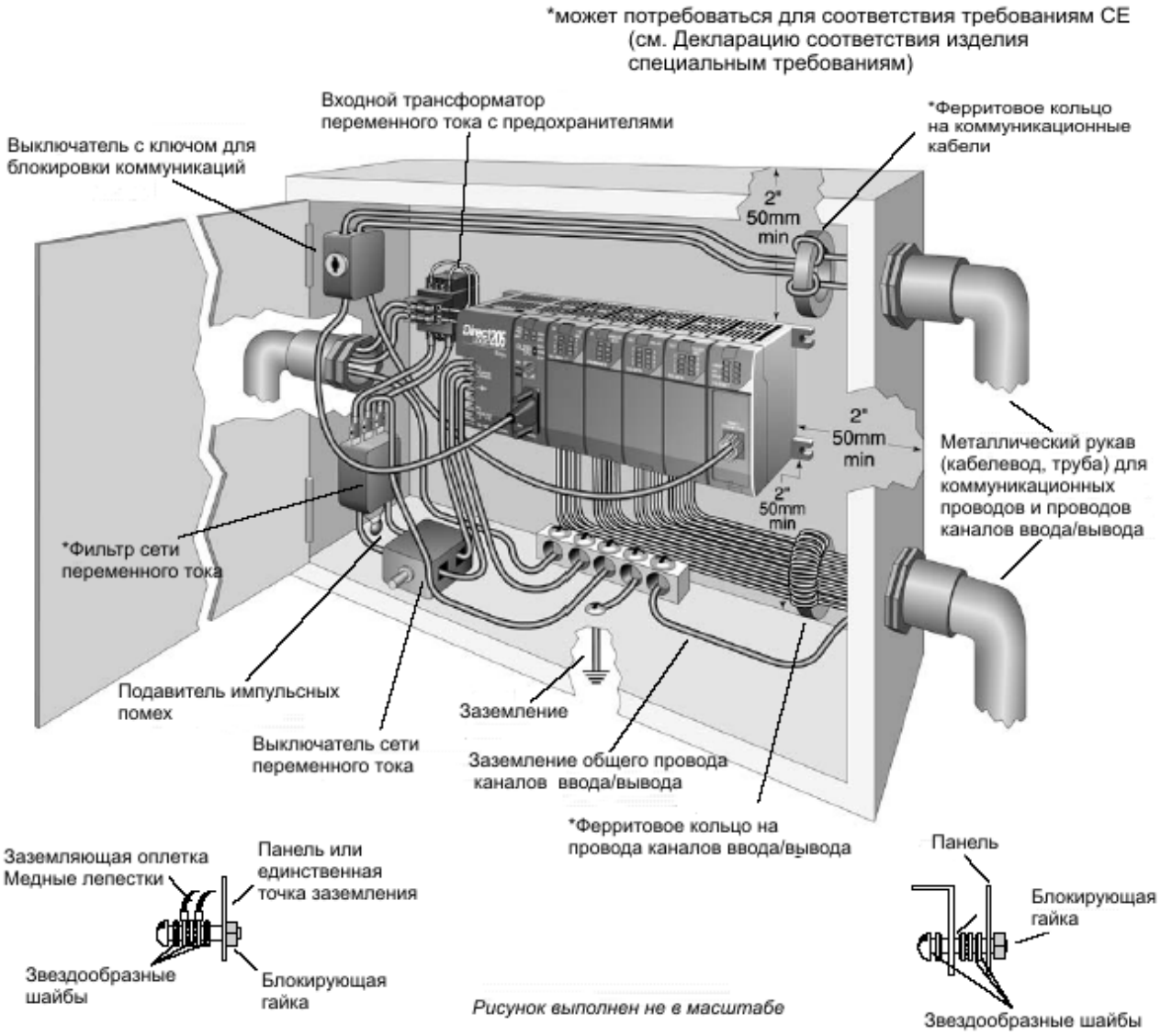


Рисунок выполнен не в масштабе

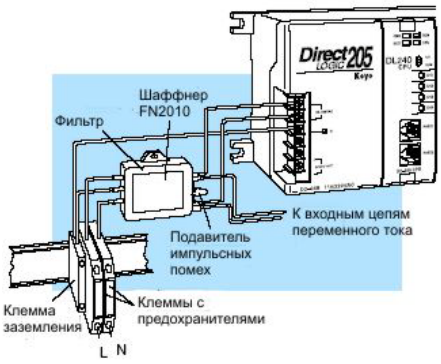
Электростатический разряд

Мы заявляем во всех декларация о соответствии, что наши изделия устанавливаются внутри металлических шкафов и внешние провода прокладываются внутри металлических рукавов; поэтому для испытаний наших изделий мы их устанавливаем в типичный шкаф. Однако, мы хотели бы подчеркнуть, что наши изделия работают нормально в присутствии источников электростатического разряда только в том случае, когда они помещены внутри закрытого промышленного шкафа управления. Когда шкаф открыт во время установки оборудования или при техобслуживании, то оно и/или программы подвергаются риску повреждения от электростатического разряда, источником которого может оказаться обслуживающий или оперативный персонал.

Поэтому мы рекомендуем обслуживающему персоналу принимать все необходимые меры предосторожности к тому, чтобы избежать риска передачи статического электричества на компоненты, установленные внутри шкафа управления. Если это необходимо, то следует снаружи шкафа устанавливать четкие предупреждения и инструкции, в которых предписывается использовать заземляющие шины или подобные устройства, или отключать питание устройств внутри шкафа.

Фильтры в сети переменного тока

Чтобы обеспечить соответствие требованиям Директив по электромагнитной совместимости в части кондуктивного радиоизлучения, блок питания каркаса контроллера DL205 должен иметь дополнительный фильтр в сети переменного тока. Вся аппаратура ПЛК испытывается с фильтрами фирмы Schaffner, которые уменьшают уровень излучения в сеть при их надлежащем заземлении. Следует выбирать фильтр с номинальным током, достаточным для того, чтобы запитать все источники питания ПЛК и все входы переменного тока. Для систем на основе контроллера DL205 мы предлагаем фильтр FN2010.



ПРИМЕЧАНИЕ: *Очень немногие фильтры в питающей сети переменного тока могут уменьшить проникновение помех в сеть до пренебрежимо малых уровней. В некоторых случаях фильтры могут даже увеличить уровень кондуктивных помех в сети, если эти фильтры подобраны неправильно. Фильтры, показанные выше, не являются «силовыми фильтрами», которые применяются для того, чтобы предотвратить попадание переходных помех из сети в источник питания ПЛК.*

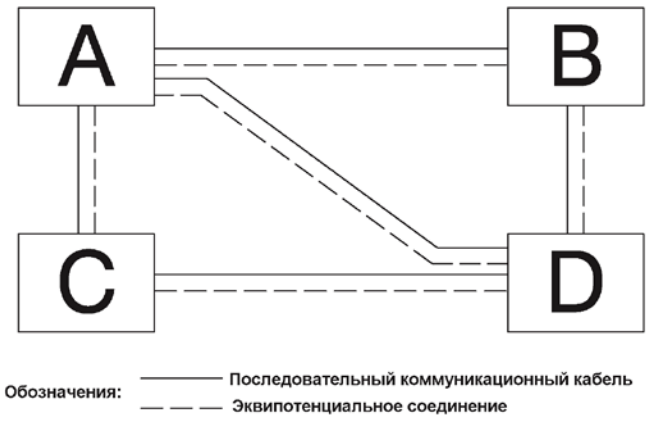
Подавление помех и установка предохранителей

Чтобы обеспечить соответствие требованиям электротехнических стандартов EN61010-1 и EN60204-1 Директив по машинному оборудованию и низкому напряжению в части пожарной безопасности посредством ограничения мощности в схемах сетевого питания «без ограничений» с реверсивными силовыми приводами, необходимо устанавливать предохранители на входах питания как переменного, так и постоянного тока. Необходимо также установить между точками подключения сети электропитания ПЛК подавитель бросков напряжения источника питания. Выбирайте подавитель типа варистор на окислах металла (MOV) с номиналом рабочего напряжения 275 В переменного тока при номинальном напряжении питания 230 В (с рабочим напряжением 150 В при напряжении питания 115 В) и с высокой энергетической мощностью (например, 140 джоулей). Подавитель импульсов питания должен быть защищен предохранителями, его мощность должна быть больше мощности перегорания предохранителя или автомата-прерывателя цепи, чтобы избежать риска возникновения пожарной опасности. Рекомендуемая схема входного питания переменного тока для ПЛК Koyo включает применение спаренных клемм ТТ на 3 А, снабженных предохранителями с индикацией их перегорания, или спаренных прерывателей цепи, подсоединенных к фильтру Schaffner FN2010 или к его эквиваленту, с высокоэнергетическим подавителем импульсов тока, расположенному непосредственно между выходными клеммами фильтра. Входы ПЛК также должны быть защищены от импульсов напряжения с помощью предохранителей, фильтров и ограничителей перенапряжения, установленных в схеме их питания.

Внутреннее заземление шкафов

В каждом шкафу должны быть предусмотрен мощный клеммный блок для звездообразного подключения всех шин заземления, схем защитного заземления, проводов заземления фильтров сетевого питания, а также для подключения заземления механических узлов. Это необходимо для выполнения требований по электробезопасности и электромагнитной совместимости, местных стандартов, а также стандарта IEC 1000-5-2. Директива по машинному оборудованию требует, чтобы общие полюса входных модулей ПЛК и общий провод питания нагрузок, управляемых выходными модулями ПЛК, также подключались к клемме защитного заземления.

Эквипотенциальное заземление



Для оборудования, содержащего современные электронные схемы, должен быть предусмотрен соответствующий узел заземления. Применение для электронных систем отдельных заземляющих проводов в некоторых странах запрещено. Проверьте все требования для вашего конкретного размещения заземления. Стандарт IEC 1000-5-2 предусматривает эквипотенциальное соединение всех сеток заземления. Но особое внимание следует уделить аппаратуре и шкафам управления, которые содержат устройства ввода/вывода, каркасы удаленного ввода/вывода, или блоки, имеющие межсистемные связи с основным шкафом системы с ПЛК. Необходимо проложить эквипотенциальный провод параллельно всем последовательным коммуникационным кабелям и параллельно кабелям к любым отдельным объектам предприятия, которые имеют устройства ввода/вывода, подключенные к ПЛК. На схеме выше показан пример из четырех физических объектов, которые соединены коммуникационным кабелем.

Связи и экранированные кабели

Для кабельной разводки аналоговых сигналов и кабельной разводки коммуникаций вне шкафа, в котором установлен ПЛК, рекомендуется экранированный кабель высокого качества из витой пары типа 24 AWG (минимум 0,2 мм^2) с экраном из фольги и оплетки.
До последнего времени считалось, что экран кабеля следует заземлять только с одного конца, чтобы снизить уровень помех, вызванных контурными токами между отдельными приборами. Метод заземления только с одного конца возник, как попытка уменьшить радиопомехи в аудиосистемах, и он больше не применяется в сложном промышленном оборудовании. Экранированные кабели являются эффективными передатчиками радиочастотных помех от систем, построенных на ПЛК, и передаваемый ими шум может отрицательным образом взаимодействовать с сигналами в коммуникационных сетях и с другими источниками помех.
В настоящее время рекомендуется использовать экранированные кабели в качестве электростатических «трубок» между приборами и системами и прокладывать вдоль всех экранированных кабелей провода большого диаметра для выравнивания потенциалов между устройствами (эквипотенциальное соединение). В том случае, когда экранированный кабель проходит через металлическую стенку шкафа или кожух машины в IEC 1000-5-2 рекомендуется присоединять экран по всему его периметру к стенке, предпочтительно используя проводящий адаптер. Не рекомендуется соединять экран через гибкий провод к болту заземления. Экраны должны присоединяться к каждой стенке шкафа или к кожуху каждой машины, через которые они проходят.

ПРИМЕЧАНИЕ: *В том случае, когда кабели, в независимости от того экранированы они или нет, находятся снаружи шкафа управления с ПЛК, их НЕОБХОДИМО укладывать внутри заземленных металлических рукавов (труб) или других металлических каналов (лотков).*

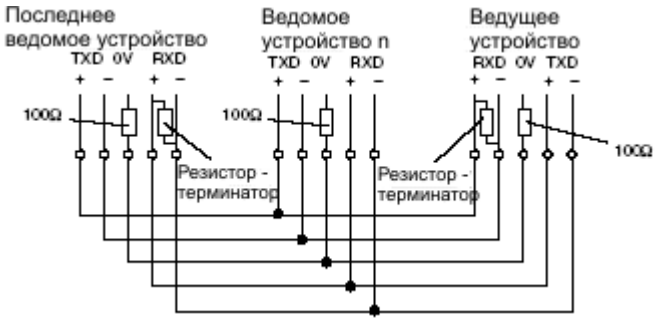
Кабели для аналоговых сигналов и для RS-232

Заземление обоих концов экрана в аналоговых цепях обеспечивает идеальную электрическую среду для витой пары, так как контур включает сигнальный и обратный провода. Это справедливо для полностью сбалансированных схем и при подключении к общему проводу входных схем, введенному на клеммы модуля.

Этот метод применим также к кабелям RS-232.

Многоточечные кабели

В кабелях из двух витых пар для RS-422 и из одной витой пары для RS-485 требуются предусматривать нулевой провод (линия связи 0 В), роль которого раньше часто выполнял экран кабеля. В настоящее время рекомендуется использовать кабель из трех витых пар для линий связи RS-422 и кабель из двух витых пар для линий связи RS-485. Дополнительная пара может использоваться как 0 В межсистемная линия связи. Источники питания постоянного тока, заземленные в обоих системах, также соединяются между собой 0 В межсистемной линией связи. Инструкции по установке требуют создания таких контуров заземления, которые имеют малое сопротивление за счет применения эквипотенциальных проводов большого диаметра. **Для не европейских систем, которые используют заземление на одном конце и которые установлены в местах с характеристиками заземления, далекими от идеальных, мы рекомендуем установить дополнительные резисторы по 100 Ом на каждой линии связи 0 В в сетевых и коммуникационных кабелях.**



Экранированные кабели внутри шкафов

При прокладке кабелей между различными элементами контроллера внутри шкафа, в котором также находится чувствительная электронная аппаратура других изготовителей, следует помнить, что эти кабели могут быть источником радиочастотных помех. Есть способы уменьшения этих помех. Стандартные кабели для передачи данных, соединяющие ПЛК и интерфейсы оператора, должны прокладываться вдали от другого оборудования и связывающей его кабельной проводки. Можно использовать специальные кабели для связи по последовательному каналу, в которых экран подсоединяется с обоих концов к заземлению шкафа таким же образом, как и внешние кабели.

Отключение устройств связи

В целях обеспечения безопасности в Директивах по машинному оборудованию предусмотрено требование установки выключателя, который отключает на период обслуживания входные сигналы сети, при этом удаленные команды не могут повлиять на работу оборудования. FA-ISONET не имеет выключателя. Используйте блокировку клавиатуры и выключатель на вашем шкафу, который при открытии шкафа отключает питание FA-ISONET. Чтобы исключить ввод помех в систему, необходимо все узлы выключателей помещать в отдельные заземленные стальные корпуса, а также поддерживать целостность экранированного кабеля.
Для получения дополнительной информации по директивам ЕС мы рекомендуем вам получить копию Руководства ЕС по Установке (DA-EU-M). Вы можете также просмотреть официальный сайт Комиссии ЕС: http://europ.eu.int/

Аналоговые модули и влияние электромагнитного излучения

Все изделия AutomationDirect протестированы на выдерживание напряженности электромагнитного поля – 10 В/м, что соответствует максимальным требованиям европейских стандартов. Хотя все продукты проходят это тестирование, аналоговые модули показывают отклонения в измерениях. Это совершенно нормально, но разработчики систем управления должны избегать этого и планировать меры предотвращения наводок.

Требования, специфические для контроллеров DL205

- Это устройство необходимо установить должным образом в соответствии с указаниями руководства по установке ПЛК DA-EU-M и требований стандартов по установке IEC 1000–5–1, IEC 1000–5–2 и IEC 1131–4.
- Номинальное значение изоляции между всеми цепями в контроллере определяет только основное значение изоляции, соответствующее условию возникновения одной неисправности.
- В этих изделиях не предусмотрена изоляция между ПЛК и аналоговыми входами
- Системный инженер отвечает за правильность заземления всех управляющих и силовых схем, а также за заземление оплетки экранированных кабелей.
- Входные силовые кабели должны иметь внешние предохранители и смонтированные снаружи выключатели, которые предпочтительнее устанавливать вблизи ПЛК. **Замечание:** Внутренний источник питания каркаса контроллера DL205 имеет медленнодействующий предохранитель 2A 2A/250B; однако, он незаменяемый, поэтому требуется внешний предохранитель 3 A типа T(с замедленным срабатыванием).
- Инструкции по техническому обслуживанию и поиску неисправностей вы найдете в разделе «Обслуживание и поиск неисправностей» этого руководства пользователя. В этом же разделе находится информация о замене батареи. Также следует использовать для замены при ремонте только модули, которые поставляются ***AutomationDirect.com*** или их партнерами.
- Кабели, в независимости от того экранированы они или нет, если они находятся снаружи шкафа управления с ПЛК, **НЕОБХОДИМО** укладывать внутри заземленных металлических труб или других металлических каналов (лотков).
- Защита, установленная в устройстве может быть ухудшена, если устройство используется способом, не предусмотренным производителем.