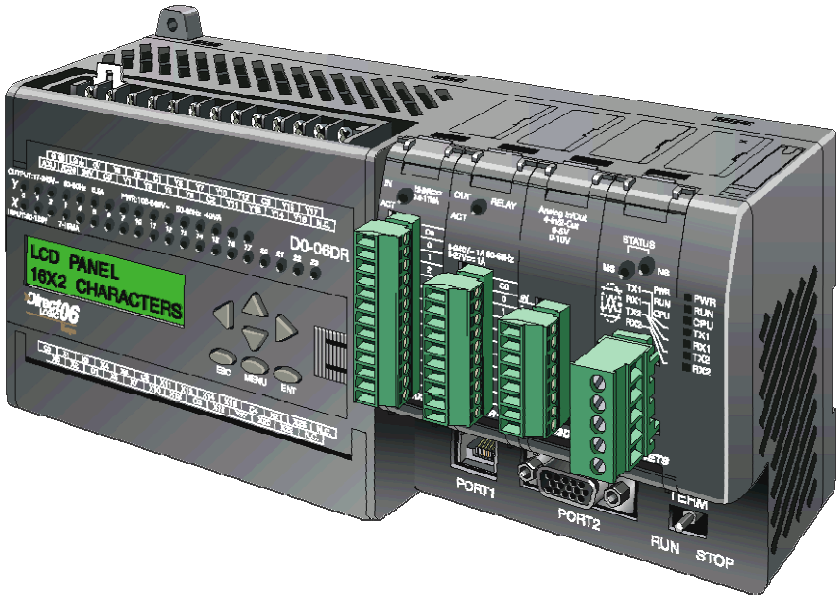
Аппаратные и программные средства АСУ ТП
тел.: (495) 105.77.98
E-mail: info@plcsystems.ru
Сайт: www.plcsystems.ru

Руководство пользователя контроллера DL06

Номер руководства D0-06USER-M-RUS
Часть 2



Книга 2.

Оглавление.

10.ЖИДКОКРИСТАЛЛИЧЕСКАЯ ПАНЕЛЬ

11.ПРИЛОЖЕНИЕ A. ВСПОМОГАТАЛЬНЫЕ ФУНКЦИИ

12.ПРИЛОЖЕНИЕ B. КОДЫ ОШИБОК DL06

13.ПРИЛОЖЕНИЕ C. ВРЕМЯ ВЫПОЛНЕНИЯ КОМАНД

Глава 6

6. Программирование с помощью барабанного командоаппарата

В данной главе…

Введение

Назначение

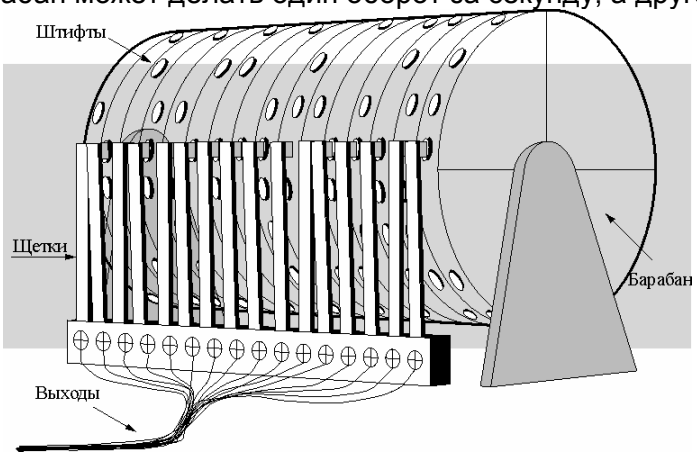
Команды DRUM и Event DRUM (EDRUM), существующие в ЦПУ DL06, служат для программной имитации электромеханического барабанного командоаппарата. Команды предполагают некоторое расширение принципов работы электромеханического барабанного командоаппарата, о чем написано ниже.

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.

Описав 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.

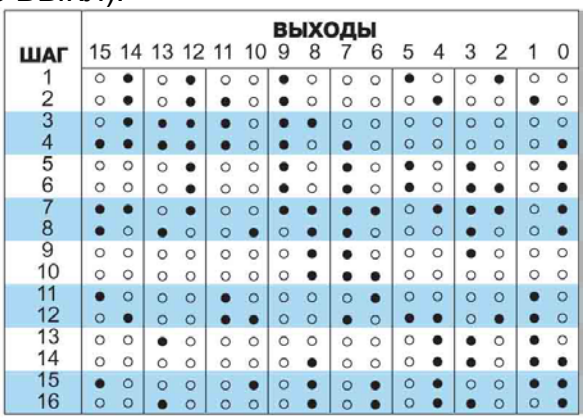
За один оборот барабан проходит конечное число положений, называемых **шагами(steps)**. Каждый шаг представляет собой некоторое состояние процесса. При включении питания барабан **возвращается(reset)** к определенному шагу. Барабан переходит от одного шага к другому по времени или по событию. При определенных 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)**, которая 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!

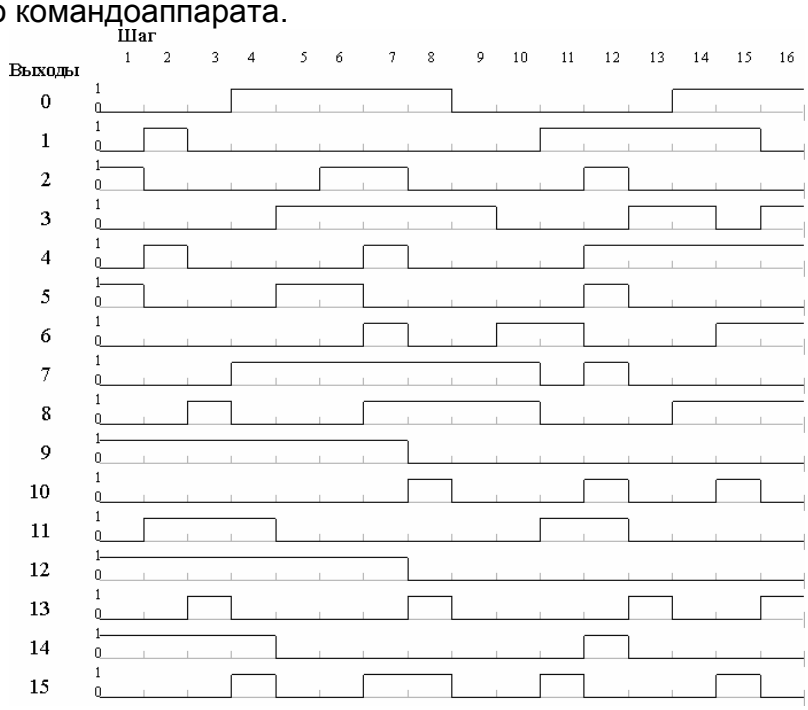
Представление барабанного командоаппарата в виде диаграммы

В этом руководстве, как и при редактировании в *DirectSOFT32,* 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 эту эквивалентность, вам будет несложно понять принцип работы команд барабанного командоаппарата.



Переходы между шагами

Типы команд барабанного командоаппарата

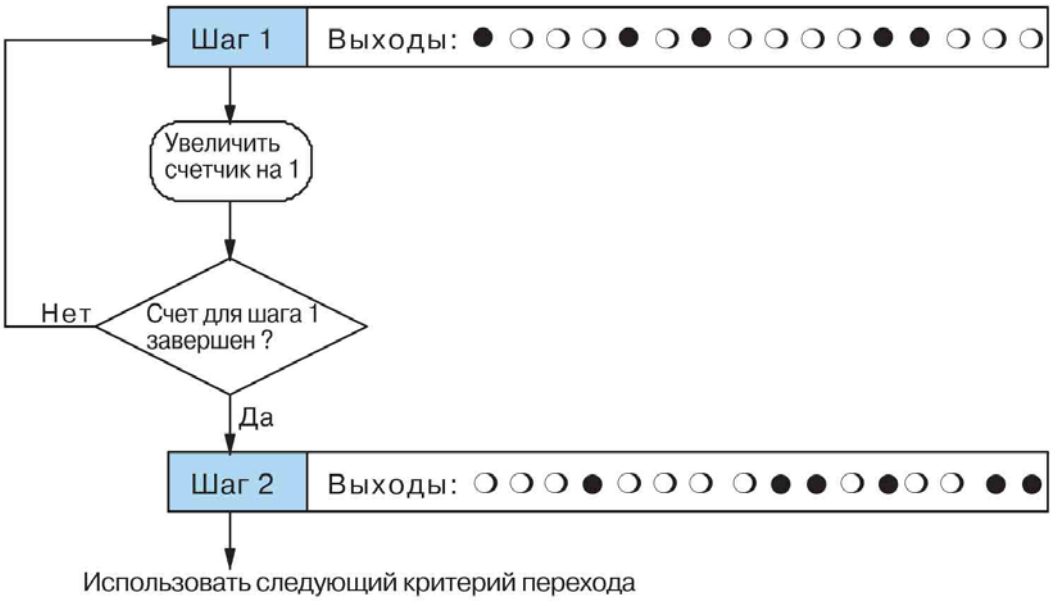
В ЦПУ DL06 имеются два типа команд барабанного командоаппарата.

- 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).

Оба типа команд поддерживают переходы по времени. EDRUM также поддерживает переход между шагами по событию. Каждый барабанный командоаппарат обеспечивает 16 шагов, по 16 выходов на каждом шаге. Обратимся к рисунку ниже. Каждый из выходов может задавать значение X, Y или обмотку C, обеспечивая гибкость программирования. Присвоим шагу 1 произвольную уникальную комбинацию состояний выходов.

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, задается 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.

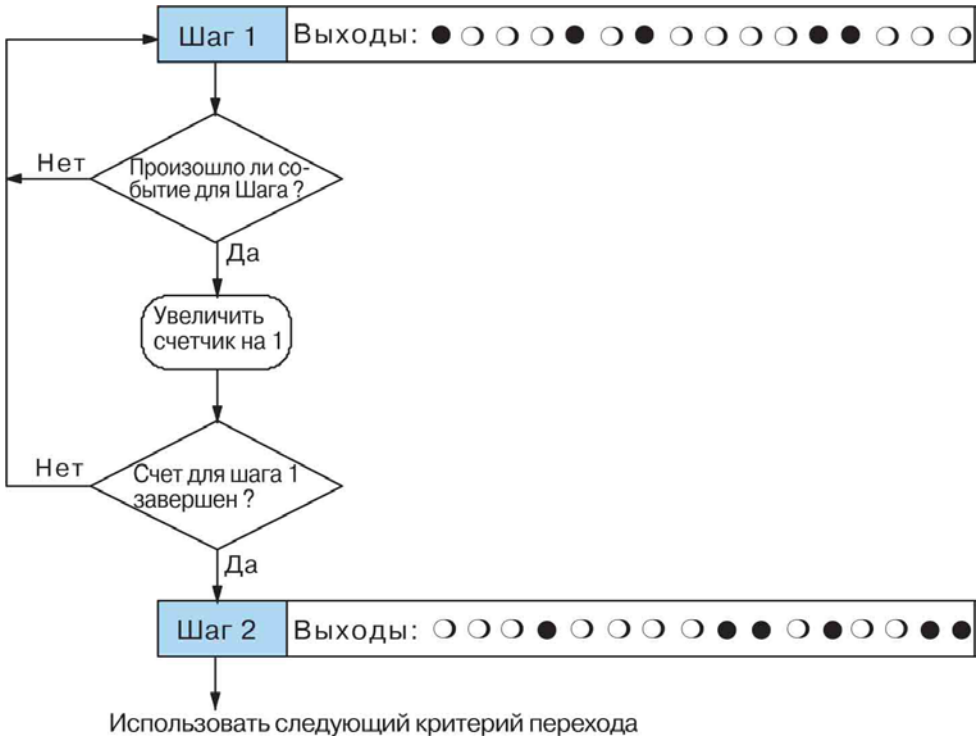
Например, если вы задаете в программе временной масштаб (длительность такта) 5 секунд и назначаете 12 тактов для Шага 1, длительность Шага 1 после перехода к нему барабанного командоаппарата составит 60 секунд. Максимальная длительность каждого шага определяется по следующей формуле:

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 позволит 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. Поэтому задавать длительность такта барабана меньше длительности цикла ЦПУ нет никакого смысла.

Переходы по времени и по событию

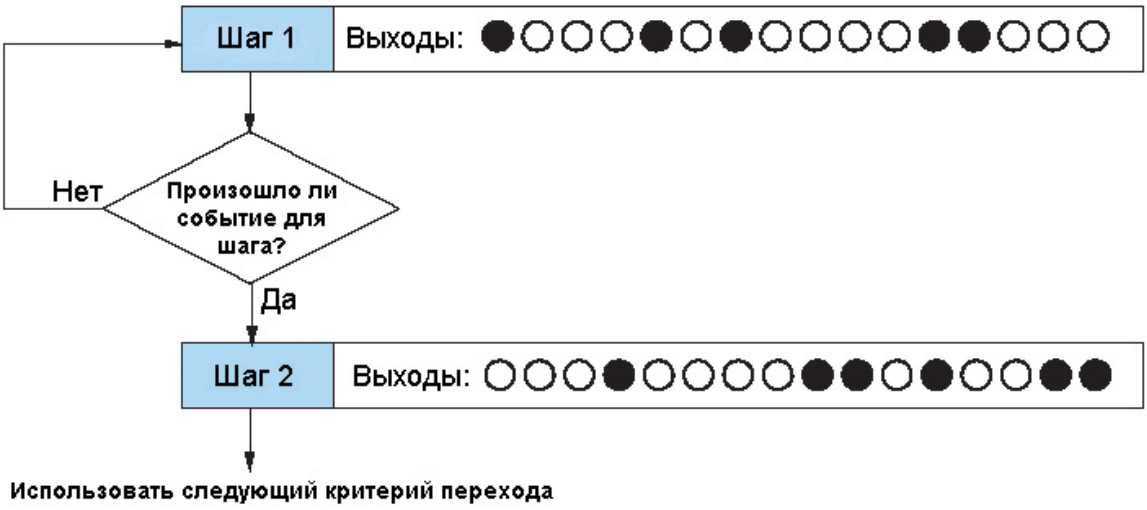
Переходы между шагами могут также осуществляться как по времени, так и /или по событию. На рисунке ниже показано, как происходят переходы в этом случае.



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 ЦПУ. 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.

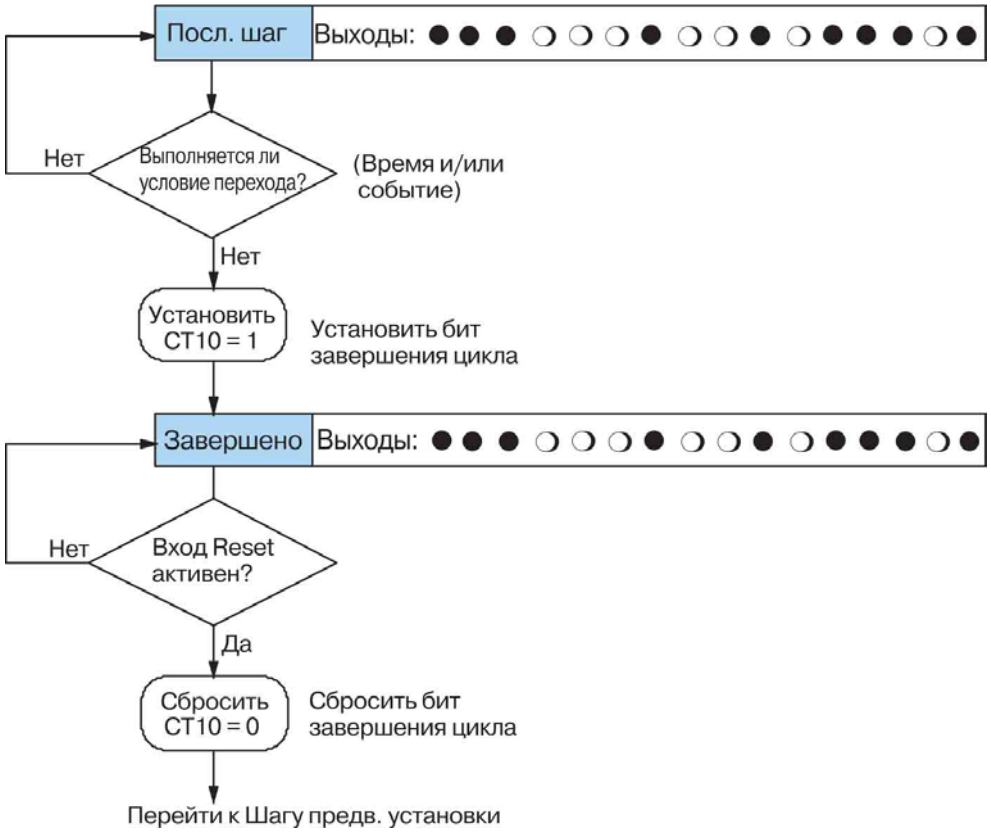
Предположим, что мы программируем счетчик на 8 шагов и выбрали для этих целей счетчик CT10 (следует помнить о восьмеричной нумерации счетчиков). Использование счетчиков показано в таблице справа. Правый столбец содержит типичные значения, пояснения к которым приводятся ниже.

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 исполняет 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 100 = 1528 тактов). 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

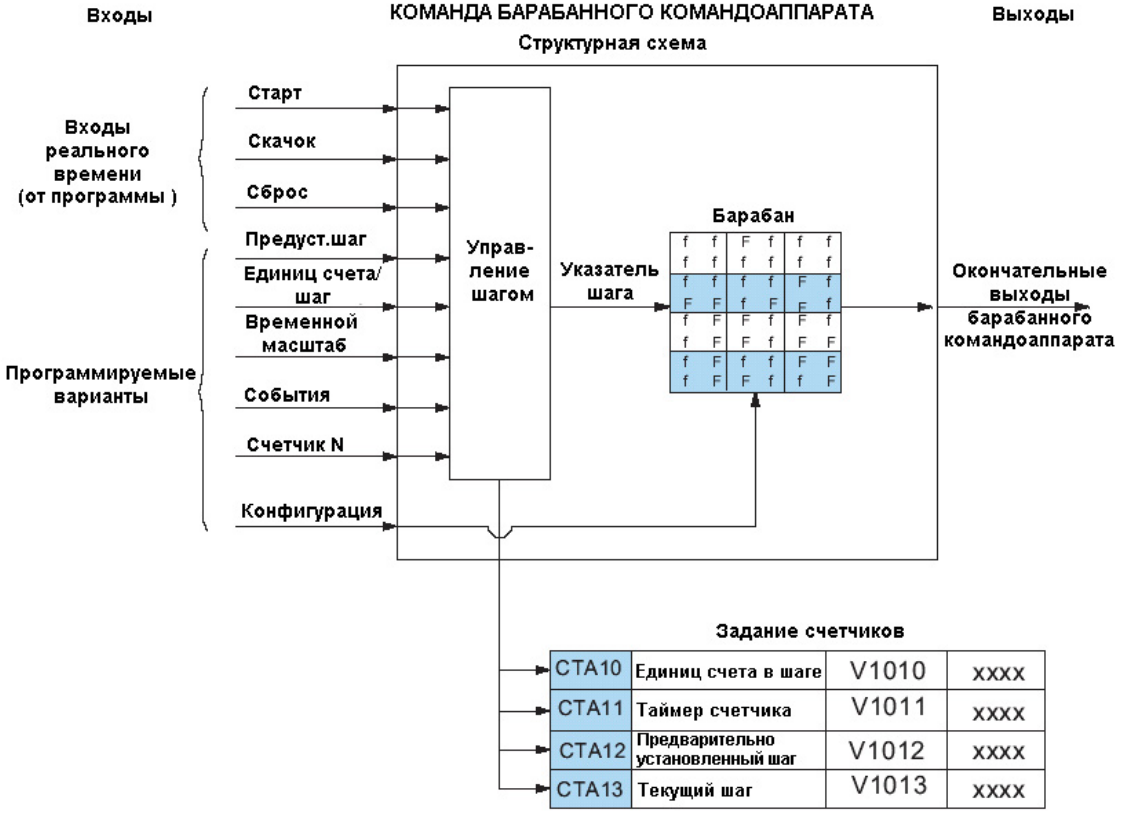
В качестве последнего шага в последовательности барабанного командоаппарата может быть шаг с любым номером, так как возможна реализация неполных барабанных командоаппаратов (см. рисунок ниже). Когда выполнены условия перехода для последнего шага, командоаппарат устанавливает бит счетчика, заданного в 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 из состояния ВЫКЛ в состояние ВКЛ входа Jog 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 сброшен, 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 определяет первый из расположенных друг за другом счетчиков, использующихся барабанным командоаппаратом для управления шагами. Их значение можно контролировать, чтобы следить за ходом выполнения цикла барабанного командоаппарата. В DL06 имеется 128 счетчиков (CT0 – CT077, восьмеричная нумерация).
- **Events (События)** – дискретная точка типа X, Y, C, S, T или CT, служащая в качестве входа для перехода между шагами. Для барабанных командоаппаратов с переходом по времени /событию программировать события не обязательно.

Предупреждение: Выходы барабана задействованы все время, пока ЦПУ находится в режиме выполнения. Вход Start не должен быть включен. Выходы не отключаются входом Reset. При переходе к режиму выполнения выходы барабана автоматически устанавливаются или сбрасываются в соответствии с конфигурацией текущего шага барабанного командоаппарата. Номер первоначального шага зависит от настройки памяти счетчиков: с запоминанием (retentive) или без запоминания (non-retentive).

Состояние регистров барабанного командоаппарата перед включением питания

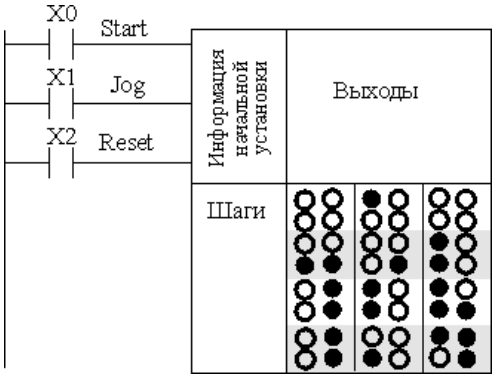
Для приложения важен выбор шага, который будет начальным при включении питания и при переходе из режима программирования к режиму выполнения. Обратимся к следующей таблице. Если память счетчика настроена, как память с запоминанием (см. настройку - retentive range), барабанный командоаппарат сохранит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

Управление барабанным командоаппаратом

Управляющие входы барабанного командоаппарата

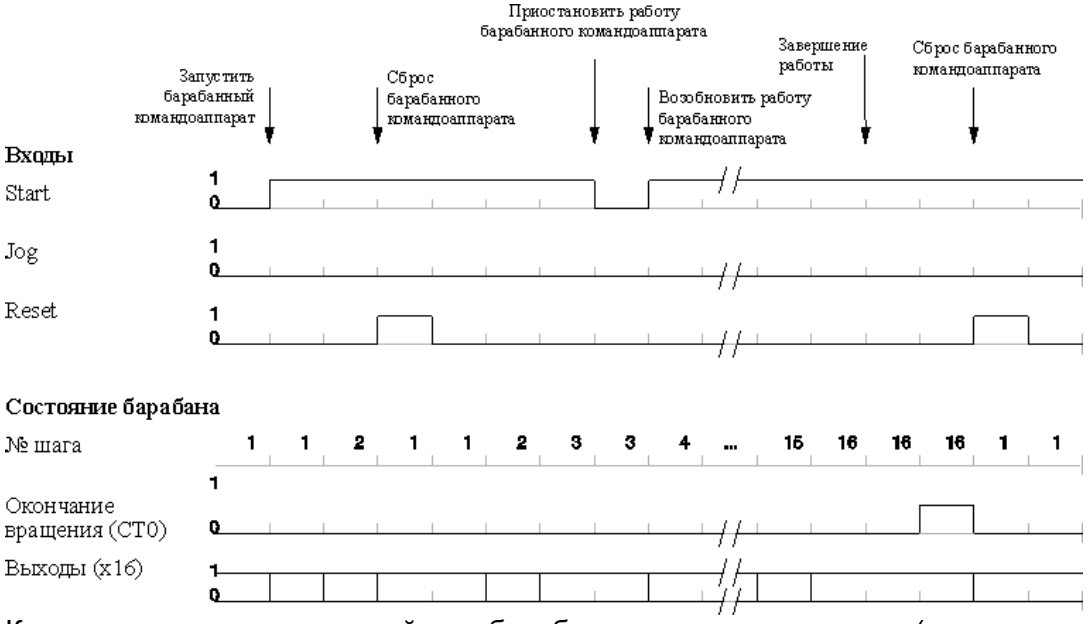


Теперь мы готовы собрать воедино изложенные ранее принципы и продемонстрировать управление с помощью команд барабанного командоаппарата. На рисунке справа показана упрощенная общая команда барабанного командоаппарата. Выходы релейной программы управляют входами Start, Jog и Reset (вход Jog поддерживается только командой EDRUM). Бит первого счетчика барабана (например, CT10) является признаком завершения цикла барабанного 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.

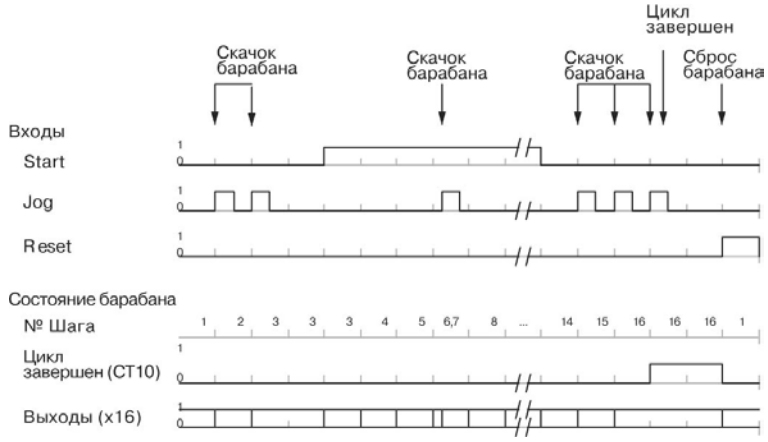
После того, как барабан переходит к Шагу 3, сразу же выключается вход Start, останавливая приращение таймера барабанного командоаппарата до тех пор, пока Start не включится снова.



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.

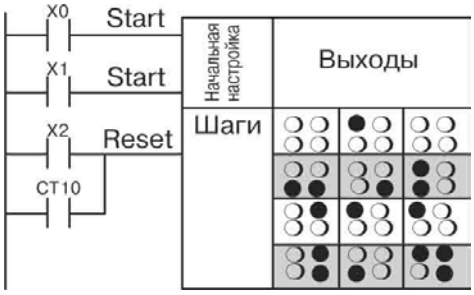
Примечание: на временной диаграмме все шаги имеют одинаковую длительность. На практике длительность шагов может существенно различаться в зависимости от выбранного для них количества тактов.

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

Часто требуется реализация барабанных командоаппаратов, которые автоматически начинают работать снова после завершения цикла. Эта возможность легко реализуется с помощью бита завершения цикла барабанного командоаппарата. В команде барабанного командоаппарата, показанной на рисунке справа, используется CT10, поэтому мы выполняем логическое ИЛИ для бита завершения (CT10) и входа Reset. Когда последний шаг выполнен, командоаппарат устанавливает CT10, возвращая себя к Шагу предварительной установки и сбрасывая одновременно CT10. Контакт X2 по-прежнему используется для ручного сброса.



Инициализация выходов барабана

Выходы барабанного командоаппарата задействованы все время, пока ЦПУ находится в режиме выполнения. При переходе из режима программирования к режиму выполнения барабанный командоаппарат переходит к Шагу предварительной установки и выходы устанавливаются в состояния, соответствующие конфигурации данного шага. Если требуется, чтобы все выходы были отключены при включении питания, необходимо в качестве Шага предварительной установки выбрать "шаг сброса", в котором все выходы будут выключены

Использование переходов по сложным событиям

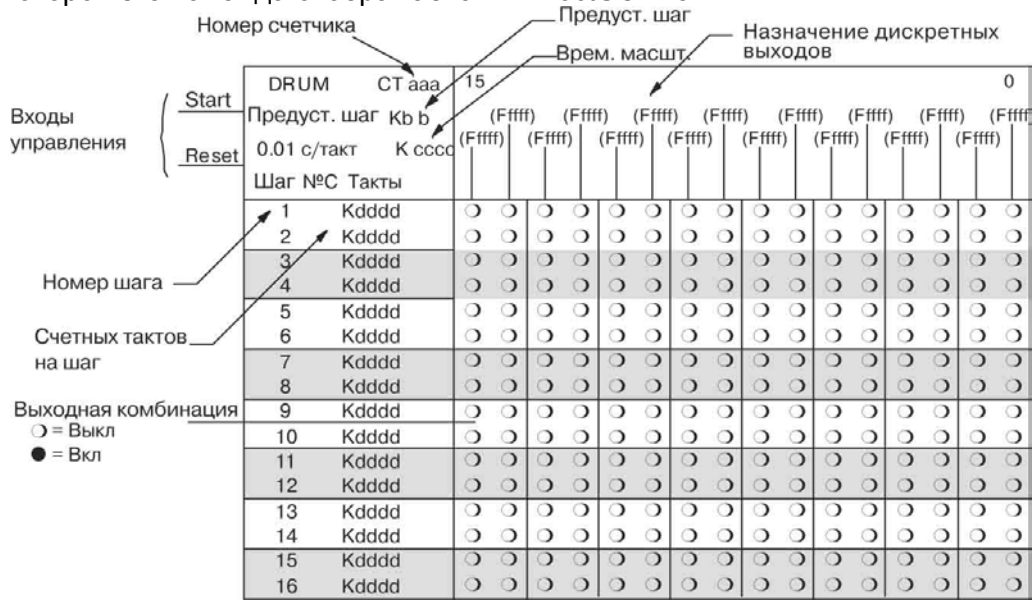
Каждый переход по событию происходит по срабатыванию только одного контакта. Однако это не ограничивает количество событий. Используйте в качестве события перехода шага контакт управляющего реле, например, реле C0. Являясь выходной катушкой в программе релейной логики, C0 может управляться множеством других "событий" (контактов).

Команды барабанного командоаппарата

Команды барабанного командоаппарата DL06 могут быть запрограммированы с помощью *DirectSOFT32* или (только для команды EDRUM) можно воспользоваться ручным программатором (с версией программного обеспечения 2.21 или более поздней). В данном разделе описано использование *DirectSOFT32* для всех команд и мнемоника ручного программатора для команды EDRUM.

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

Барабанный командоаппарат с дискретными выходами и переходом по времени – это основная команда барабанного командоаппарата DL06. Она работает в соответствии с правилами, описанными ранее. Ниже представлена диаграмма, в виде которой эта команда отображается в *DirectSOFT32*.



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 *DirectSOFT32* графически.

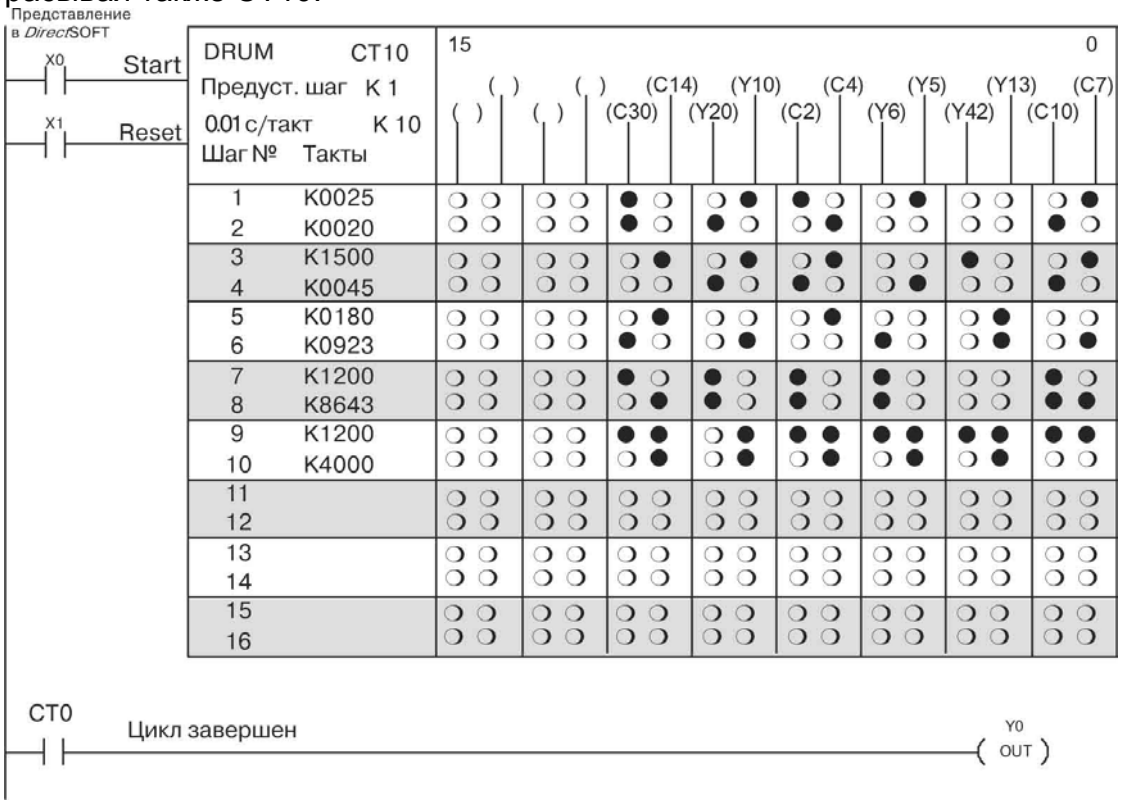
Когда вход Start включен, работает таймер барабанного командоаппарата. Таймер прекращает работу, когда завершается последний шаг или включается вход Reset. При переходе ЦПУ из режима программирования в режим выполнения, а также при включении входа Reset барабан возвращается к выбранному шагу предварительной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

Параметры барабанного командоаппарата	Поле	Типы данных	Диапазон
Номер счетчика	aaa	0 -- 174	
Шаг предварительной установки	bb	K	1 - 16
Временной масштаб	cccc	K	0 - 99.99 секунд
Число тактов на шаг	dddd	K	0 - 9999
Дискретные выходы	Fffff	X, Y, C	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.

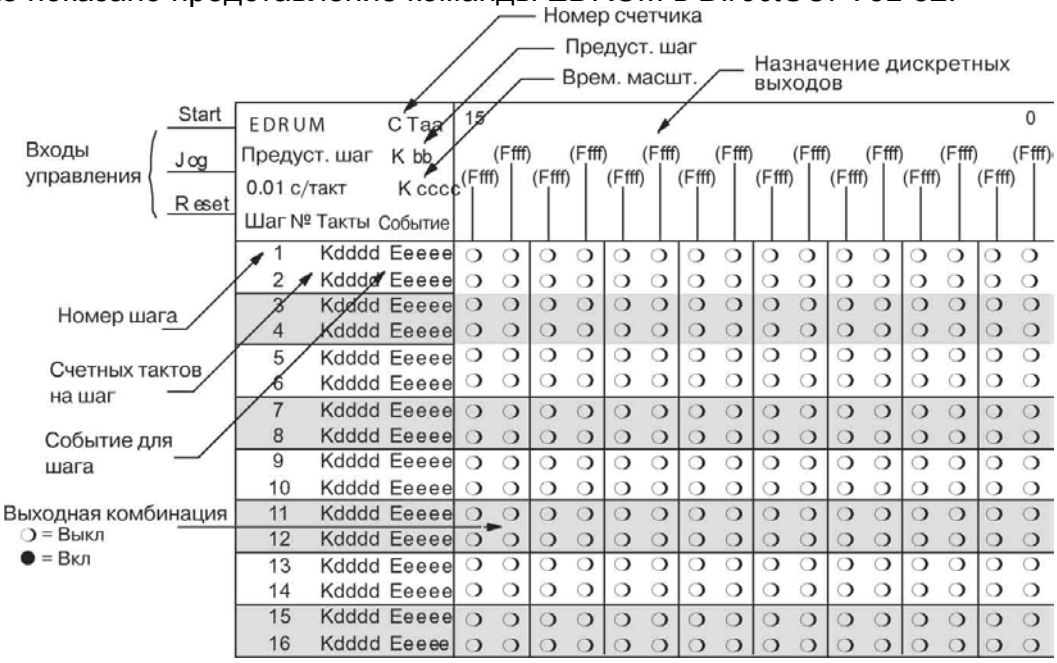
Номер счетчика	Диапазон (n)	Функция	Функция бита счетчика
CTA(n)	0 - 174	Текущее число тактов шага	CT(n) = Цикл завершен
CTA(n+1)	1 - 175	Значение счетчика времени	CT(n+1) = (не используется)
CTA(n+2)	2 - 176	Шаг предварительной установки	CT(n+2) = (не используется)
CTA(n+3)	3 - 177	Текущий шаг	CT(n+3) = (не используется)

На следующем рисунке показано представление типичной релейной программы, использующей команду DRUM, в *DirectSOFT32*. 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 переходами по событиям (EDRUM)

В барабанном командоаппарате EDRUM возможны переходы между шагами по времени и по событию. Он работает в соответствии с общими правилами барабанных командоаппаратов, которые изложены в начале этой главы. Ниже показано представление команды EDRUM в *DirectSOFT32* 32.



Барабанный командоаппарат с переходом по событию обеспечивает 16 шагов и 16 дискретных выходов. 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 задается в импульсах, а в качестве событий указываются дискретные контакты. Поля неиспользуемых шагов и событий должны оставаться незаполненными. Выходные дискретные точки могут присваиваться индивидуально.

Параметры барабанного командоаппарата	Поле	Типы данных	Диапазон
Номер счетчика	aaa	-	0 – 174
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, CT, SP	см. карту памяти
Дискретные выходы	ffff	X, Y, C	см. карту памяти

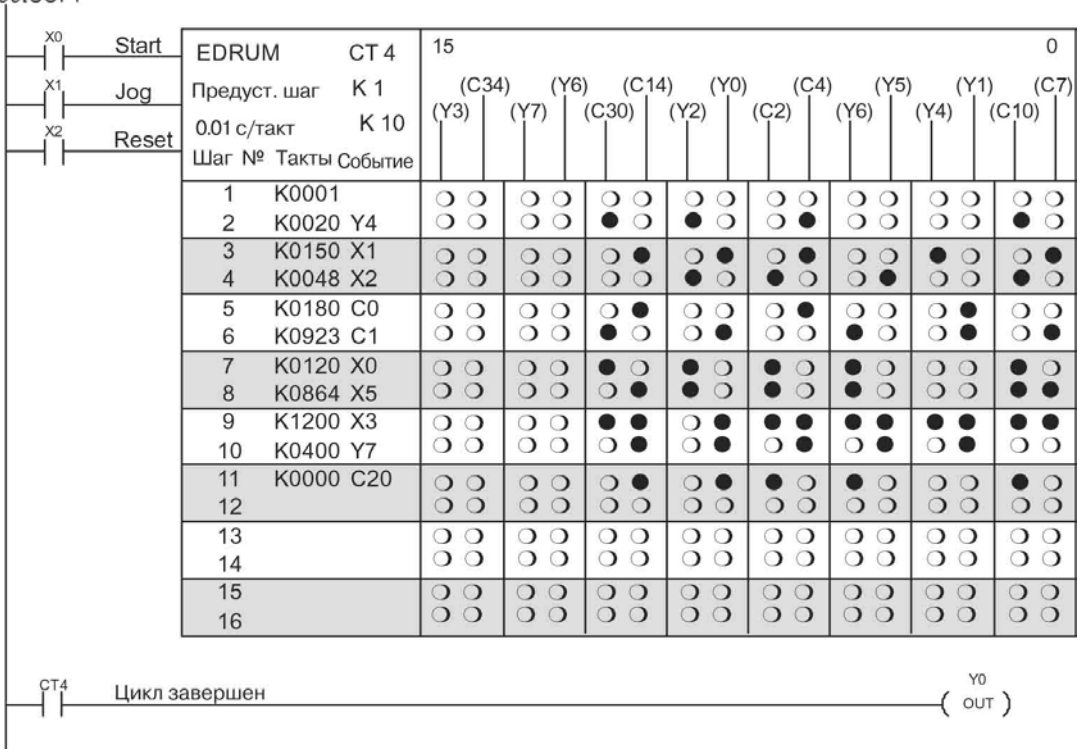
Когда включен вход Start, разрешена работа таймера барабанного командоаппарата. Наращивание таймера происходит тогда, когда выполняется критерий события для текущего шага. Когда количество тактов достигает заданного для данного шага порогового значения, барабан переходит к следующему шагу. Этот процесс прекращается после завершения последнего шага либо по включению входа Reset. При переходе ЦПУ из режима программирования в режим выполнения, а также при включении входа Reset барабанный командоаппарат возвращается к выбранному шагу предварительной установки.

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 СТА(n+2) новый номер шага предварительной установки. Что касается других счетчиков, они служат только для цели контроля.

Номер счетчика	Диапазон (n)	Функция	Функция бита счетчика
СТА(n)	0 - 174	Текущее число тактов шага	СТ(n) = Цикл завершен
СТА(n+1)	1 - 175	Значение счетчика времени	СТ(n+1) =(не используется)
СТА(n+2)	2 - 176	Шаг предварительной установки	СТ(n+2) =(не используется)
СТА(n+3)	3 - 177	Текущий шаг	СТ(n+3) =(не используется)

На следующем рисунке показано представление типичной релейной программы, использующей команду EDRUM, в *DirectSOFT32*. Используются шаги 1…11, задействованы все 16 выходных точек. Шагом предварительной установки является Шаг 1. Длительность одного такта (временной масштаб) = (К10 x 0.01) = 0.1 секунды. Следовательно, длительность Шага 1 составляет (1 x 0.1) равно 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 (СТ4) устанавливает выход Y0 по завершении последнего шага (Шага 11). Барабанный командоаппарат сбрасывается, сбрасывая также СТ4.

Представление в *Direct*SOFT

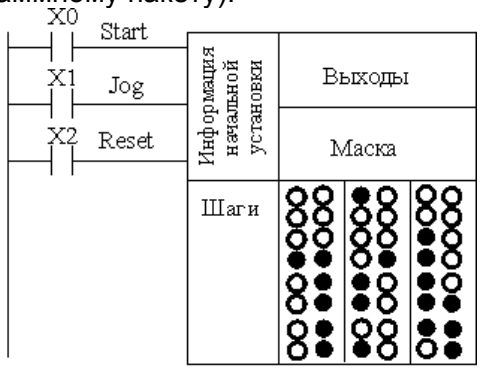


Мнемоника барабанного командоаппарата для ручного программатора

Для программирования команды EDRUM можно использовать как *DirectSOFT32*, так и ручной программатор. В этом разделе описан ввод с помощью ручного программатора (ввод команд барабанного командоаппарата с помощью *DirectSOFT32* описан в руководстве по этому программному пакету).

Сначала в цепях программы релейной логики, управляющих входами барабанного командоаппарата, следует ввести команды Store. В примере справа входы барабанного командоаппарата Start, Jog и Reset управляются элементами X0, X1 и X2 соответственно. Требуемые нажатия клавиш программатора перечислены на мнемонической схеме.

Нажатие этих клавиш предшествует вводу мнемоники EDRUM. Заметим, что цепи релейной программы, управляющие входами Start, Jog и Reset, не обязательно должны содержать только по одному контакту.



Нажатия клавиш ручного программатора

Сохранение X0 | $ STR | → | A 0 | ENT

(Повторите для сохранения X1 и X2)

Нажатие клавиш ручного программатора

EDRUM CNT4 | SHFT | E 4 | D 3 | R ORN | U ISG | M ORST | → | E 4 | ENT

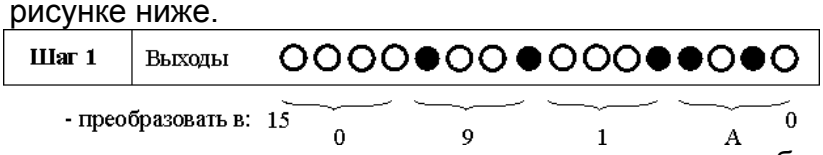
После ввода команд Store введите команду EDRUM (используя счетчик CT0) согласно представленной выше мнемонической схеме.

После ввода мнемоники EDRUM ручной программатор создает форму для ввода всех параметров барабанного командоаппарата. Форма ввода содержит около 50 стандартных (принимаемых по умолчанию) мнемонических обозначений (нажимаемых клавиш ручного программатора), содержащих операторы назначения DEF. Принимаемые по умолчанию мнемонические обозначения – это уже "нажатые" за вас клавиши ручного программатора, и они появляются автоматически. Для перемещения вдоль формы используйте клавиши NXT и PREV. Изменять требуется только принимаемые по умолчанию значения, что существенно сокращает количество нажатий на клавиши. Вводы, которые необходимо сделать для основного барабанного командоаппарата, перечислены в таблице ниже.

Примечание: для выходов и событий по умолчанию введено "DEF 0000", то есть, по умолчанию они не используются. Если ранее назначенный выход требуется сделать неиспользуемым, введите "K0000". В соответствующей строке вновь будет отображено "DEF 0000".

Параметры барабанного командоаппарата	Кол-во входов	Мнемоника/ ввод	Мнемоника по умолчанию	Допустимые типы данных	Диапазон
Вход Start	-	STR (+ цепь)	-	-	-
Вход Jog	-	STR (+ цепь)	-	-	-
Вход Reset	-	STR (+ цепь)	-	-	-
Мнемоника барабанного командоаппарата	-	DRUM CNT aa	-	CT	0 - 174
Шаг предвар. установки	1	bb	DEF K0000	K	1 - 16
Временной масштаб	1	cccc	DEF K0000	K	1 - 9999
Дискретные выходы	16	ffff	DEF 0000	X,Y,C*	См. стр.4-28
тактов на шаг	16	dddd	DEF K0000	K	0-9999
События	16	dddd	DEF K0000	X,Y,C,S,T,CT	См. стр.4-28
Конфигурация выходов	16	gggg	DEF K0000	K	0 - FFFF

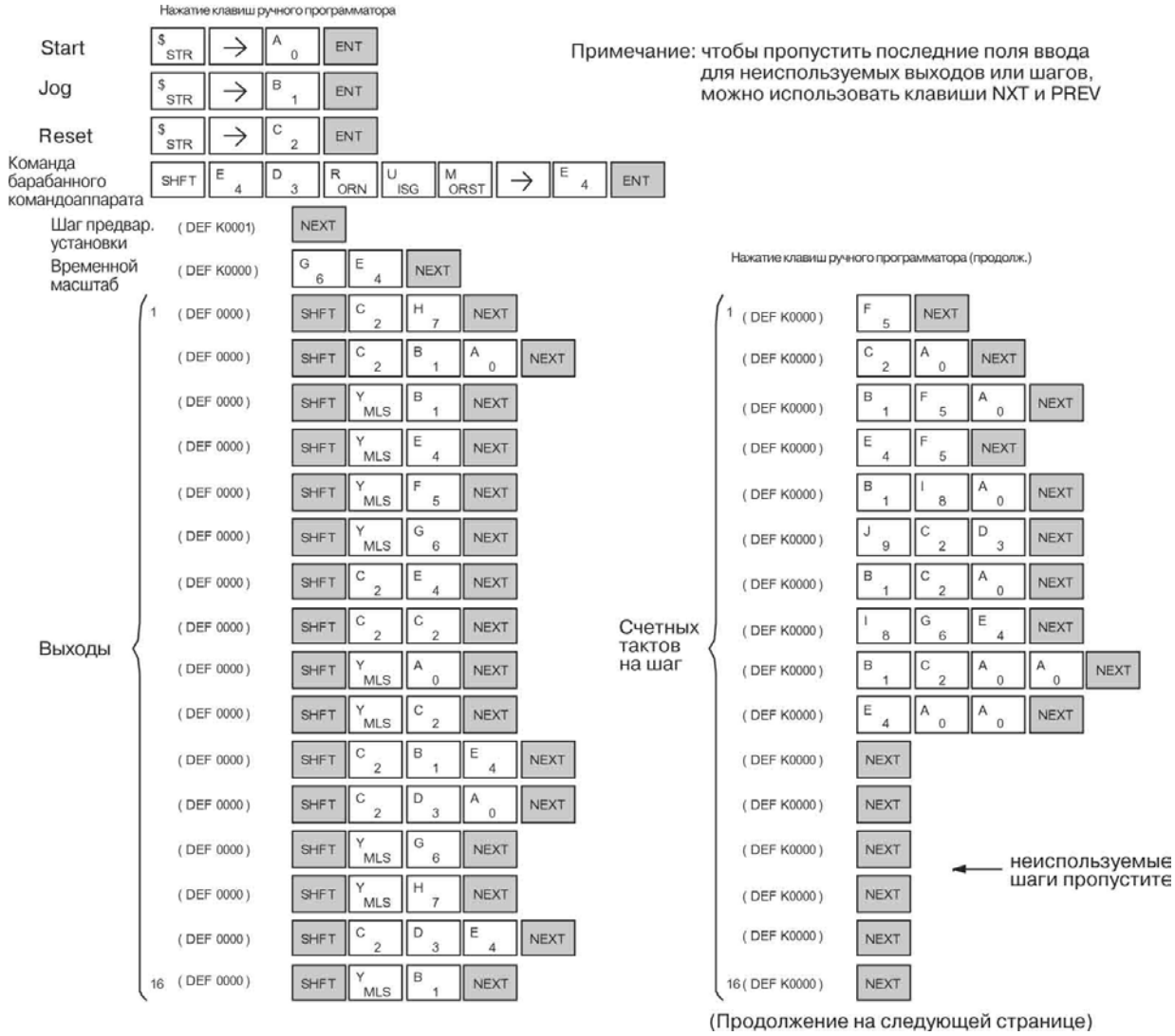
Ранее с помощью диаграммы ввода команды DRUM был показан способ ввода основной команды барабанного командоаппарата с переходом по времени/событию. Сначала мы преобразуем комбинацию выходных состояний для каждого шага в эквивалентное шестнадцатеричное число, как показано на рисунке ниже.

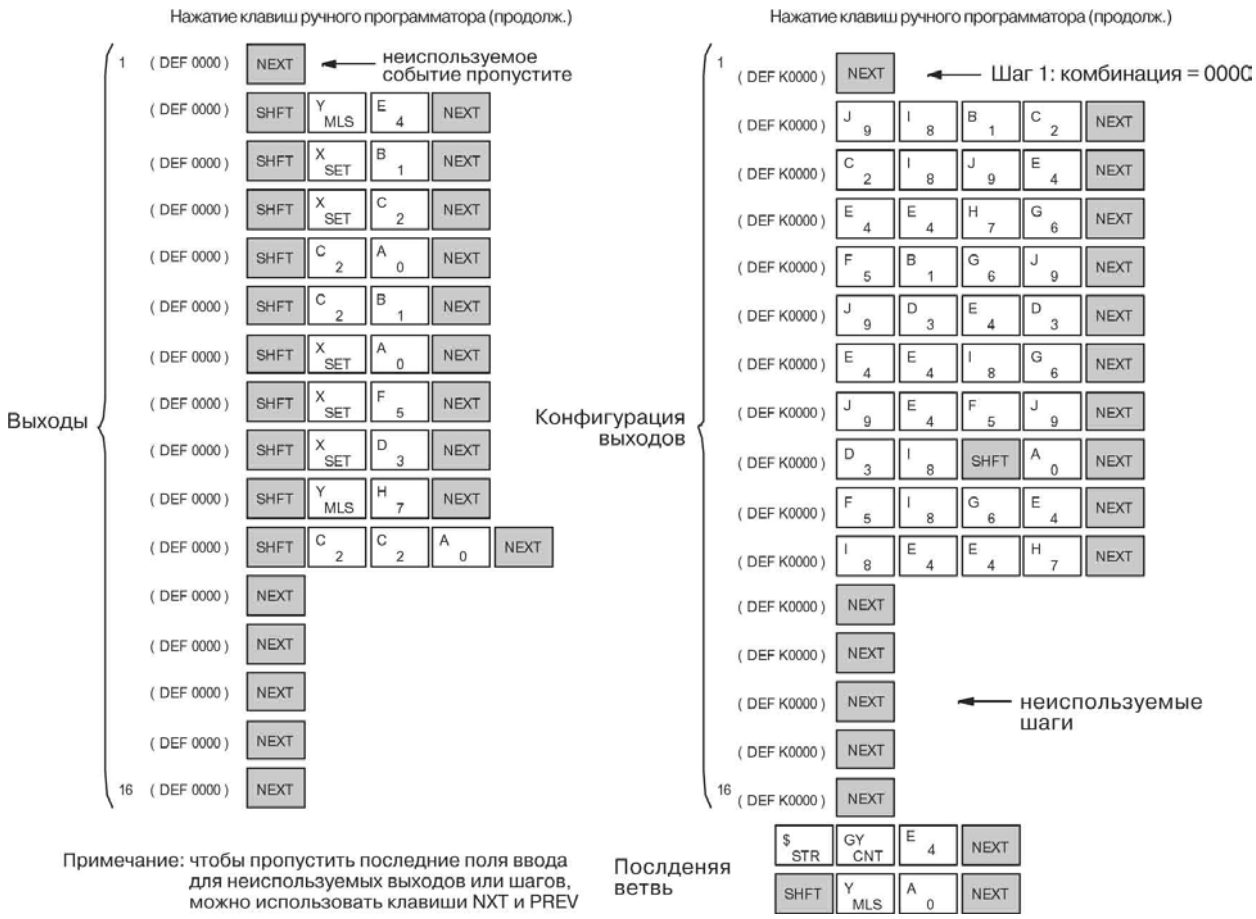


На следующем рисунке перечислены нажатия клавиш, необходимые для ввода предыдущего примера команды EDRUM с помощью ручного программатора. Поля формы, соответствующие стандартным вводам, заключены в скобки. После ввода команды барабанного командоаппарата (в четвертой строке) последующие нажатия на клавиши перезаписывают цифровую часть каждого оператора назначения DEF.

ПРИМЕЧАНИЕ: для редактирования барабанного командоаппарата требуется ручной программатор с прошивкой ПЗУ версии 2.21 или более поздней.

ПРИМЕЧАНИЕ: чтобы пропустить последние поля ввода для неиспользуемых выходов или шагов, можно использовать клавиши NXT и PREV.





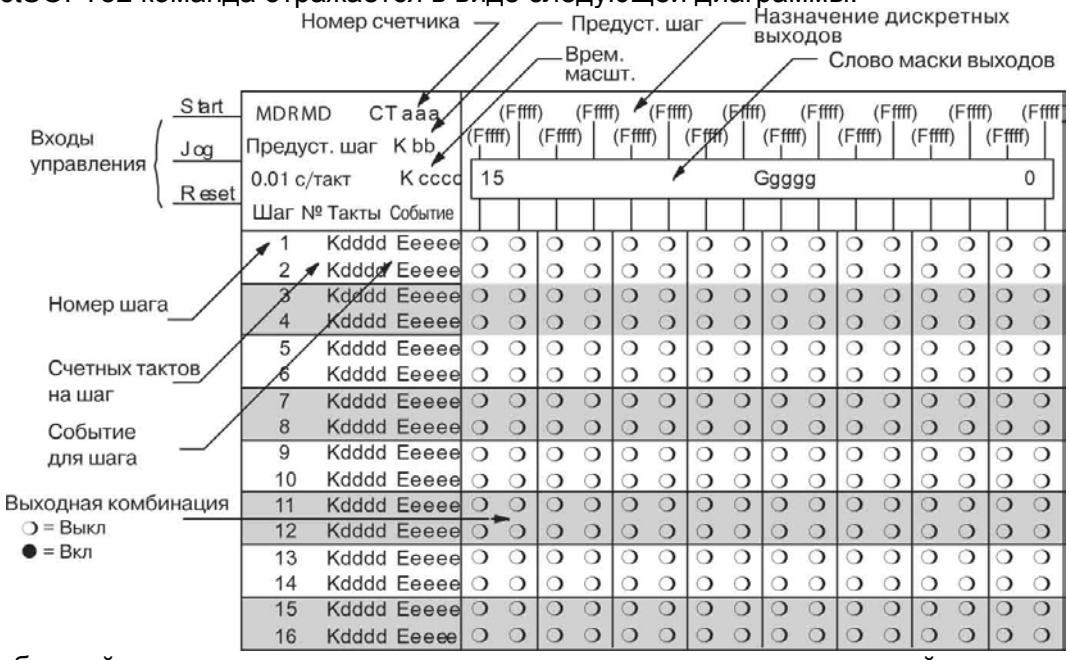
Примечание: чтобы пропустить последние поля ввода для неиспользуемых выходов или шагов, можно использовать клавиши NXT и PREV.

Примечание: при использовании команды EDRUM проще работать не с ручным программатором, а с пакетом *DirectSOFT32*

.

Барабанный командоаппарат с дискретными выходами и маской выходов, с переходами шагов по событию (MDRMD)

Барабанный командоаппарат с дискретными выходами, маской выходов и с переходами 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 DirectSOFT32 команда отражается в виде следующей диаграммы.



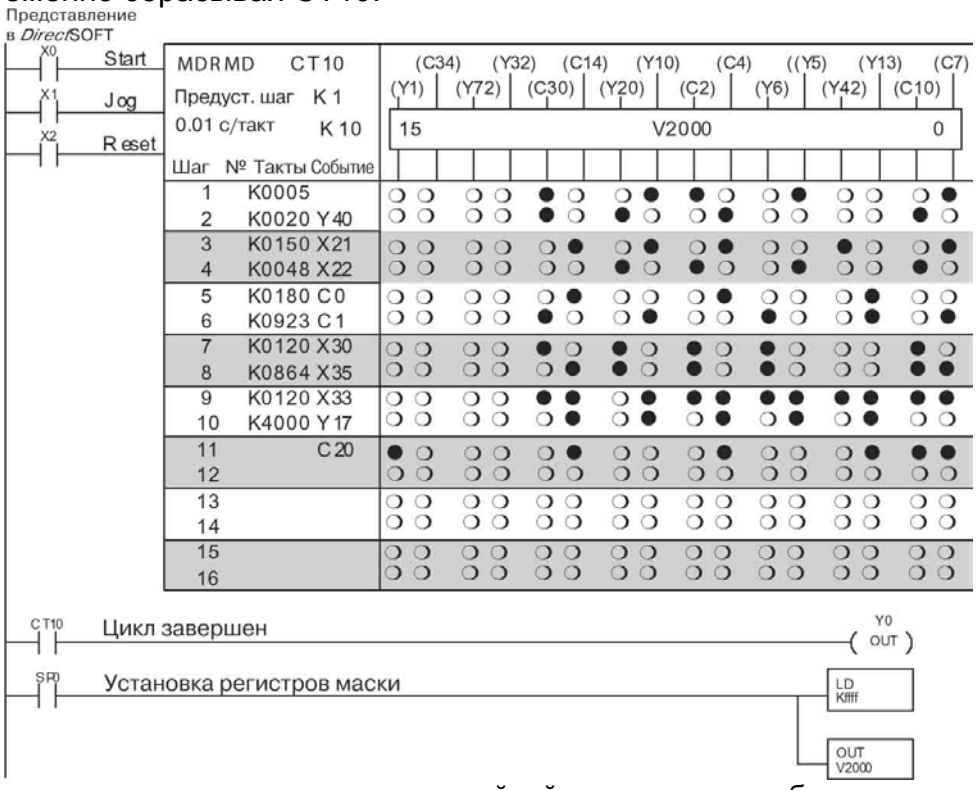
Барабанный командоаппарат с дискретными выходами и маской выходов, с переходом между шагами по событию обеспечивает 16 шагов и 16 выходов. На каждом шаге выполняется побитовое логическое "И" выходов барабана и слова маски выходов. Позиция первого из 16-ти слов маски указывается в поле Ggggg. Переходы 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). 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	Поле	Типы данных	Диапазон
Номер счетчика	aaa	-	0 - 174
Шаг предварительной установки	bb	K	1 - 16
Временной масштаб	cccc	K	0 - 99.99 с
Число тактов на шаг	dddd	K	0 - 9999
Событие	eeee	X, Y, C, S, T, ST, GX, GY. CT, SP	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. Что касается других счетчиков, они служат только для целей контроля.

Номер счетчика	Диапазон (n)	Функция	Функция бита счетчика
CTA(n)	0 - 174	Текушее число тактов шага	CT(n) = Цикл завершен
CTA(n+1)	1 - 175	Значение счетчика времени	CT(n+1) = (не используется)
CTA(n+2)	2 - 176	Шаг предварительной установки	CT(n+2) = (не используется)
CTA(n+3)	3 - 177	Текущий шаг	CT(n+3) = (не используется)

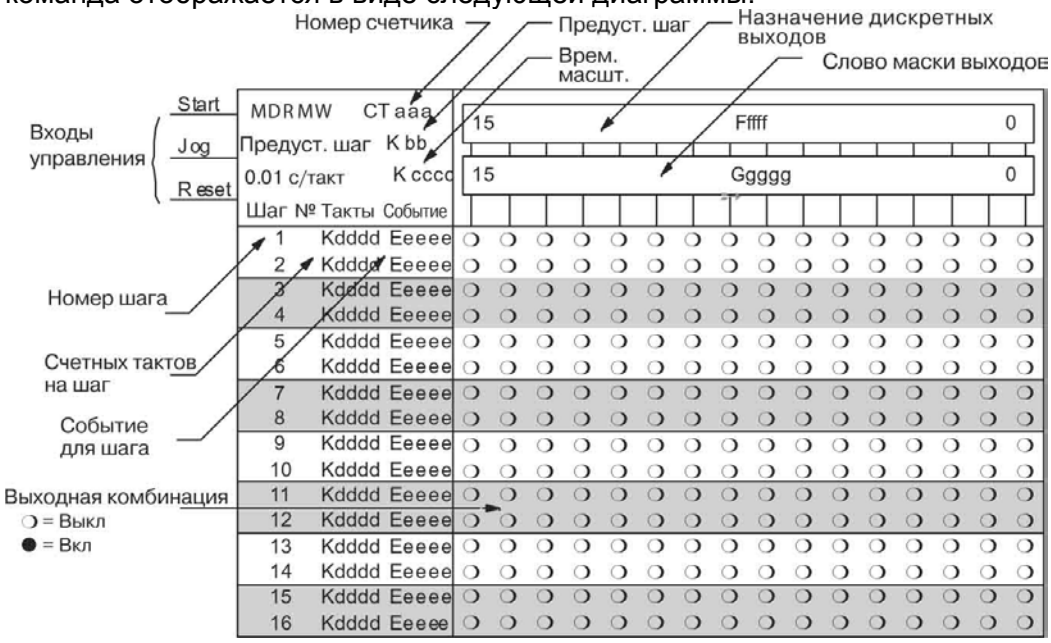
На следующем рисунке показано представление типичной программы релейной логики, использующей команду MDRMD, в *DirectSOFT32*. Используются шаги 1…11 и все 16 выходных точек. Словом маски выходов является V2000. Конечное состояние выходов барабанного командоаппарата показано над словом маски. Биты слова 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.



Примечание: в программе релейной логики должна быть предусмотрена запись констант в слова V2000…V2012, чтобы были охвачены все регистры масок для одиннадцати шагов, использующихся в барабанном командоаппарате.

Барабанный командоаппарат с выходным словом и с маской выхода, с переходом по событию (MDRMW)

В барабанном командоаппарате с выходным словом и с маской выходов, с переходом шагов 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 *DirectSOFT32* эта команда отображается в виде следующей диаграммы.



Барабанный командоаппарат с выходным словом, маской выходов и с переходами шагов по событию 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).

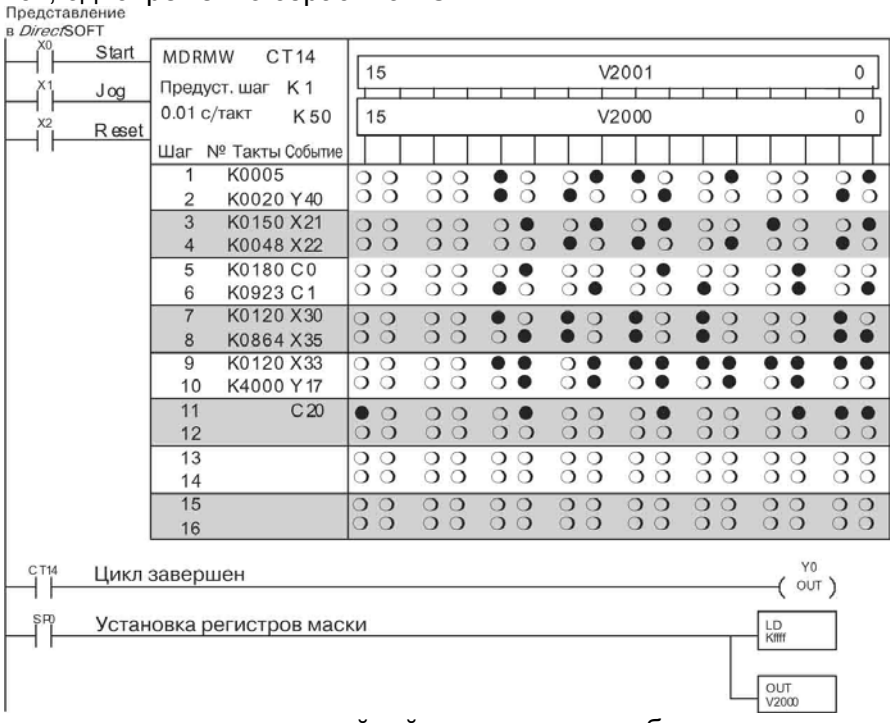
Когда вход Start включен, разрешается работа счетчика барабанного командоаппарата. Пока критерий события для текущего шага выполняется, происходит наращивание таймера. Когда количество тактов достигает порогового значения для данного шага, барабанный командоаппарат переходит к следующему шагу. Процесс 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	Поле	Типы данных	Диапазон
Номер счетчика	aaa	-	0 - 174
Шаг предварительной установки	bb	K	1 - 16
Временной масштаб	cccc	K	0 - 99.99 секунд
Число тактов на шаг	dddd	K	0 - 9999
Событие	eeee	X, Y, C, S, T, ST, GX, GY, SP	см. карту памяти
Выходное слово	Fffff	V	
Маска выхода	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. Что касается других счетчиков, они служат только для целей контроля.

Номер счетчика	Диапазон (n)	Функция	Функция бита счетчика
CTA(n)	0 - 174	Текущее число тактов шага	CT(n) = Цикл завершен
CTA(n+1)	1 - 175	Значение счетчика времени	CT(n+1) = (не используется)
CTA(n+2)	2 - 176	Шаг предварительной установки	CT(n+2) = (не используется)
CTA(n+3)	3 - 177	Текущий шаг	CT(n+3) = (не используется)

На следующем рисунке показано представление типичной программы релейной логики, использующей команду MDRMW, в *DirectSOFT32*. 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 (K50 x 0.01) =0.5 секунды. Следовательно, длительность Шага 1 = (5 x 0.5) =2.5 секунды. 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.



Примечание:в программе релейной логики должна быть предусмотрена запись констант в слова V2000…V2012, чтобы были охвачены все регистры масок для одиннадцати шагов, использующихся в барабанном командоаппарате.

Глава 7

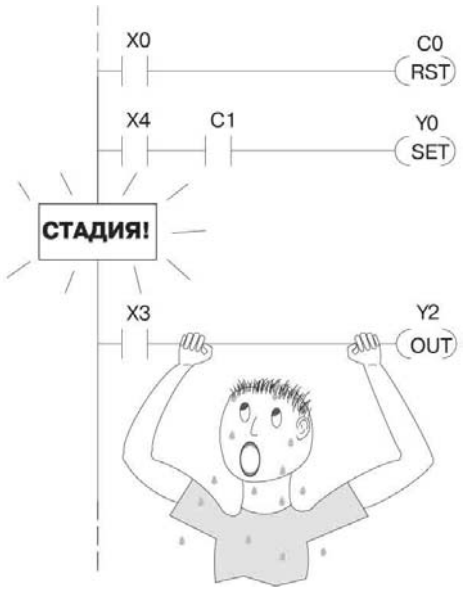
7. Стадийное программирование RLL^{PLUS}

В данной главе…

Введение в стадийное программирование

По сравнению с решениями, использующими язык релейной логики (RRL) в чистом виде, стадийное программирование предоставляет более легкий способ организации и построения программ для решения сложных прикладных задач. Стадийное программирование не заменяет и не отвергает использование традиционных программ на языке релейной логики. Именно поэтому стадийное программирование называют RLL*plus* . Полученные ранее знания, а также предыдущий опыт не окажутся лишними. 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.

Преодоление "боязни стадий"



Программисты, работающие с промышленными ПЛК, 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:

С большими программами практически невозможно управляться из-за отсутствия в них какой-либо структуры.

Когда в процессе наступает сбой, довольно сложно обнаружить цепь, в которой произошла ошибка.

В дальнейшем программу очень сложно модифицировать, поскольку ее структура имеет другую логику, отличную от логики решаемой прикладной задачи.

Нетрудно заметить, что эти недостатки влекут затрату дополнительного времени, а время, как известно, это деньги. *Эти преграды позволяет преодолеть стадийное программирование!* Мы уверены, что время, затраченное на изучение принципов стадийного программирования, позволит программисту ПЛК существенно повысить продуктивность и эффективность процесса написания программ!

Таким образом, мы призываем читателя к изучению техники стадийного программирования с тем, чтобы пополнить ею "инструментарий", используемый для создания программ в области автоматизации. Эта глава построена в виде самоучителя по стадийному программированию. Для достижения результатов:

Начните с самого начала и не пропускайте ни одного раздела.

Изучите каждый принцип стадийного программирования, проработав каждый пример. Каждый последующий пример строится на основе предыдущих примеров.

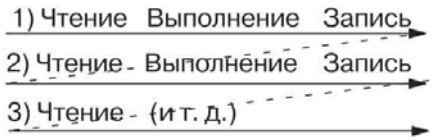
Прочитайте раздел "Стадийное программирование в вопросах и ответах", содержащий краткий обзор изложенного материала.

Учимся рисовать диаграммы переходов состояний

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:

1. Чтение входов.
2. Исполнение релейной программы.
3. 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!* Мы вскоре убедимся, что если картина процесса была воссоздана правильно, **также будет правильной и создаваемая программа**!

Процесс с двумя состояниями

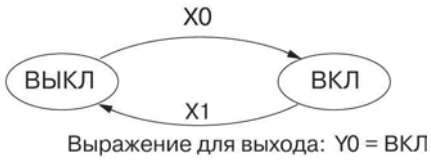
На рисунке справа показан простой процесс, заключающийся в управлении промышленным двигателем. Для включения двигателя будем использовать зеленую кнопку без фиксации, а для отключения - красную кнопку. Управляя установкой, оператор нажимает на соответствующую кнопку. Управление длится около секунды.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

На следующем этапе мы рисуем диаграмму переходов состояний, показанную на рисунке справа. 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!*

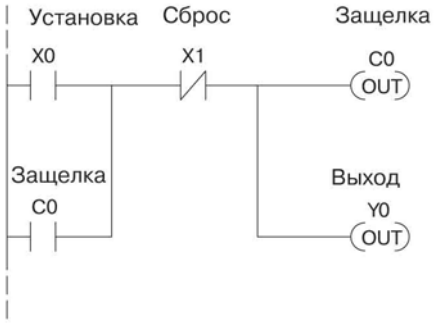


Выражение для выхода: Y0 = ВКЛ

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

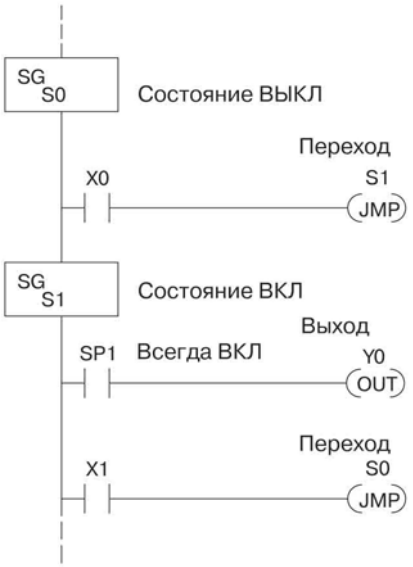
Решение в виде RLL-программы показано справа. В состав программы входит самоудерживающееся реле управления C0. Когда нажата кнопка включения (X0), возбуждается выходная обмотка C0, и контакт C0 во второй цепи переходит во включенное состояние с самоудерживанием. Итак, X0 **переводит защелку C0 во включенное 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.

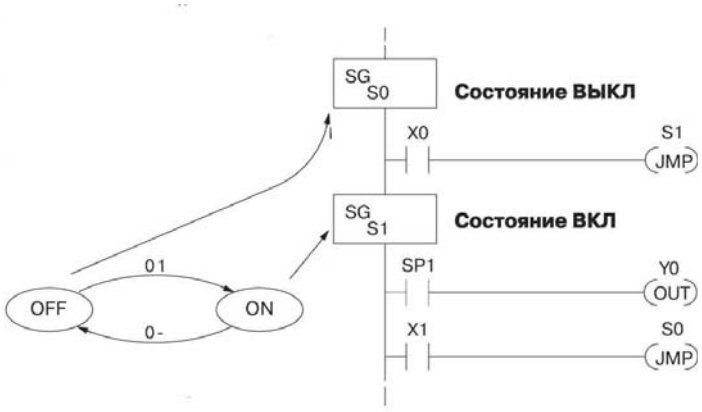
Мы хотим, чтобы в состоянии включения (S1) двигатель работал. Поскольку специальный контакт реле SP1 всегда включен, Y0 включит двигатель.

По нажатию кнопки выключения (X1) происходит возврат к состоянию выключения. 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

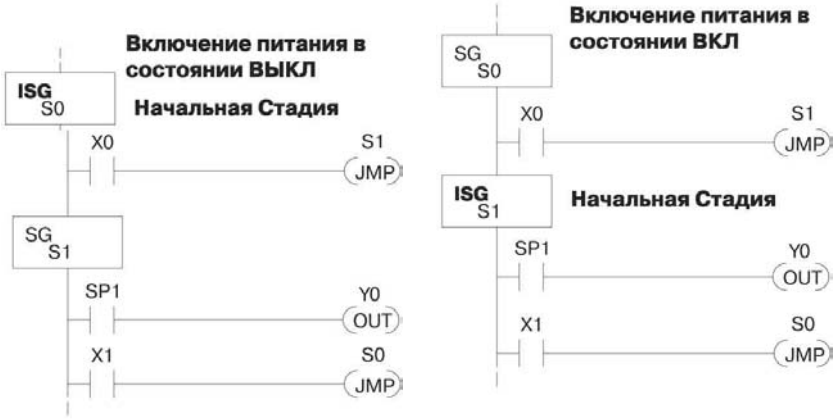
К настоящему моменту читатель, возможно, думает: "Не вижу никаких особых преимуществ стадийного программирования. Напротив, стадийная программа занимает больше места по сравнению с простой RLL-программой". Что ж, пришло время испытать свою веру. С возрастанием сложности задач управления стадийное программирование 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. А теперь попробуйте так же просто идентифицировать те же состояния в программе на языке RLL на предыдущей странице!



Начальные Стадии

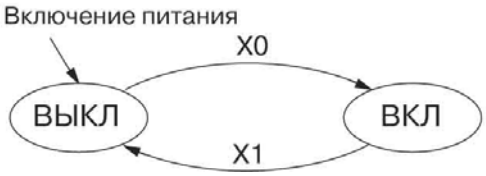
При включении питания и при переходе программы в режим исполнения ПЛК всегда начинает работу с отключенными Стадиями рабочего режима (SG). Поэтому исполнение показанных Стадийных программ, на самом деле, начаться не могло (поскольку цепи не опрашиваются, если их Стадии не активны).



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, что позволяет 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 не активен в определенный момент времени. Все Биты Включения Стадий (S0…1777) размещаются в регистре образа 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, принадлежащие ему цепи не опрашиваются (не выполняются).

Если Бит Стадии S0 = 1, принадлежащие ему 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.

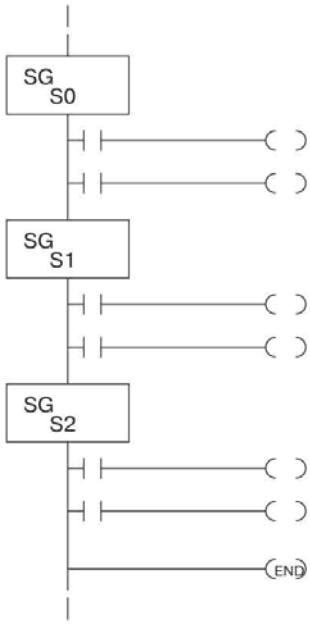
Восьмеричная нумерация - Для нумерации Стадий, как и для точек ввода/вывода, используется восьмеричная система. Поэтому обозначение "S8" не допускается.

Общее количество Стадий - DL06 поддерживает до 1024 Стадий (S0…1777 в восьмеричной системе).

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.

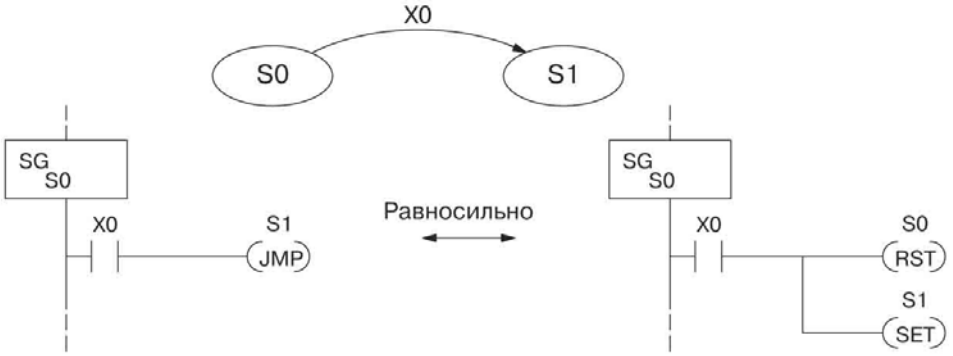
Последняя Стадия - Последняя Стадия релейной программы охватывает все цепи, расположенные под ее обозначением, до завершающей обмотки (END).



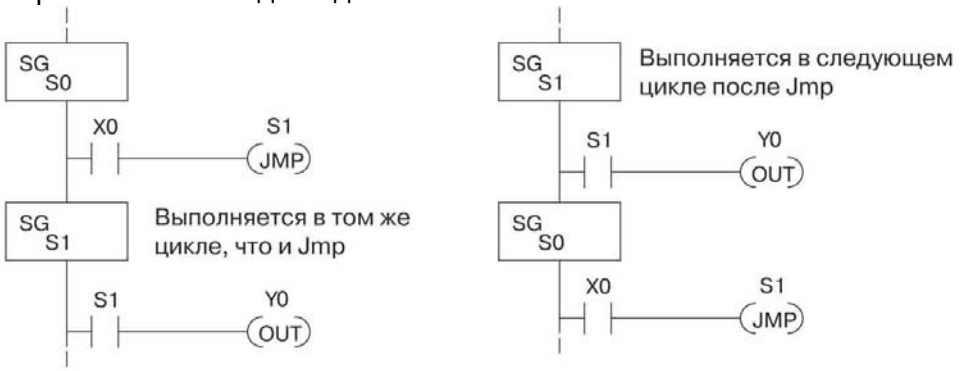
Использование команды Stage Jump для переходов состояний

Команды 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.



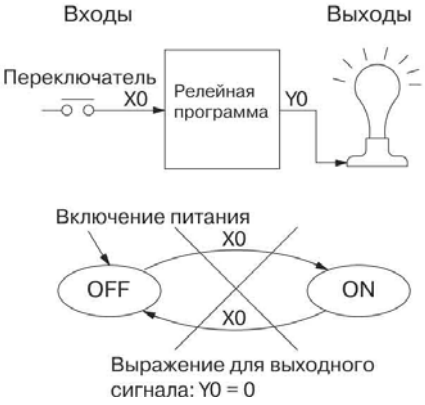
Прочитайте, пожалуйста, внимательно - Можно легко прийти к неправильному истолкованию команды перехода. При выполнении команды перехода "переход" не происходит мгновенно, как по команде управления программой, например, GOTO или GOSUB. Работает она следующим образом:
Команда перехода сбрасывает Бит Включения 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.



Примечание: Полагаем, что в обоих примерах в начале выполнения Стадия 0 активна, а Стадия 1 - не активна.

Пример стадийной программы: включение/ выключение лампы с помощью контроллера

Процесс с четырьмя состояниями

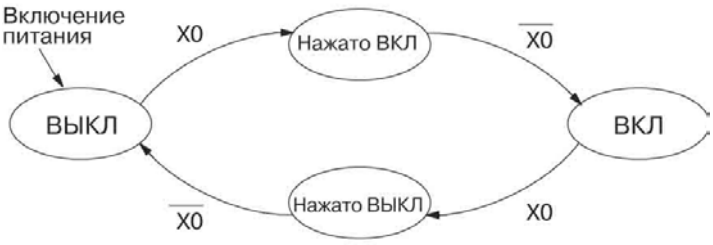


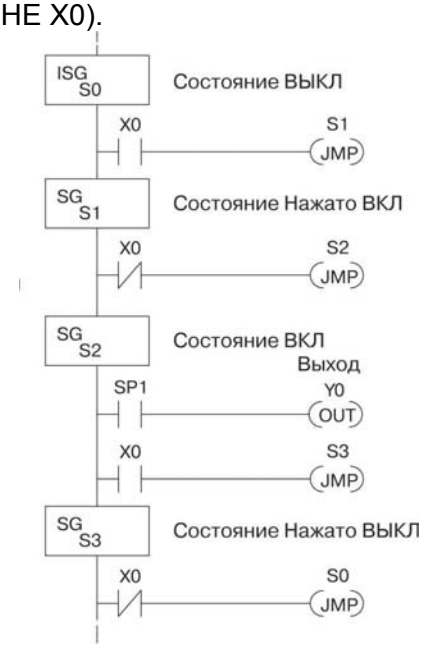
На рисунке справа показан процесс, в котором для управления лампочкой используется одиночная кнопка без фиксации. В релейной программе на входе переключения будет предусмотрен триггер (защелка). Чтобы включить лампу, следует нажать и отпустить кнопку, чтобы выключить - вновь нажать и отпустить кнопку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

Первое, что напрашивается, это использовать X0 для обоих переходов (как на примере, показанном справа). Тем не менее, *это неправильно* (читайте 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 Первичной 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 переходов 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. Помните о восьмеричной нумерации Стадий. В DL06 возможно создать до 1024 Стадий, которые нумеруются от 0 до 1777 в восьмеричной системе.
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 стадийныее программы, начиная со словесного описания задачи.

Пример стадийной программы: управление дверью гаража

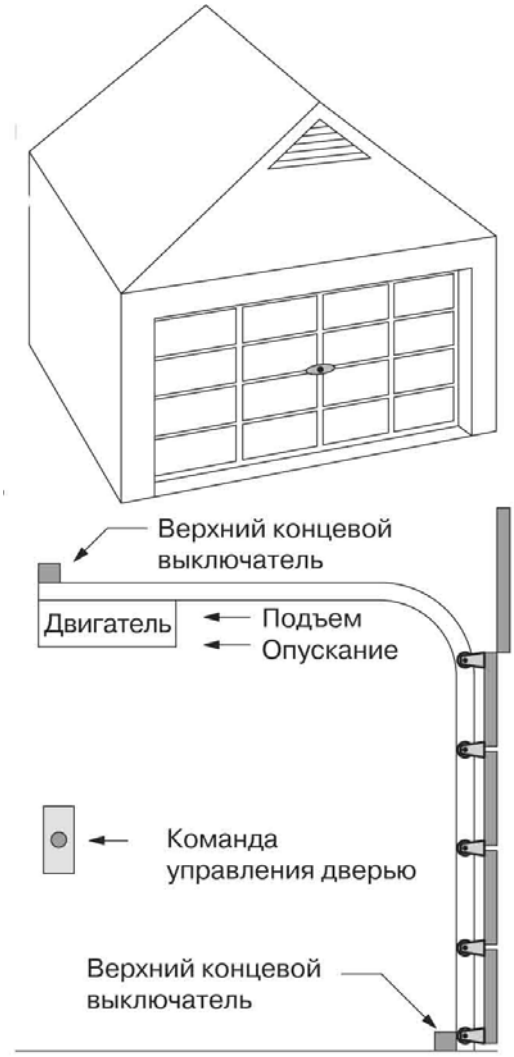
Пример управления дверью гаража

В нашем следующем примере программирования состояний (создания стадийных программ) мы создадим контроллер 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 нефиксирующуюся 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.

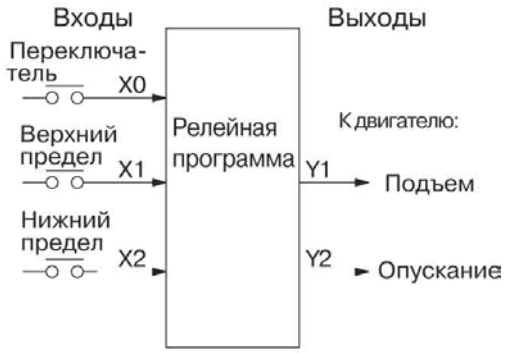
Двигатель имеет два входа управления: подъем и опускание. Когда ни один из входов не активен, двигатель остановлен. Командой на перемещение двери является простое нажатие на кнопку. Вмонтирована ли кнопка в стену, как показано на рисунке, или это кнопка пульта дистанционного управления, в любом случае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



Нарисуем 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.

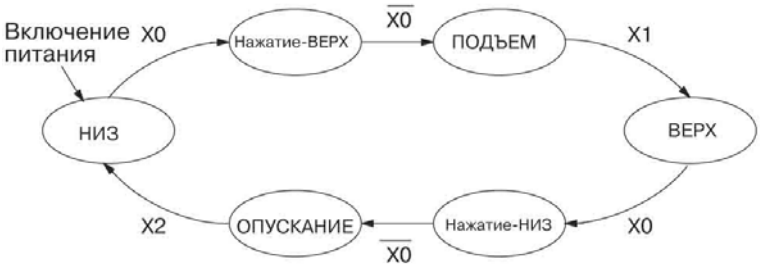
В контроллере предусмотрено два выхода для управления двигателем. Y1 служит командой подъема, а Y2 - командой опускания двери.



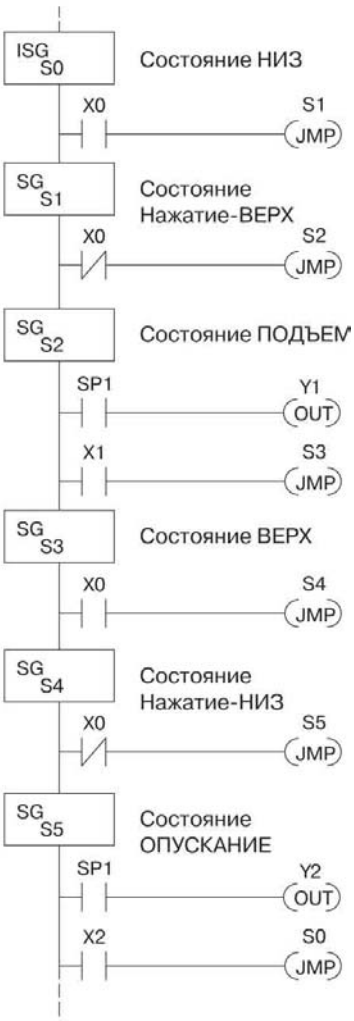
Рисуем диаграмму состояний

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.

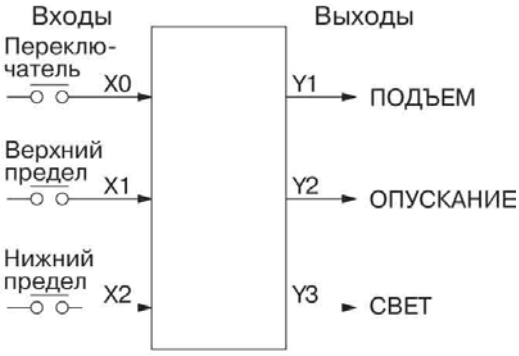
Добавляем сигнальную лампу

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 добавим на структурной схеме контроллера выход управления лампой Y3 (см. рисунок справа).

На рисунке, показанном снизу, добавим состояние, которое назовем "СВЕТ". Всякий раз, когда владелец гаража нажимает кнопку управления дверью и отпускает ее, активно состояние ПОДЪЕМ или ОПУСКАНИЕ, *и одновременно активно состояние СВЕТ.* Переход к состоянию СВЕТ показан штриховой линией, поскольку он не является основным переходом.

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!

Выражения для выходов: Y1 = ПОДЪЕМ
Y2 = ОПУСКАНИЕ
Y3 = СВЕТ

Использование таймера внутри Стадии

Конечная 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 (программы переключения состояний), зададутся вопросом, куда разместить Стадию СВЕТ и какой номер ей присвоить. К их великой радости это не имеет никакого значения!

Просто присвойте новой Стадии еще неиспользуемый номер.

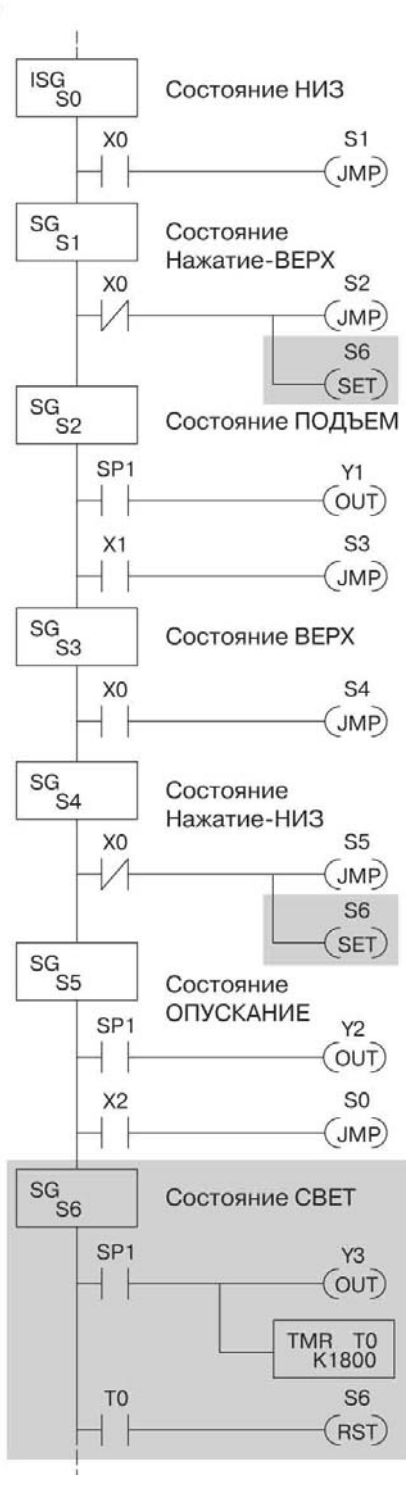
Место размещения в программе не имеет значения, поэтому разместим команду в конце.

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.**

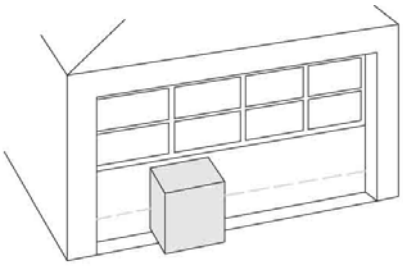
Пока Стадия S6 остается активной, таймер продолжает отсчет времени. По завершении счета устанавливается соответствующий бит таймера T0. Таким образом, спустя три минуты устанавливается T0, и команда сброса S6 (Reset S6) делает Стадию неактивной.

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

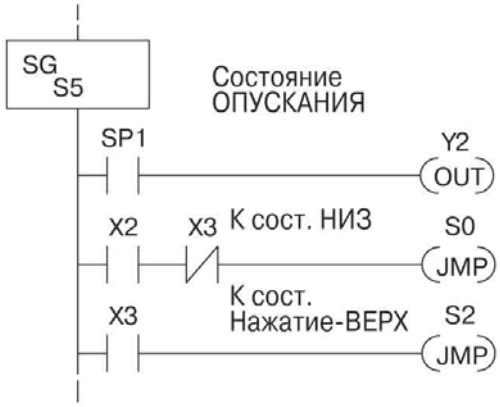
Некоторые современные устройства открывания гаражной двери способны обнаружить объект, находящийся под дверью. Опускающаяся дверь, снабженная, как правило, фотодатчиком ("электронным глазом"), остановится и начнет подниматься. Запрограммируем работу функции защиты именно таким образом, добавив в структурную 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.

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.

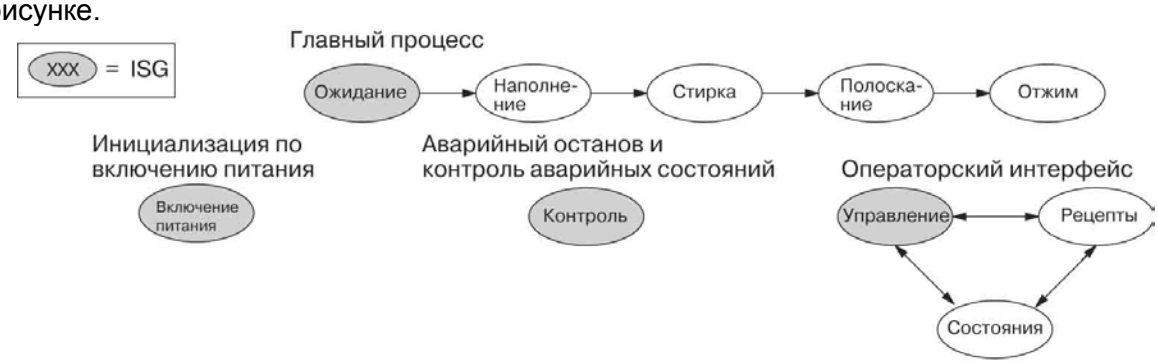


Правила создания стадийных программ

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 течении 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.

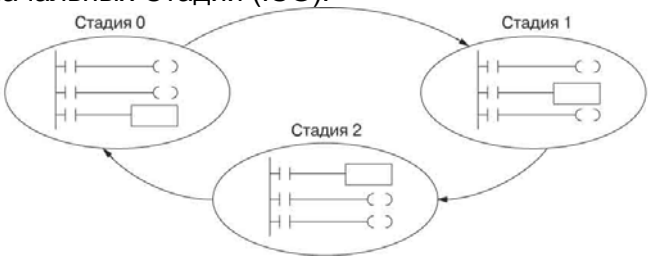
Аварийный останов и контроль аварийных состояний - Данная Стадия всегда активна, поскольку она следит за ошибками, которые являются признаками аварийных состояний или требуют аварийного останова. Как правило, данная Стадия приводит к сбросу Стадий главного процесса или где-либо еще с целью их инициализации 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 (т.е., установить Бит Стадии) 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 одночного импульса (PD) - Будьте внимательны при использовании в Стадиях обмотки реле одночного 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 одночного импульса на сработает, поскольку переход из 0 в 1 уже произошел.

Альтернатива обмотке одночного 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.

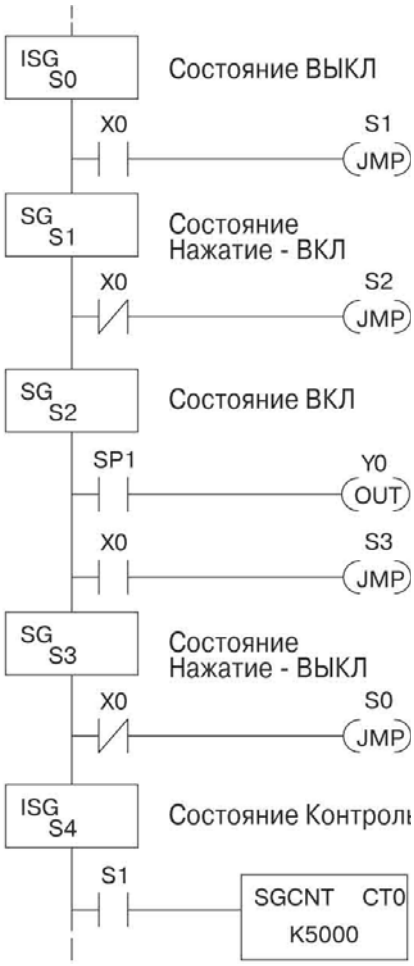
На обычный счетчик, использующийся внутри Стадии, накладываются ограничения: он может не сброситься из других Стадий с помощью команды RST для выходного бита счетчика. Эта проблема, впрочем, устраняется за счет применения специального счетчика.

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!*

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

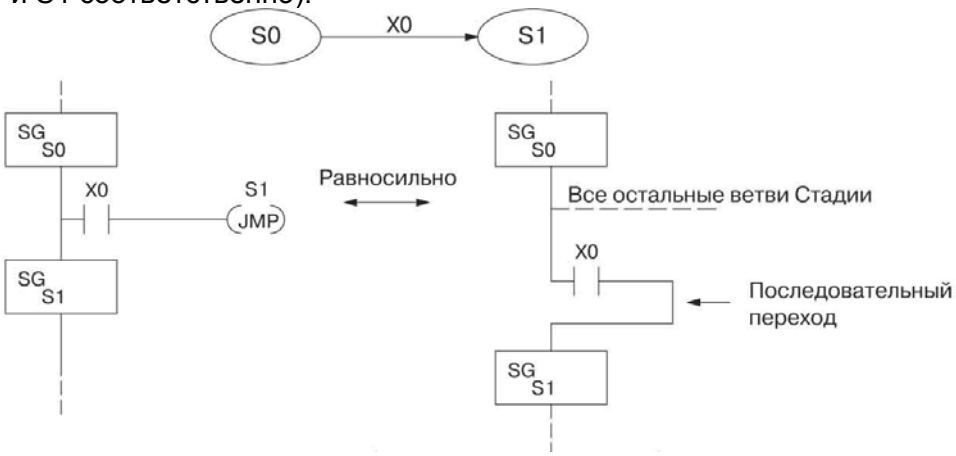
Счетчик для стадийной программы

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.

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 *DirectSOFT32* 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).

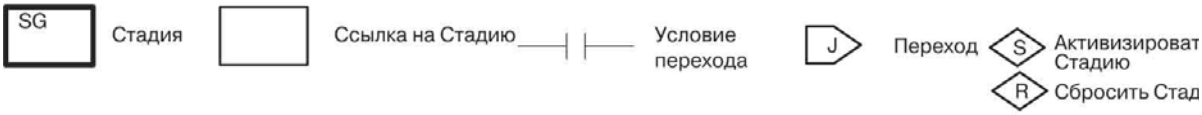


Вспомним, что команда Stage JMP может быть расположена в любом месте текущей Стадии, но результат будет один и тот же. В то же время, переход состояния методом последовательного перехода (показанный выше), должен быть реализован в последней цепи Стадии. Все остальные цепи Стадии будут ему предшествовать.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

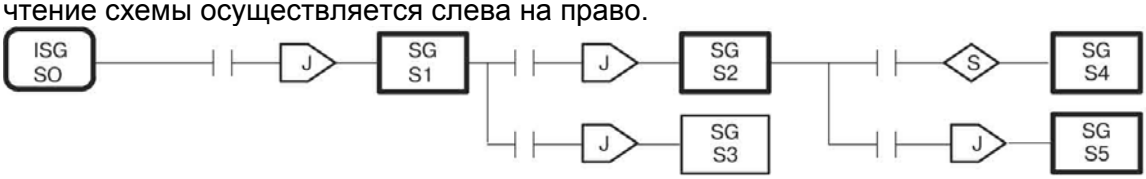
При использовании метода последовательного перехода остается только одна команда Stage JMP, и в этом преимущество данного метода. В то же время, вносить в программу изменения сложнее, чем при использовании Stage JMP, поэтому большинству программистов для реализации переходов состояний рекомендуется использовать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

Вид стадийной программы в пакете DirectSOFT32

Функция Stage View (Вид стадийной программы) в DirectSOFT32 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.



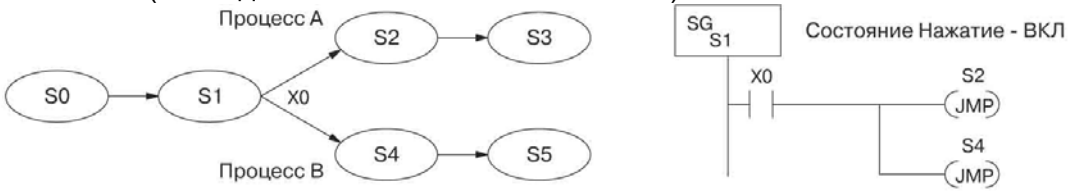
На следующем рисунке показано типичное представление релейной программы, содержащей Стадии, в виде последовательной блок-схемы. Обратите внимание, что чтение схемы осуществляется слева на право.



Принципы параллельного выполнения процессов

Параллельные процессы

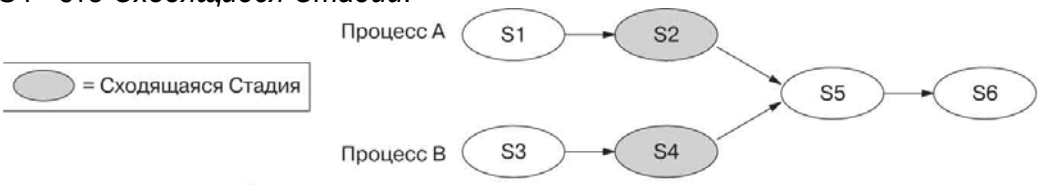
В предыдущих разделах данной главы мы рассматривали ситуацию, когда переход из текущего состояния осуществлялся либо к одному, либо к другому состоянию, называя такие переходы *взаимоисключающими*. В ряде случаев может потребоваться разветвление на два или более параллельных процесса, которые должны 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 7-7). 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*.



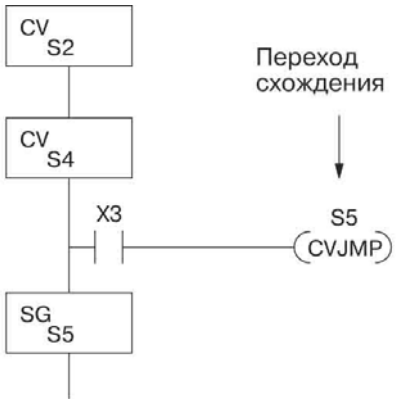
Сходящиеся Стадии (CV)

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)

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 устанавливается 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 перечислены обобщенные требования к использованию Сходящихся Стадий, а также некоторые подсказки по их 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.
- В группу можно объединить до 16-ти Сходящихся Стадий. Другими словами, в одну Стадию можно свести не более 16-ти 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 вводится 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

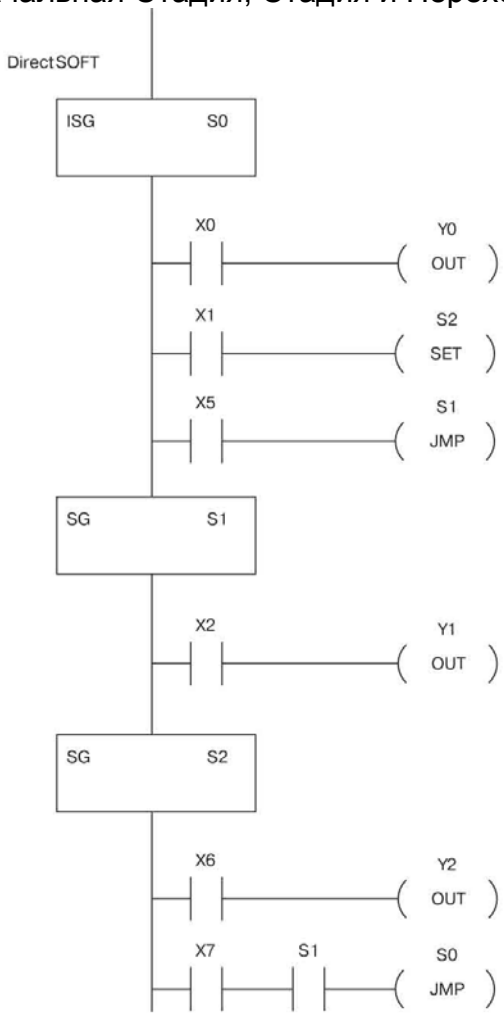
Команды языка RLL*PLUS*

Stage (SG) (Стадия)

Команды Stage (Стадия) используются 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 DL06
		aaa
Stage	S	0–1777

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

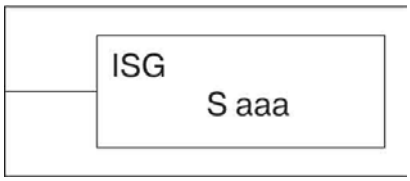


Последовательность нажатия клавиш ручного программатора

U ISG	→	A 0	ENT		
$ STR	→	A 0	ENT		
GX OUT	→	A 0	ENT		
$ STR	→	B 1	ENT		
X SET	→	SHFT	S RST	C 2	ENT
$ STR	→	F 5	ENT		
K JMP	→	B 1	ENT		
2 SG	→	B 1	ENT		
$ STR	→	C 2	ENT		
GX OUT	→	B 1	ENT		
2 SG	→	C 2	ENT		
$ STR	→	G 6	ENT		
GX OUT	→	C 2	ENT		
$ STR	→	H 7	ENT		
V AND	→	SHFT	S RST	B 1	ENT
K JMP	→	A 0	ENT		

Initial Stage (ISG)(Начальная Стадия)

Команда Initial Stage (Начальная Стадия) используется, как правило, в первом сегменте 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.

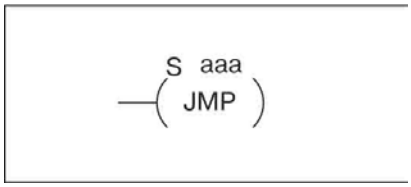


Тип данных операнда		Диапазон для DL06
		aaa
Stage	S	0–1777

Начальные Стадии также активизируются по логическим условиям перехода и по командам перехода или установки Стадии, исполняемым в текущей активной Стадии.

Jump (JMP)(Переход)

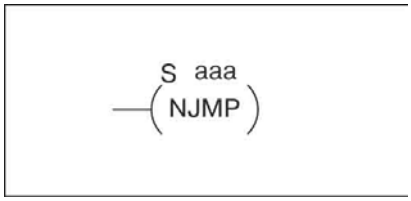
Команда Jump (Переход) позволяет программе перейти из текущей активной Стадии, в которой находится команда перехода, к другой Стадии (указанной в команде). Переход осуществляется, когда выполняется входное логическое условие. Активная Стадия, содержащая команду перехода, перестанет быть активной спустя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



Тип данных операнда		Диапазон для DL06
		aaa
Stage	S	0–1777

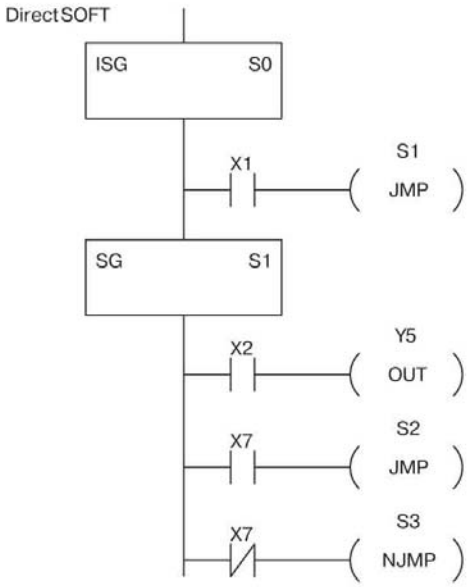
Not Jump (NJMP)(Переход по НЕ)

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 DL06
		aaa
Stage	S	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".



Последовательность нажатия клавиш ручного программатора

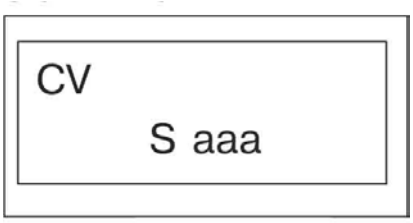
U ISG	→	A 0	ENT
$ STR	→	B 1	ENT
K JMP	→	B 1	ENT
2 SG	→	B 1	ENT
$ STR	→	C 2	ENT
GX OUT	→	F 5	ENT
$ STR	→	H 7	ENT
K JMP	→	C 2	ENT

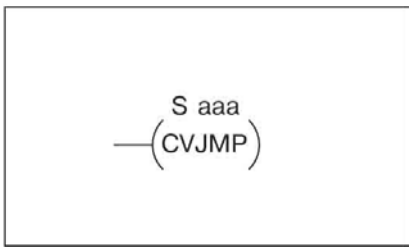
SHFT	N TMR	SHFT	K JMP	→	D 3	ENT	

Converge Stage (CV)(Сходящаяся Стадия) и Converge Jump (CVJMP) (Переход схождения)

Команда Converge Stage (Сходящаяся Стадия) используется с целью сведения определенных Стадий к одной Стадии. Сходящимся Стадиям присваивается тип "Сходящихся Стадий".

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.





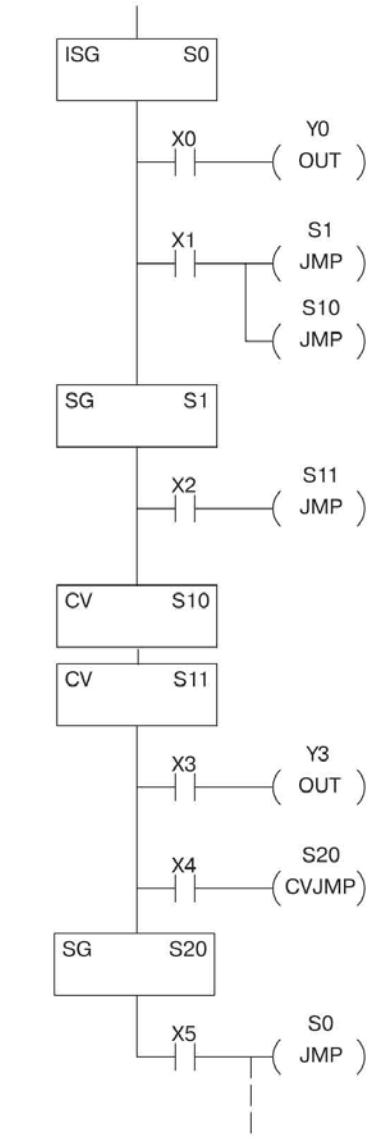
Дополнительные логические команды допускается вставлять только между последней Сходящейся Стадией и командой CVJMP. Включать несколько команд CVJMP не допускается.

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 DL06
		aaa
Stage	S	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.

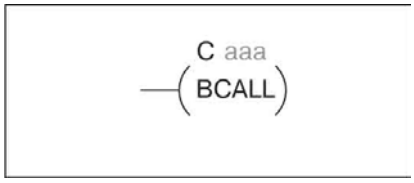
Отображение в Direct SOFT



Последовательность нажатия клавиш ручного программатора

U ISG	→	A 0	ENT				
$ STR	→	A 0	ENT				
GX OUT	→	A 0	ENT				
$ STR	→	B 1	ENT				
K JMP	→	B 1	ENT				
K JMP	→	B 1	A 0	ENT			
2 SG	→	B 1	ENT				
$ STR	→	C 2	ENT				
K JMP	→	B 1	B 1	ENT			
SHFT	C 2	V AND	→	B 1	A 0	ENT	
SHFT	C 2	V AND	→	B 1	B 1	ENT	
$ STR	→	D 3	ENT				
GX OUT	→	D 3	ENT				
$ STR	→	E 4	ENT				
SHFT	C 2	V AND	SHFT	K JMP	→	C 2	A 0	ENT
2 SG	→	C 2	A 0	ENT			
$ STR	→	F 5	ENT				
K JMP	→	A 0	ENT				

Block Call (BCALL)(Вызов блока)



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 CR - 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.

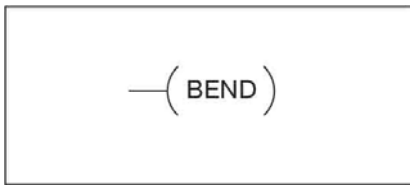
Тип данных операнда	Диапазон для DL06
	aaa
Control Relay …… S	0–1777

Block (BLK)(Блок)

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.

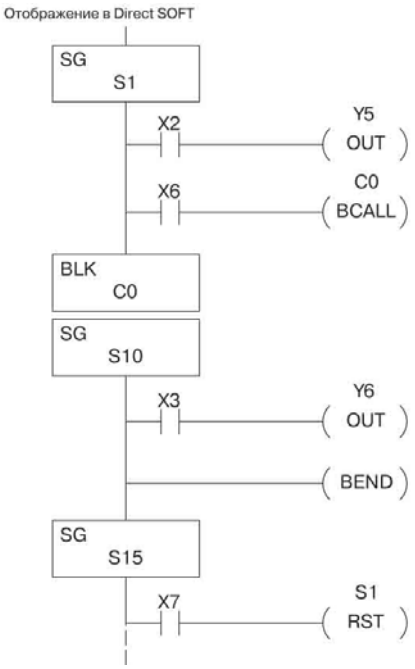
Тип данных операнда	Диапазон для DL06
	aaa
Control Relay …… S	0–1777

Block End (BEND)(Завершение блока)



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.

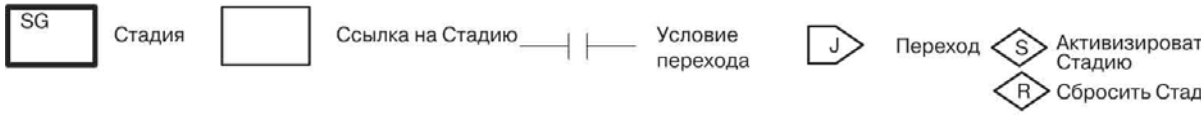


Последовательность нажатия клавиш ручного программатора

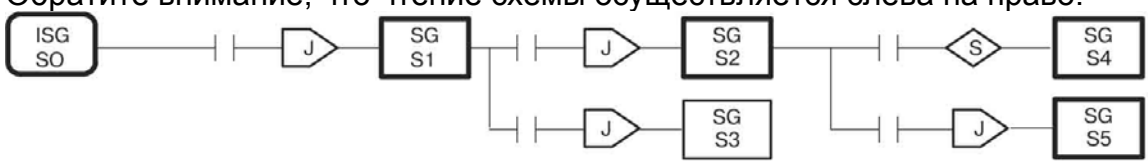
SG	→	S(SG)	1	ENT					
STR	→	X(IN)	2	ENT					
OUT	→	Y(OUT)	5	ENT					
STR	→	X(IN)	6	ENT					
SHFT	B	C	A	L	L	→	C(CR)	0	ENT
SHFT	B	L	K	→	C(CR)	0	ENT		
SG	→	S(SG)	1	0	ENT				
STR	→	X(IN)	3	ENT					
OUT	→	Y(OUT)	6	ENT					
SHFT	B	E	N	D	ENT				
SG	→	S(SG)	1	5	ENT				
STR	→	X(IN)	7	ENT					
RST	→	S(SG)	1	ENT					

Вид стадийной программы в пакете DirectSOFT32

Функция Stage View (Вид стадийной программы) в DirectSOFT32 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.

Что такое биты Включения Стадий?

Бит Стадии - это отдельный бит в регистре образа 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, используемый для смены Стадий (переходов состояний)?

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 DirectSOFT32 редактировать гораздо сложнее, и мы отделили их от двух предыдущих вопросов.

Можно ли сделать так, чтобы Стадия был активной только в течении 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

8. Работа контура ПИД-регулирования

В данной главе…

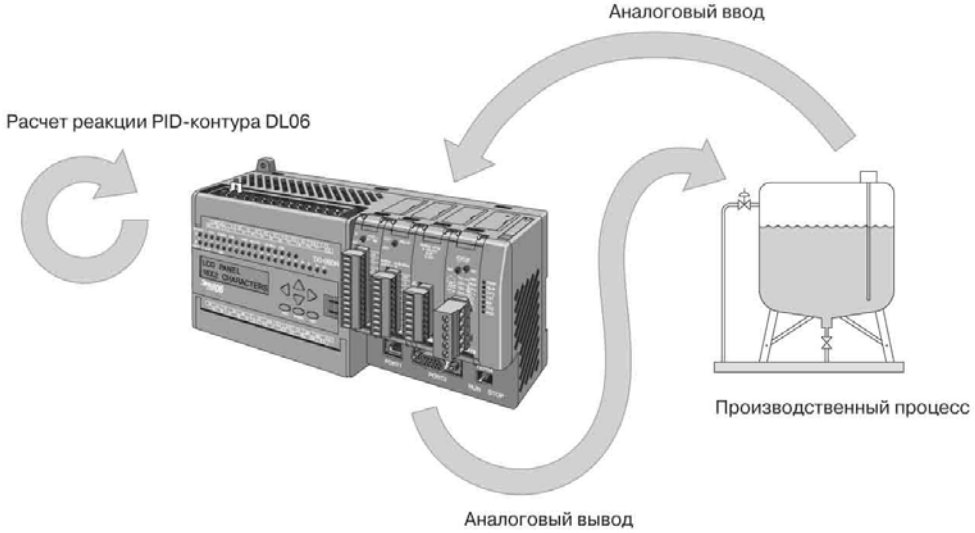
Характеристики контуров ПИД-регулирования в DL06

Основные свойства и функции.

Контуры регулирования, предусмотренные в DL06, обеспечивают обширный набор функций, позволяющий решать множество различных задач управления процессами. К основным свойствам и функциям относятся:

- Наличие до 8 контуров с независимым программированием частоты опроса
- Возможность ручного /автоматического /каскадного управления контуром
- Предусмотрено два типа безударных переходов
- Полнофункциональные аварийные сигналы
- Программный задатчик с профилем до 16 участков
- Автонастройка

Помимо выполнения релейной программы, в ЦПУ DL06 предусмотрена возможность организации контуров регулирования. Можно выбрать и сконфигурировать до 8 контуров. Все цепи датчиков и исполнительных механизмов подключаются непосредственно к модулям аналогового ввода /вывода DL06. Все измеряемые переменные процесса, величины коэффициентов передачи, аварийные уровни сигналов и т.п., связанные с каждым контуром, хранятся в Таблице переменных контура регулирования в ЦПУ. ЦПУ DL06 считывает переменные процесса (PV), присутствующие на его входах, в каждом цикле. 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 контур регулирования). В этой главе описывается работа контуров регулирования и действия, необходимые для их конфигурирования и настройки.



Лучшим средством конфигурирования контуров регулирования DL06 является программный пакет *DirectSOFT32*, версия 4.0 или более поздняя. Предусмотренные в *DirectSOFT32* 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 *DirectSOFT32*. Выбранные при конфигурировании и настройке параметры хранятся в энергонезависимой флэш-памяти DL06. Параметры контура также можно сохранить на диск, чтобы в дальнейшем считать их с диска и использовать повторно.

Свойства контура ПИД-регулирования	Характеристики
Количество контуров	Настраивается, максимум 8
Требуемая V-память ЦПУ	32 слова (V ячейки) для каждого выбранного контура, 64 слова при использовании программного задатчика
Алгоритм ПИД-регулирования	ПИД-регулирование по положению или по скорости (Position / Velocity)
Направление управляющего выхода (Control Output polarity)	Выбирается прямое или обратное действие
Зависимость от рассогласования	Выбирается зависимость от рассогласования: линейная, квадратичная или пропорциональная корню квадратному
Скорость обновления контура (время между расчетами ПИД уравнения)	Программируется пользователем в диапазоне 0.05…99.99 секунд
Минимальная скорость обновления контура	0.05 секунды для 1…4 контуров, 0.1 секунды для 5…8 контуров,
Режимы контуров	Автоматический, ручной (управление оператором) или каскадный
Генератор профиля программного задатчика	До 8 шагов "наклон /выдержка" (16 участков) на 1 контур. Индикация номера шага "наклон /выдержка"
Функция переменной процесса	Выбирается стандартная линейная функция или функция с извлечением квадратного корня (для входа измерителя расхода)
Границы задания (Set point 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 (плюс/минус) 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 заданию (требуемому значению). В правильно работающем контуре управления поддерживается минимальное значение рассогласования.

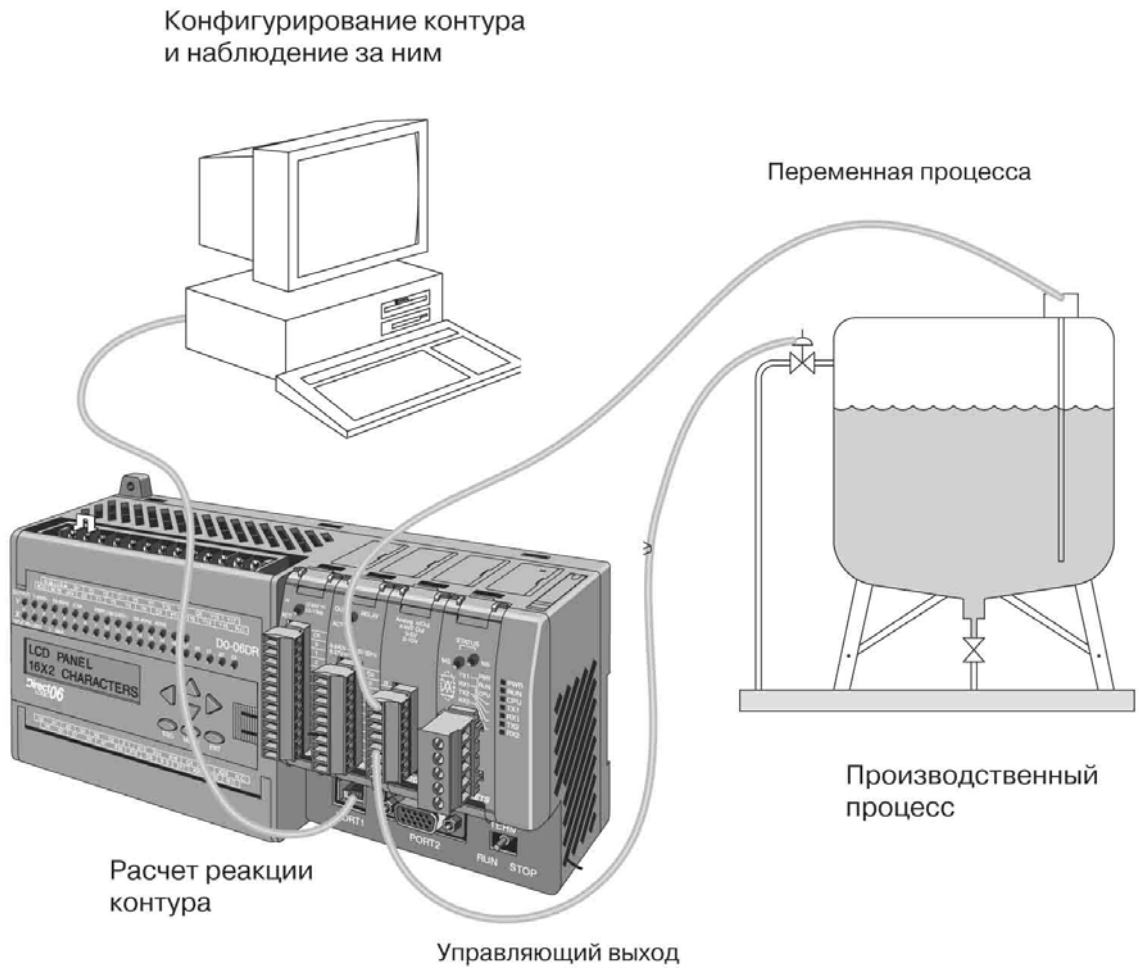
Расчет реакции контура – применение в реальном времени к сигналу рассогласования математического алгоритма, на основании которого вырабатывается такой управляющий сигнал, который минимизирует величину рассогласования. Существуют различные типы алгоритмов управления. В DL06 используется алгоритм ПИД – регулирования (Пропорциональное-Интегральное-Дифференциальное), 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).

На следующем рисунке каждому элементу контура поставлен в соответствие реальный физический объект. В производственном процессе, представленном на рисунке, участвует жидкость, содержащаяся в танке реактора. Зонд датчика измеряет некую переменную процесса, которой может быть давление, температура или другой любой параметр. Сигнал датчика усиливается преобразователем и передается по проводам в виде аналогового сигнала в модуль ввода ПЛК.
Контроллер (ПЛК) считывает переменную процесса (PV) на своем аналоговом входе. ЦПУ выполняет расчет реакции контура и выставляет определенный уровень на аналоговом выходе. Сигнал поступает на исполнительный механизм, задействованный в производственном процессе, например, нагреватель, клапан, насос и т.п. Спустя некоторое время регулируемый параметр изменяется в достаточной степени, чтобы изменение было обнаружено датчиком. Соответственно, изменяется и переменная процесса. Выполняется новый расчет реакции контура, и таким образом контур работает непрерывно.



На изображенном на рисунке компьютере работает программный пакет *DirectSOFT32*, служащий для программирования контроллеров *Direct*LOGIC. Для программирования ПЛК DL06 (включая функцию ПИД-регулирования) может использоваться *DirectSOFT32* версии 4.0 или более поздней. 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 *DirectSOFT32*.

Параметры настройки контура

Таблица контуров и количество контуров

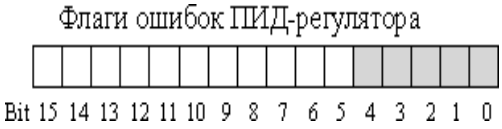
Источником команд для реализации ПИД-регулирования в ПЛК DL06 являются только таблицы, хранящиеся в V-памяти. Это означает, что в языке релейной логики RLL нет команды "ПИД-регулирование". Вместо этого процессор читает параметры контура из зарезервированных ячеек V-памяти. Как показано в таблице ниже, 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	Восьмеричные	V1200 – V7340, V10000 – V17740	Запись
V7641	Количество контуров	BCD	0 – 8	Запись
V7642	Флаги ошибок контура	Двоичные	0 или 1	Чтение

Если указано нулевое количество контуров, то во время выполнения релейной программы задача расчета контуров будет отключена. Контроллер контуров регулирования позволяет использовать контуры в порядке возрастания, начиная с контура 1. Например, нельзя использовать контуры 1 и 4, пропустив 2 и 3. Контроллер контуров регулирования пытается управлять всеми контурами, указанными в V7641.

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.



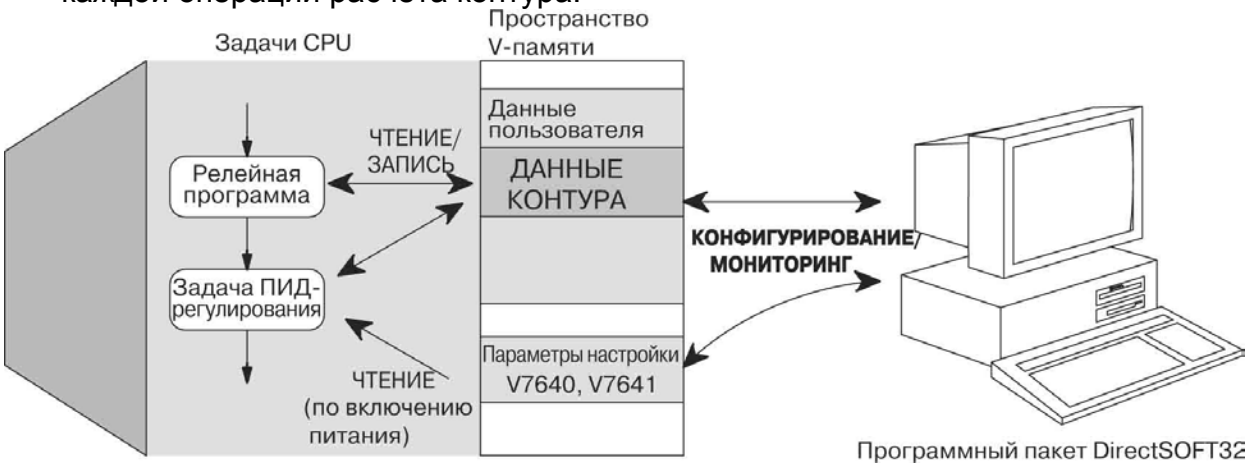
При использовании диалогового окна настройки контуров в *DirectSOFT32* 32 встроенная автоматическая проверка диапазона препятствует возможным ошибкам настройки. Однако параметры настройки могут быть записаны другими способами, например, из программы релейной логики, поэтому в подобных случаях регистр 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 - памяти.
2	Выбранное количество контуров (в V7641) превышает 8.
3	Таблица контуров вышла за пределы V7377. Используйте адрес ближе к V1200

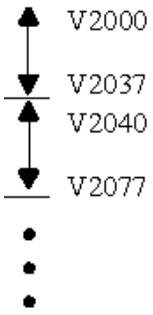
В режиме выполнения отсутствие ошибок программирования можно быстро проверить по значению регистра V7642. Оно должно быть равно 0000.

Задание размера и расположение 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.



Данные пользователя
Контур 1 32 слова
Контур 2 32 слова
Контур 3 32 слова
Контур 4 32 слова

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	Чтение в любой момент
1	Addr + 0	Настройки контура ПИД-регулирования. Режим I	биты	Да
2	Addr + 1	Настройки контура ПИД-регулирования. Режим II	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	времени 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 аварийного сигнала	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 предельне 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	Веохнее 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	
32	Addr + 37	Прямой доступ к управляющему выходу, номер канала	слово / шестнадцатеричный	

*Чтение данных только при переходе бита активации аварийных сигналов из 0 в 1.

**Чтение данных только при изменении режима ПЛК.

Параметры режима 1 ПИД-регулятора (Addr+00)

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

Бит	Описание настройки режима 1 работы контура ПИД-регулирования	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

Параметры режима 2 ПИД-регулятора. Описание битов (Addr+01)

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

Бит	Описание настройки режима 2 работы контура ПИД-регулирования	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 входа 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	–	Каскадный
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	–	Ошибка
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".

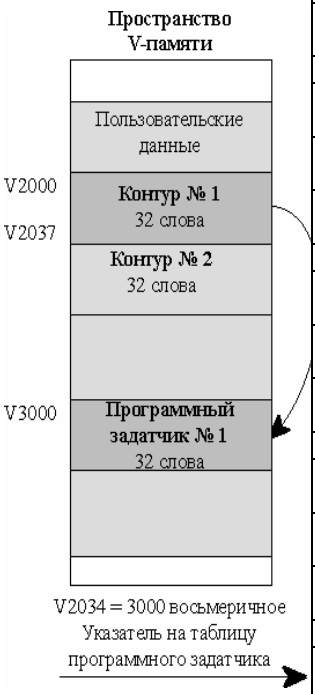
Бит	Описание битов флагов программного задатчика	Чтение/ Запись	Бит=0	Бит=1
0	Запуск программного задатчика	запись	–	0⇒1 Старт
1	Остановка программного задатчика	запись	–	0⇒1 Пауза
2	Возобновление работы программного задатчика	запись	–	0⇒1 Возобн.
3	Перевод программного задатчика на один шаг	запись	–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

Биты 8-15 должны считываться как байт, показывающий номер текущего шага профиля 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 участки 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 поток значений сигналов задания SP, называемый профилем. Для использования программного задатчика нужно создать отдельную таблицу из 32 слов с соответствующими значениями. Диалоговое окно DirectSOFT32 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 далее в этой главе.



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 флага ошибки программного задатчика	Чтение/Запись	Бит=0	Бит=1
0	Стартовый адрес ниже диапазона V-памяти	чтение	–	Ошибка
1	Стартовый адрес выше диапазона V-памяти	чтение	–	Ошибка
2–3	Зарезервировано для будущего использования	–	–	–
4	Стартовый адрес в диапазон системных параметров V-памяти	чтение	–	Ошибка
5–15	Зарезервировано для будущего использования	–	–	–

Частота опроса и планирование контура

Частота опроса контура (Addr+07)

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.

Примечание: 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 – есть ни что иное, как частота расчета реакции контура ПИД-регулирования. Результатом каждого расчета является новое значение управляющего выхода. В ЦПУ DL06 частоту опроса контура можно установить в пределах 50 мс…99,99 с. В большинстве контуров не требуется обновление управляющего выхода (реакции ПИД-контура) в каждом цикле ПЛК. На практике расчет реакции для некоторых контуров может потребоваться один раз за тысячу циклов.

Выбор наилучшей частоты опроса

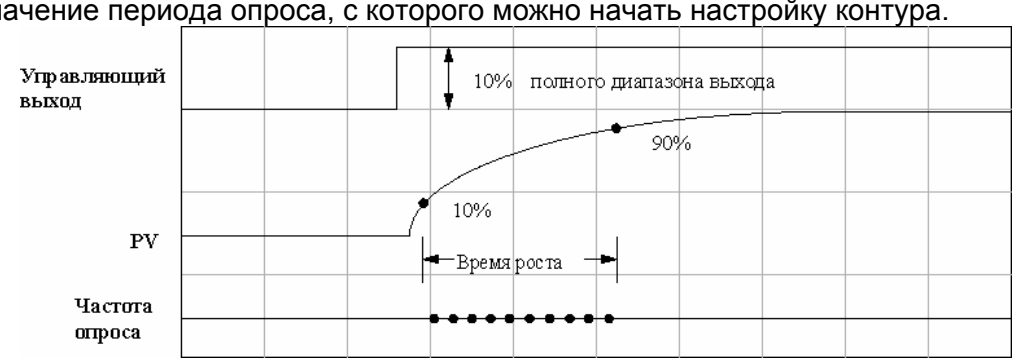
Для контура управления невозможно подобрать какую-то определенную и единственную идеальную частоту. Оптимальная частота опроса представляет собой компромисс, одновременно удовлетворяющий различным условиям:

- Требуемая частота опроса пропорциональна времени отклика переменной процесса (PV) на изменение управляющего выхода. Обычно, инерционным процессам соответствует малая частота опроса, а для быстрых (неинерционных) 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 подходящей частоты опроса (Addr+07):

1. 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.
2. В определенный момент, когда процесс испытывает незначительное внешнее возмущение, измените значение на управляющем выходе сразу на 10%.
3. Зафиксируйте время роста или спада PV (интервал между уровнем 10% и 90%).
4. Полученный интервал роста или спада поделите на 10. Это и будет начальное значение периода опроса, с которого можно начать настройку контура.

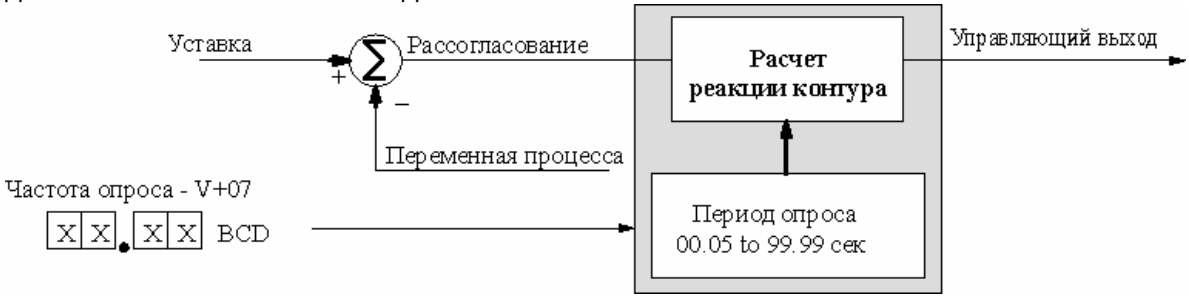


Предположим, что измеренное время роста PV вследствие изменения управляющего выхода составило 25 секунд (см. рис. выше). Рекомендуемая частота опроса для этого случая составит 2.5 секунды. Для наглядности процесс опроса показан в виде десяти точек на линии частоты опроса в пределах интервала роста PV. 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 высокая частота опроса 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 составляет 20 дискретов 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 *DirectSOFT32* 32, 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	
Минимальное значение	150 мкс
Типичное значение	250 мкс
Максимальное значение	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:

Средняя длительность цикла с ПИД-контуром =

$$\left[\frac{\text{длительность цикла без контура}}{\text{период опроса контура}} \bullet \text{время расчета ПИД} - \text{контура}\right] + \text{длительность цикла без контура}$$

Средняя длительность цикла с ПИД-контуром =

$$\left[\frac{50\ \text{мсек}}{3\ \text{сек}} \times 250\ \text{мксек}\right] + 50\ \text{мсек} = 50.004\ \text{мсек}$$

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:

Средняя длительность цикла с ПИД-контурами =

$$\left[\sum_{n=1}^{n=L} \frac{\textit{длительность цикла без контура}}{\textit{период опроса для n} - \textit{го контура}} \bullet \textit{время расчета ПИД} - \textit{контура}\right] + \textit{длительность цикла без контуров}$$

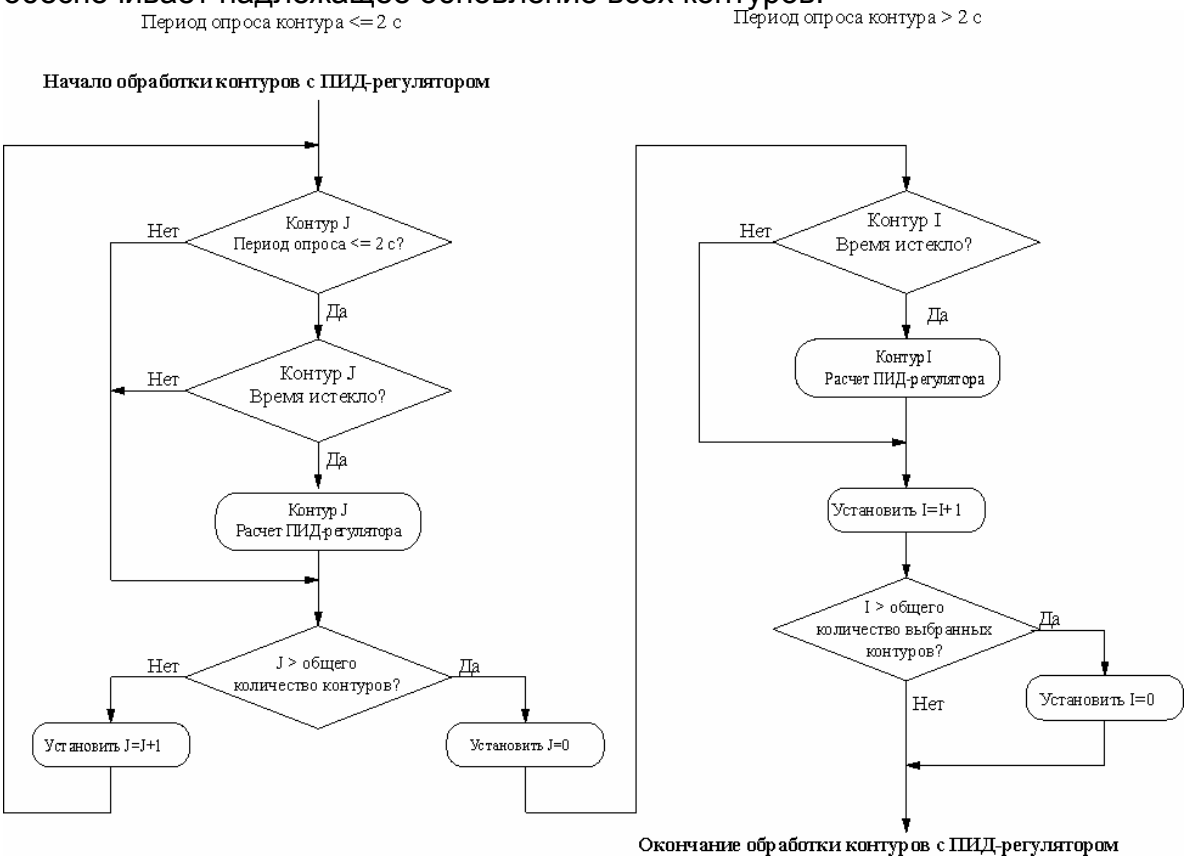
В новом выражении увеличение длительности для каждого контура от 1 до L (последний контур) суммируется (внутри квадратных скобок) и полученная сумма добавляется к правому слагаемому "длительность цикла без контуров" только один раз. Допустим, в ПЛК DL06 используются четыре контура регулирования. В следующей таблице приводятся параметры и суммарные значения для каждого контура.

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

Теперь, сложив суммируемые слагаемые и добавив значение длительности цикла без контуров, мы получим:

Средняя длительность цикла без ПИД-контуров = [50 мкс + 0.42 мкс + 1.25 мкс + 8.3 мкс] + 50 мс = 50.06 мс

ЦПУ DL06 выполняет расчет реакции ПИД-контура лишь в назначнных 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 обрабатывается столько контуров с периодами опроса <= 2 секунды, сколько требуется для обеспечения частоты опроса каждого контура. Задание большей частоты для контуров приведет к увеличению длительности цикла ПЛК. *Используйте эту возможность только в случае необходимости!*
Контуры с периодом опроса > 2 секунды обрабатываются с частотой 1 или менее контуров за цикл (с минимальной частотой, необходимой для поддержания частоты 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 ЦПУ DL06, предоставляют возможность управления сложными процессами. Сложность отладки систем автоматического управления состоит в том, что одна и та же неисправность может быть вызвана многими возможными причинами. При введении новых контуров управления в работу рекомендуется использовать осторожный, пошаговый подход.

Шаг 1: знание технологии

Самое главное – это знать, как создается конечный продукт или изделие. Знание технологии является основой проектирования эффективной системы управления. Хорошее знание технологии предполагает следующее:

Определение всех существенных переменных процесса, например, значения температуры, давления, расхода и т.п., которые требуют точного управления.

Выбор требуемых уставок для каждой переменной процесса на период цикла процесса.

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

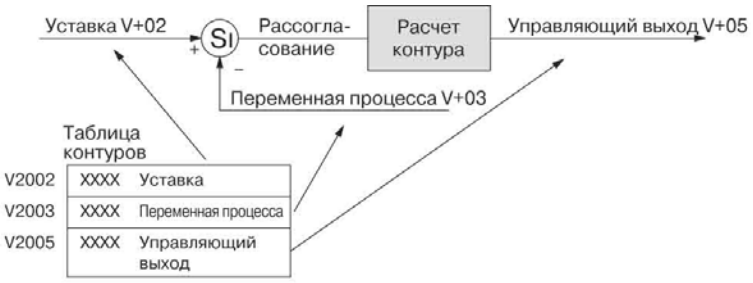
После выбора стратегии управления другим принципиальным вопросом является *выбор надлежащих размеров исполнительных механизмов и шкалы датчиков.*

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 DL06 поддерживаются двенадцати-битовые, пятнадцати-битовые и шестнадцати-битовые форматы данных со знаком и без знака, а так же шестнадцати-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 "ПЛК-системы" предлагает для серии DL06 модули аналогового ввода с четырьмя входными каналами, рассчитанными на сигналы 0…20 мА или 4…20 мА. Также в наличии имеются комбинированные модули аналогового ввода и вывода. Дополнительную информацию по этим модулям можно найти в нашем каталоге, а так же в Internet по адресу: **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 **D0–OPTIONS–M.** 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 между элементами контура.

Шаг 6: параметры контура

Выполнив проводные соединения и монтаж, следует выбрать параметры настройки контура. Программирование таблиц контуров проще всего сделать с помощью *DirectSOFT32* 32 (4.0 или более поздняя версия). В этом программном пакете предусмотрены диалоговые окна PID Setup (Настройка ПИД-контура), которые существенно упрощают задачу. **Примечание:** 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 (см. *Настройка контура* на стр.8-38), что PV изменяется правильно и управляющий выход воздействует на процесс должным образом, можно перейти к процедуре настройки контура (см. *Автоматический режим* на стр. 8-39). На данном этапе контур настраивается с таким расчетом, чтобы PV автоматически следовала за SP.

Шаг 9: выполнение цикла процесса

Если проверка замкнутого контура показала, что PV отслеживает малые изменения SP, можно 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.

Шаг 10: сохранение параметров контура

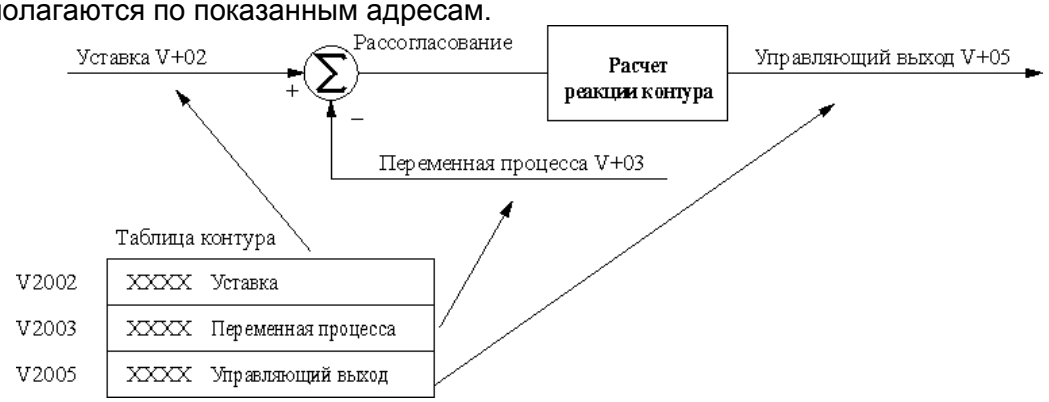
Когда проверка и настройка контуров завершены, не забудьте сохранить все параметры контура на диск.

ВНИМАНИЕ: 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.

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 (SP), переменная (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. Не забывайте, что большинство аналоговых модулей мультиплексируют данные, используя два-три бита декодирования адреса канала. Примеры программ, показывающие, как выполнять обмен аналоговыми данными между аналоговыми модулями DL06 и произвольной ячейкой V-памяти, можно найти в руководстве по аналоговому модулю.

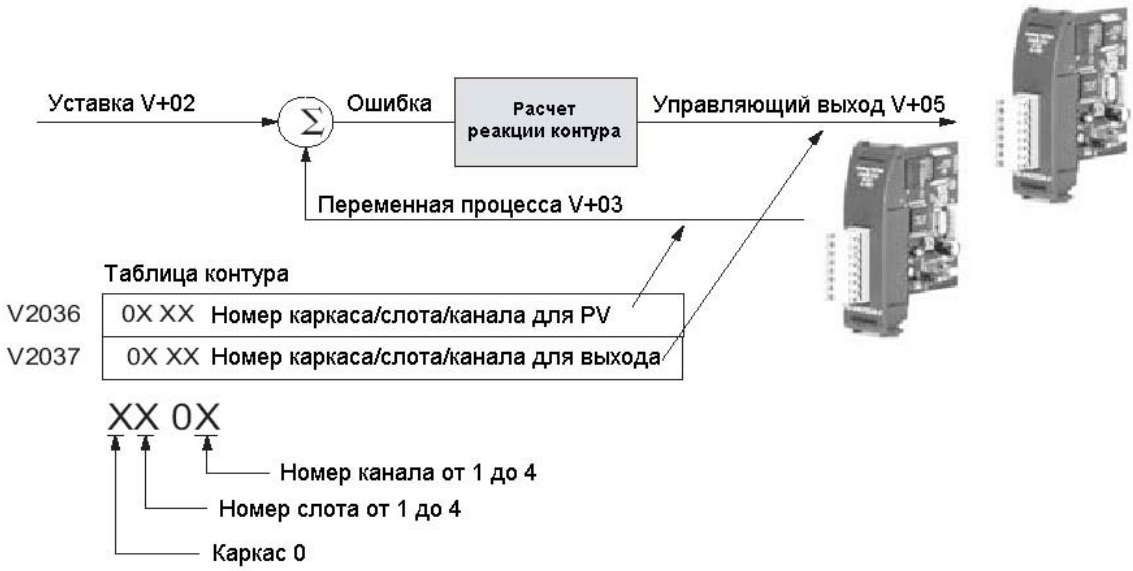
Как показано на рисунке, у уставки может быть несколько источников. Многие приложения в зависимости от режима цикла в разное время будут использовать два или более источников. Кроме того, способ генерирования уставки также определяется топологией контура и методом программирования. При использовании встроенного программного задатчика ПИД-контроллер автоматически записывает данные уставки в ячейку addr+02. **Если Вы хотите использовать уставку из любого другого источника, релейная программа должна записать уставку в ячейку addr + 02 таблицы контура.**

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 DL06 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 соответсвующих регистрах таблицы контура. На следующем рисунке показаны параметры таблицы контура с адресами V+36 и V+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 DirectSOFT32. Например, значение «0102» в регистре V2036 указывает контроллеру контура считывать данные PV из слота номер 1 и второго канала. 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. Если 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..

Функции прямого доступа PV с фильтрацией.

Встроенный фильтр использует следующий алгоритм:

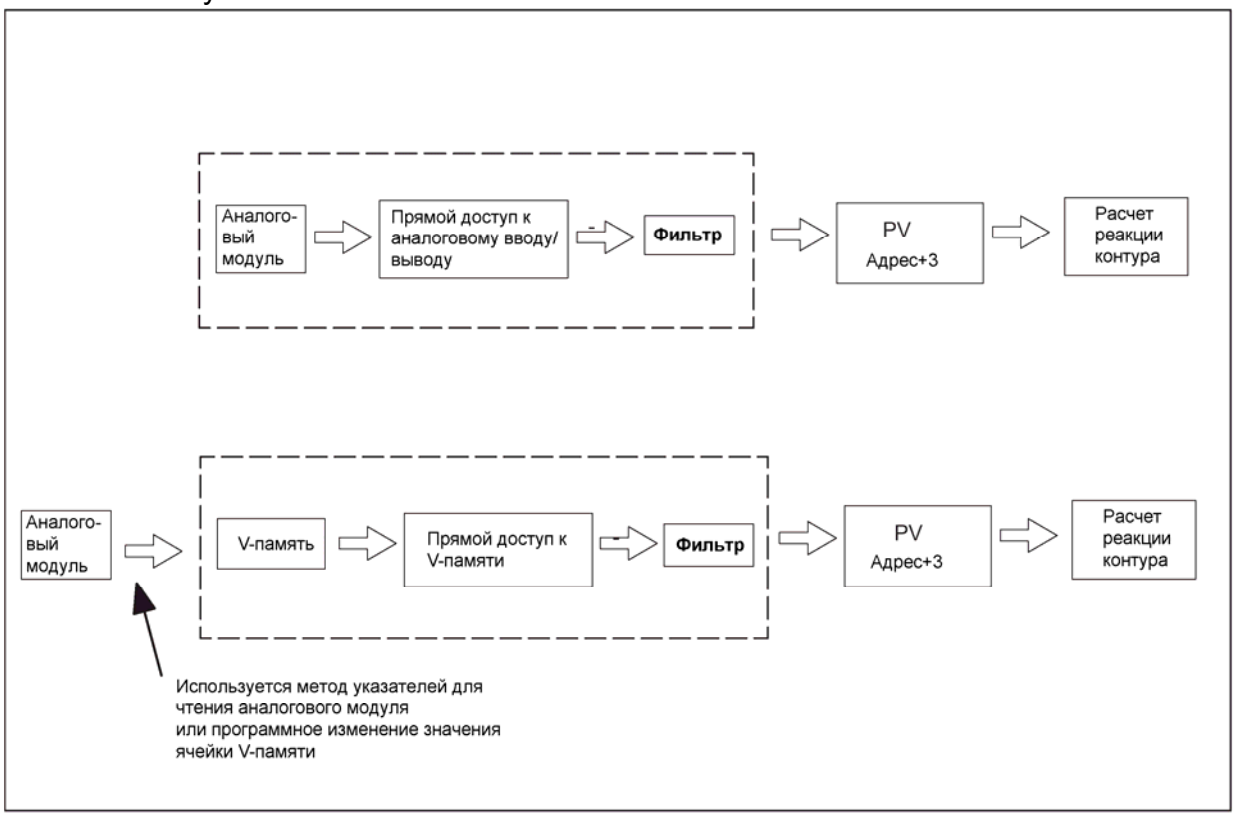
$$y_i = k\,(x_i - y_{i-1}) + y_{i-1}$$

- y_i – текущий выход фильтра
- x_i - текущий вход на фильтр
- y_i–1 - предыдущий выход фильтра
- k – коэффициент входного аналогового фильтра PV

На следующих ниже диаграммах показано как взаимодействуют функция прямого доступа (адрес + 36) и фильтрация PV (адрес + 01, бит 2). Выберите:

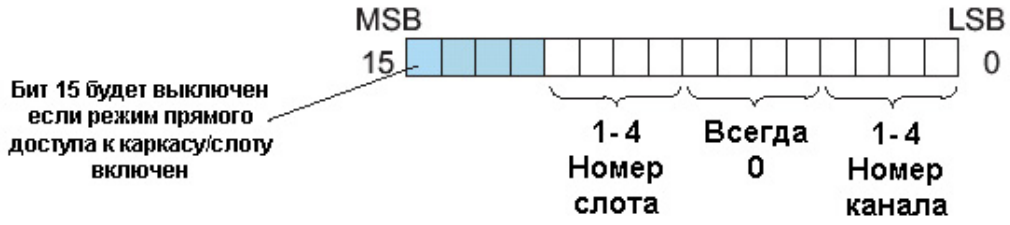
- Прямой доступ из аналогового канала модуля ввода/вывода с включенным или отключенным фильтром. Когда эта функция используется, то метод указателей для чтения аналоговых каналов модуля применять нельзя.

Прямой доступ к ячейке V-памяти с включенным или отключенным фильтром. При использовании этой функции необходимо использовать либо метод указателей для чтения аналоговых каналов, либо программное изменение значения указанной ячейки V– памяти.



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

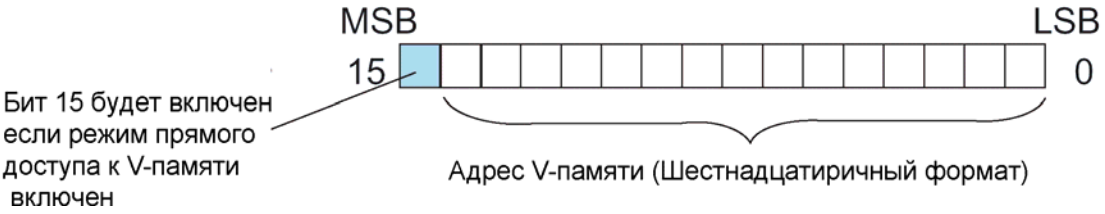
Определение полубайтных значений в слове прямого доступа PV показаны в таблице ниже для доступу к выбранному каркасу/слоту. Когда эта функция используется для любого канала на аналоговом модуле ввода, то метод указателей в релейной программе для чтения этого модуля применятся не может. (Смотрите Руководство по аналоговым модулям ввода - вывода DL06 (D0-OPTIONS-M) для дополнительной информации о методе указателей).



Тип	Номер слота	Номер канала
DL06	1-4	1-4

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

Определение слова прямого доступа PV показаны в таблице ниже для доступу к выбранной ячейке V-памяти. Когда эта функция используется, то может применяться метод указателей в релейной программе для чтения этого модуля. (Смотрите Руководство по аналоговым модулям ввода - вывода DL06 (D0-OPTIONS-M) для дополнительной информации о методе указателей).

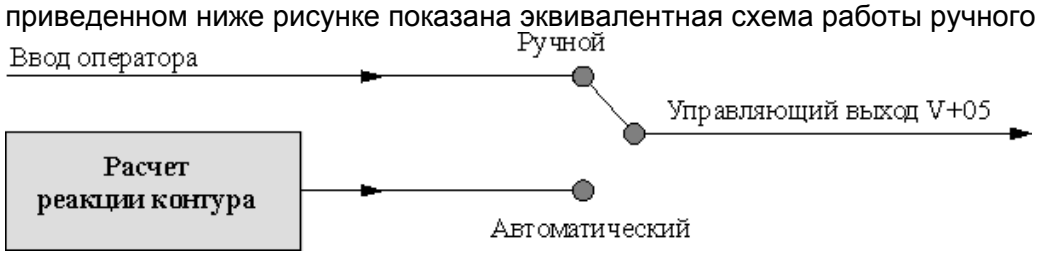


Тип	Диапазон V-памяти (Область Данных)
DL06	V1200-V7377 / V10000-V17777

Режимы контуров

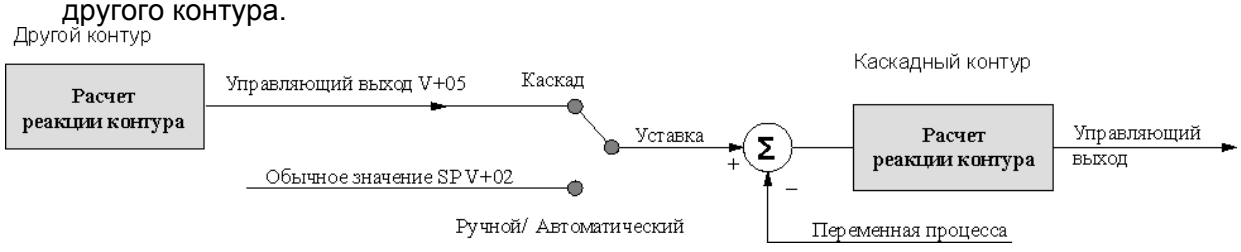
Для процессора DL06 возможны следующие три стандартных режима управления — Ручной, Автоматический и Каскадный. Источник данных для основных трех переменных: SP, PV и управляющего выхода, в каждом режиме различен. Далее следует введение в режимы управления и их источники переменных.

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.



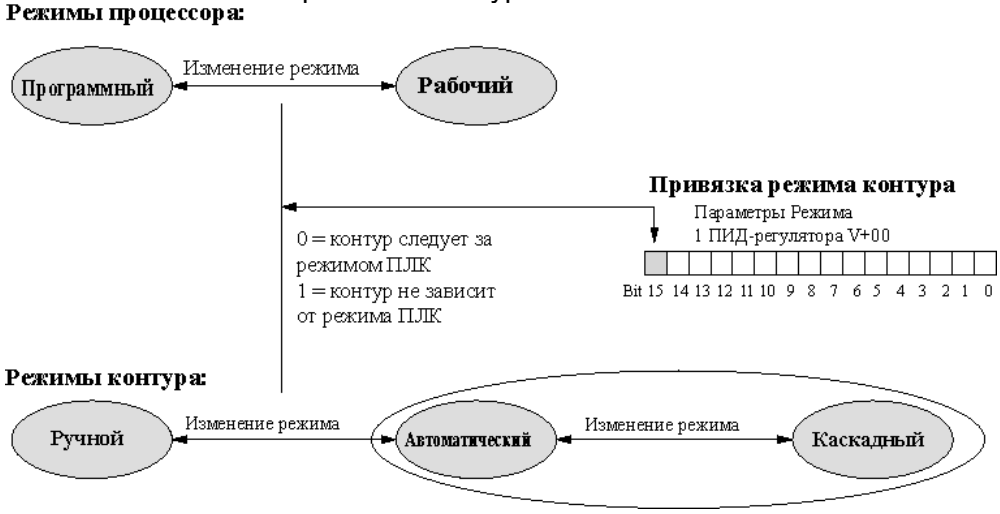
То, как расчеты реакции ПИД-регулятора изменяют источники данных для Ручного/Автоматического/Каскадного режимов, естественно приводит к возникновению определенных ограничений на изменение режимов. Как показано ниже, контур может переходить из одного режима в другой, но не может перейти непосредственно 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 в процессоре DL06 является возможность выполнять расчеты ПИД-регулятора, когда процессор находится в Программном режиме. Обычно процессор, находящийся в режиме программирования, прекращает все операции. Однако процессор DL06 в Программном режиме в зависимости от конфигурации может 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 каналам аналоговых модулей для работы со значениями PV и управляющего выхода. Контроллер контура обладает такой возможностью, описанной ранее в данной главе в разделе о прямом доступе к аналоговому вводу/выводу (ранее в данной главе).

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 параметров addr+00 Режима 1 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, а другой обеспечивает работу контуров процесса.

Примечание. Для изменения любых значений параметров таблицы контура необходимо, чтобы контур работал в ручном режиме, а контроллер остановлен. При независимой от режима ЦПУ работе контура нужно произвести некоторые дополнительные действия. Необходимо временно заставить контуры следовать режиму процессора устанавив бит 15 в 0. В этот момент средство программирования (например, DirectSOFT32) сможет перевести изменяемый контур в Ручной режим. После изменения значения параметра не забудьте восстановить независимую работу контура устанавив бит 15 в 1.

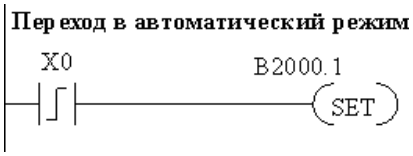
Как изменять режимы контуров

Первые три бита слова addr+00 Режима 1 ПИД-регулятора запрашивают режим работы соответствующего контура. Примечание: Эти биты представляют собой запрос режима, а не команду на переход в него (некоторые условия могут помешать изменению режима - см. на следующей странице).

Обычным состоянием этих битов запроса является «000». Для запроса изменения режима необходимо для 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:

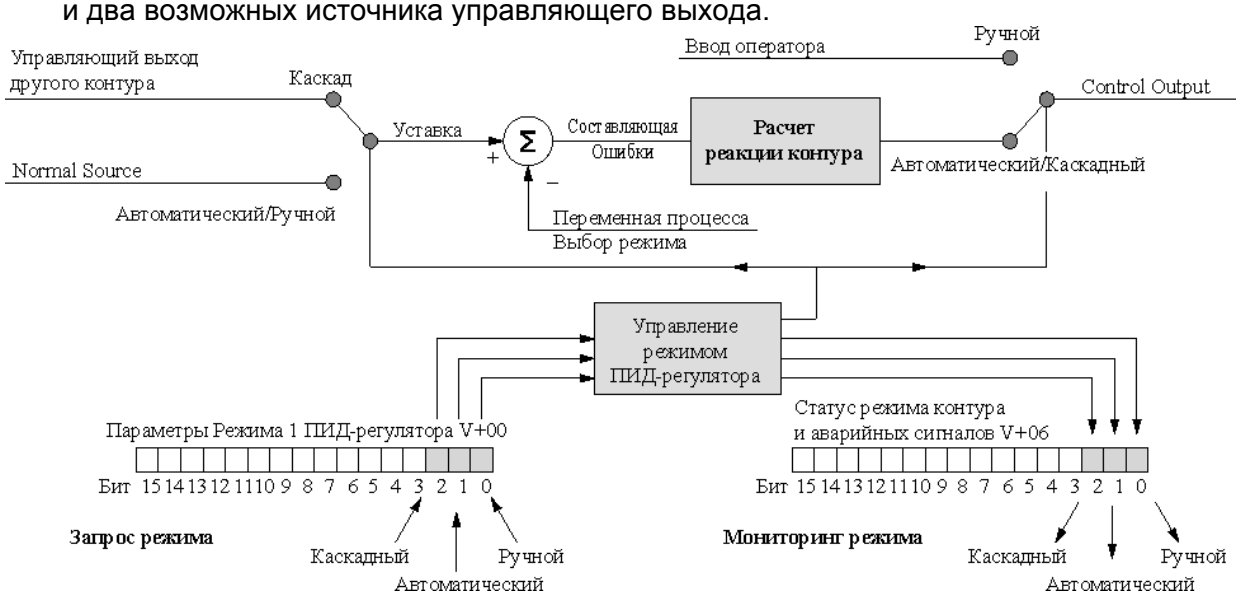
- PID View (Просмотр ПИД-регулятора) в DirectSOFT32. Самый простой способ. Щелкните по одной из кнопок, и DirectSOFT32 установит соответствующий бит.
- Ручной программатор. Используйте режим Статус слова (WD ST) для отслеживания содержимого addr+00, которое является BCD/шестнадцатеричным значением из 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 (не используйте обмотку OUT). При переходе 0-1 для X0 цепь устанавливает бит Автоматического режима = 1. Контроллер контура сбрасывает это бит.



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

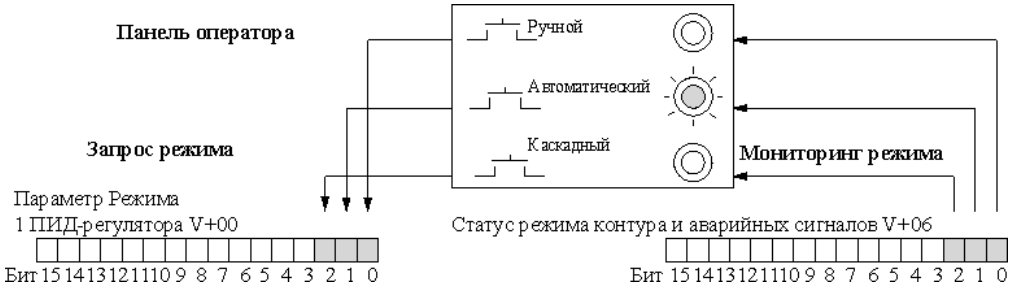
Так как изменение режима только запрашивается, контроллер контура ПИД-регулирования решает, когда разрешить изменение режима, и обеспечивает статус режима контура. Он устанавливает текущий режим в битах 0, 1 и 2 слова Статуса режима контура и аварийных сигналов, ячейка addr+06 в таблице контура. Параллельные 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, а связаны с соответствующими ячейками данных контроллера.



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 8 контуров как желаемые режимы работы. Если, пока ПЛК находился в режиме исполнения, произошли 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 сконфигурировать новые контура. Новые контура 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 реализации запрашиваемого режима:

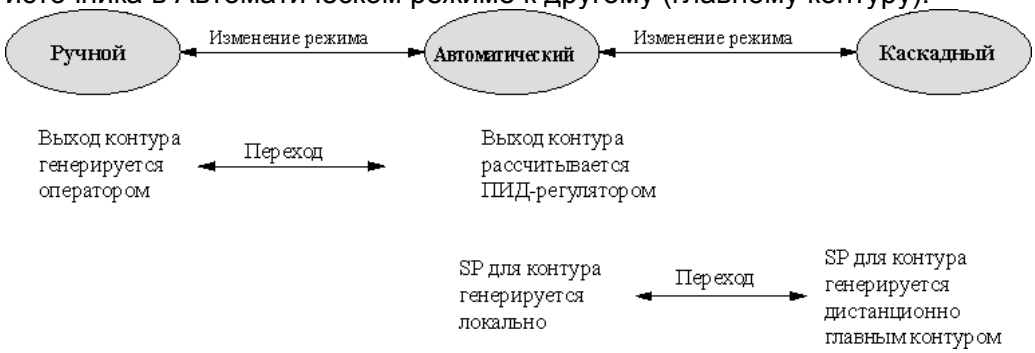
- Контур, который зависит от режима ПЛК, не может изменить режим, пока ПЛК находится в Программном режиме.
- Главный контур каскадной пары контуров не может перейти из Ручного в Автоматический режим, пока подчиненный контур находится в Каскадном режиме.

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.

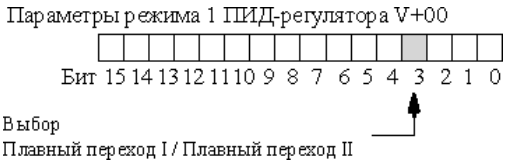
Безударные переходы

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 контуром к другому. Например, при изменении режима контура с Автоматического на Ручной 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», управляющего выхода, нарушающего до некоторой степени работу контура. Функция «безударного перехода» уравнивает параметры на момент изменения режима контура, делая переход плавным (скачок управляющего выхода отсутствует).

Существует два типа Безударного перехода контроллера контура DL06: Безударный переход I и Безударный переход II. Тип перехода выбирается с помощью диалогового окна PID Setup (Настройка ПИД-регулятора) DirectSOFT32. Или можно, как показано, использовать бит 3 addr+00 режима 1 ПИД-регулятора.



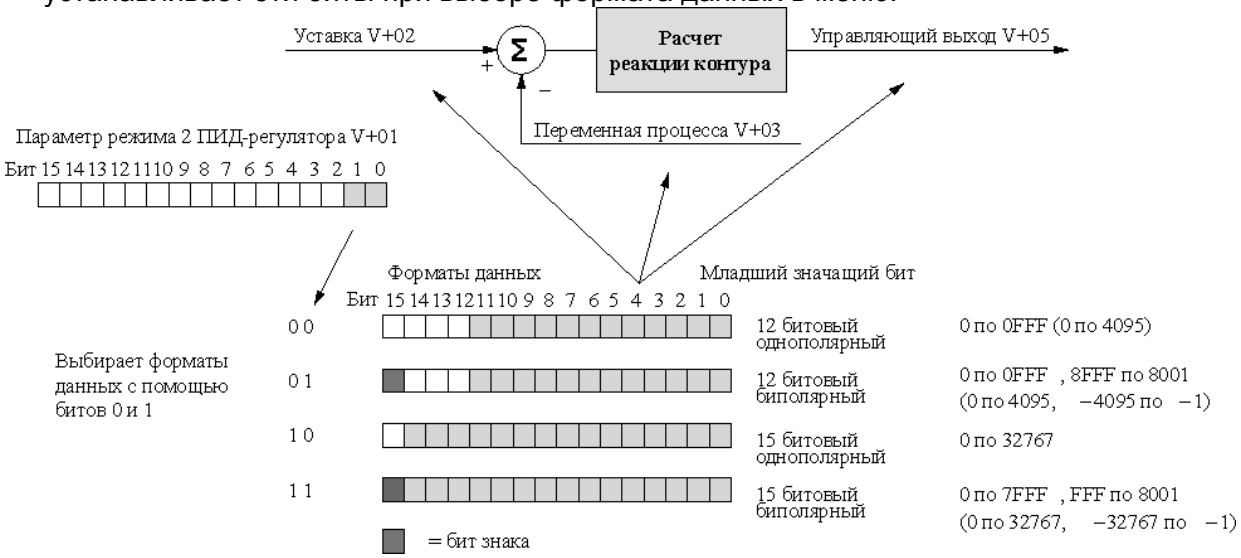
Характеристики типов безударных переходов I и II приведены в следующей таблице. Обратите внимание, их работа также зависит от используемого алгоритма ПИД-регулятора (позиционный или скоростной). Заметим, что при использовании уравнения ПИД-регулятора в скоростной форме необходимо использовать тип I безударного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

Тип перехода	Бит выбора перехода	ПИД-Алгоритм	Действие при переходе из режима Ручной в Автоматический	Действие при переходе из режима Автоматический в Каскадный
Без-ударный переход I	0	Позицион-ный	Устанавливает смещение = выходу Устанавливает SP = PV	Устанавливает выход основного контура = PV подчиненного контура
		Скоростной	Устанавливает SP = PV	Устанавливает выход основного контура = PV подчиненного контура
Без-ударный переход II	1	Позицион-ный	Устанавливает смещение = выходу	отсутствует
		Скоростной	отсутствует	отсутствует

Конфигурирование данных ПИД-контуров

Форматы данных параметров контура

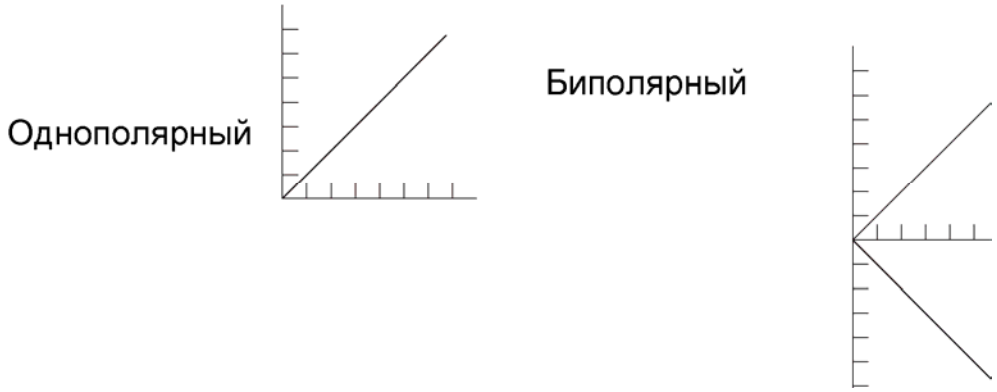
С выбором диапазона и разрешающей способности переменной процесса связан выбор формата данных трех основных переменных контура: уставки (SP), переменной (PV) и управляющего выхода (этот формат данных также используется суммой Интегратора в addr+04). Четырьмя возможными форматами данных являются 12- или 15-битовый (с выравниванием вправо), знаковый или беззнаковый (в биполярных форматах старший значащий бит является битом знака). Формат выбирается четырьмя двоичными комбинациями бит 0 и 1 слова V+01 режима 2 ПИД-регулятора. Диалог PID Setup (Настройка ПИД-регулятора) DirectSOFT32 автоматически устанавливает эти биты при выборе формата данных в меню.



Формат данных — это очень мощный установочный параметр, так как он определяет 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 уставки скорости требуемый управляющий выход также равен нулю. В этом случае должен использоваться биполярный формат.



Обработка смещения данных

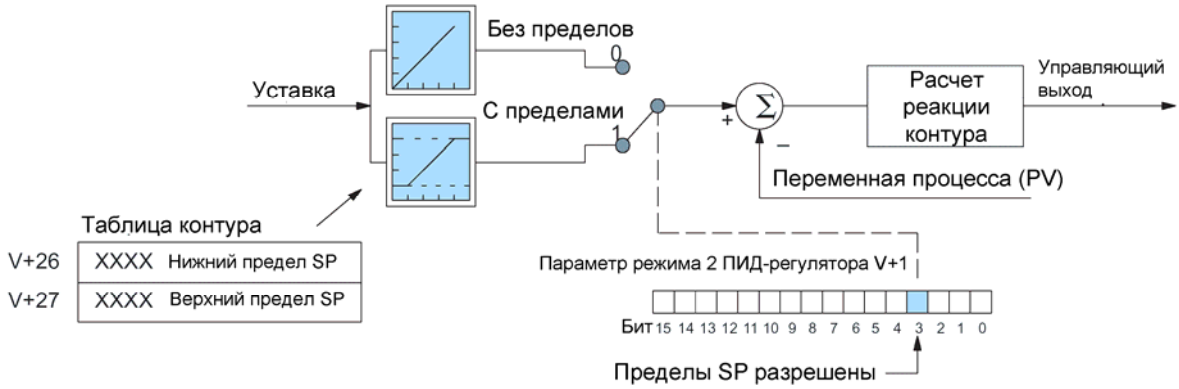
Во многих приложениях с периодическим технологическим процессом датчики или исполняющие механизмы взаимодействуют с аналоговыми модулями DL06, используя сигналы в диапазоне 4-20 мА. Этот тип сигнала обладает встроенным 20% смещением, так как нулевой точкой является 4 мА, а не 0 мА. Однако помните, что большинство аналоговых модулей преобразуют сигналы в данные *с одновременным устранением смещения*. Например, сигнал 4-20 мА часто преобразуется в 0000 - 0FFF (шестнадцатеричный), или 0 - 4095 (десятичный). В этом случае потребуется только выбрать 12-битовый однополярный формат данных и убедиться, что релейная 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 релейная программа записала значение, эквивалентное смещению, в регистр интегратора (addr+04). Тогда контур воспримет это смещение 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 addr+01 режима 2 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.

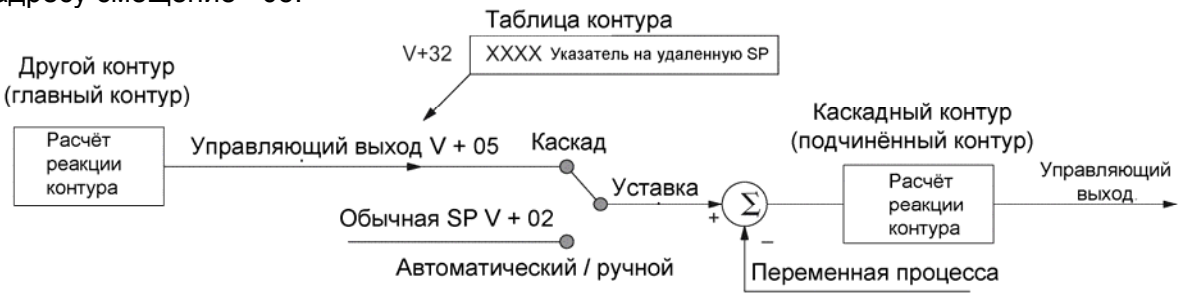


Установленные верхний и нижний пределы SP проверяются при расчете контура перед каждым вычислением. Это означает, что релейная программа в ходе выполнения процесса может изменить значения пределов, позволяя поддерживать узкий защитный диапазон изменения значения SP.

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.

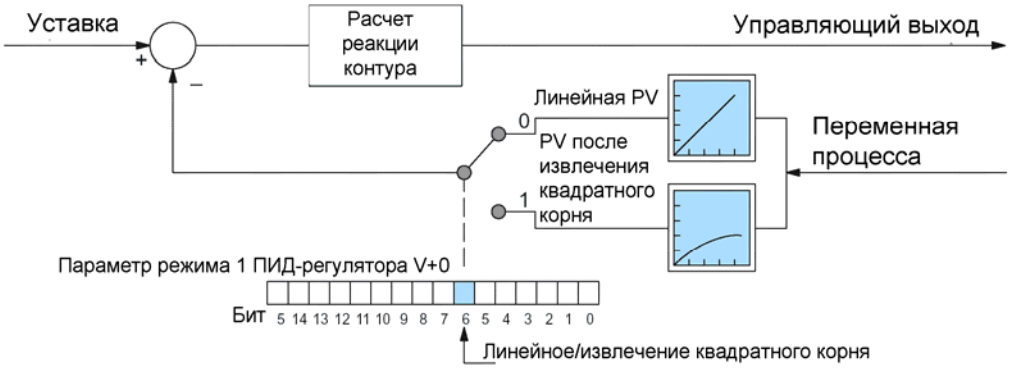


Диалоговое окно Loop Setup (Настройка контура) DirectSOFT32 позволит задать указатель удаленной уставки (Remote SP), если известен соответствующий адрес. Можно ввести значение указателя, также, с помощью ручного программатора или задать его программно с помощью инструкции LDA.

Конфигурирование переменной процесса (PV)

Вход переменной процесса каждого контура — это значение, которым контур в конечном счете пытается управлять, чтобы сделать его равным уставке, и как можно быстрее отслеживать изменения уставки. 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 addr+00 параметров режима ПИД-регулятора



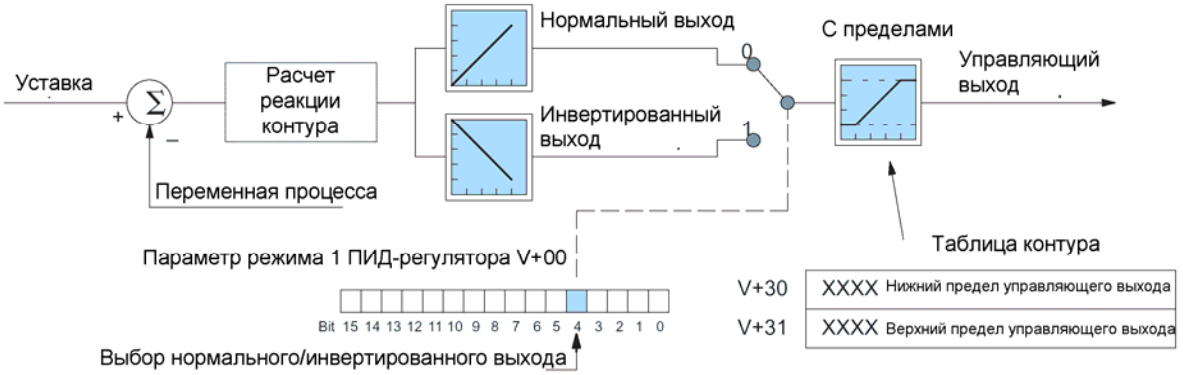
Важно. При использовании извлечения квадратного корня PV необходимо масштабировать шкалу SP, так как контур управляет выходом, так чтобы квадратный корень из PV равнялся входу PV. Разделите требуемое значение SP на квадратный корень из аналогового диапазона и используйте результат в ячейке addr+02 для SP. Это уменьшит разрешающую способность SP, но для большинства контуров управления расходом большая точность и не требуется (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-бит	SP = PV вход / 64	0 – 64	0 – 4095
15-бит	SP = PV вход / 181.02	0 – 181	0 – 32767
16-бит	SP = PV вход / 256	0 – 256	0 – 65535

Конфигурирование 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 программно нижним и верхним пределами. Значения addr+30 и addr+31 могут быть установлены один раз с помощью диалогового окна PID Setup (Настройка ПИД-регулятора) DirectSOFT32.

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 DirectSOFT32 «forward/reverse» — прямой/обратный). Для конфигурирования выхода используется бит 4 слова addr+00 параметров режима 1 ПИД-регулятора. Независимо от используемого формата (однополярный/биполярный) нормальный выход увеличивается при положительном значении рассогласования и уменьшается при отрицательном (где Рассогласование = (SP-PV)). Инвертированный выход меняет направление изменения выхода.

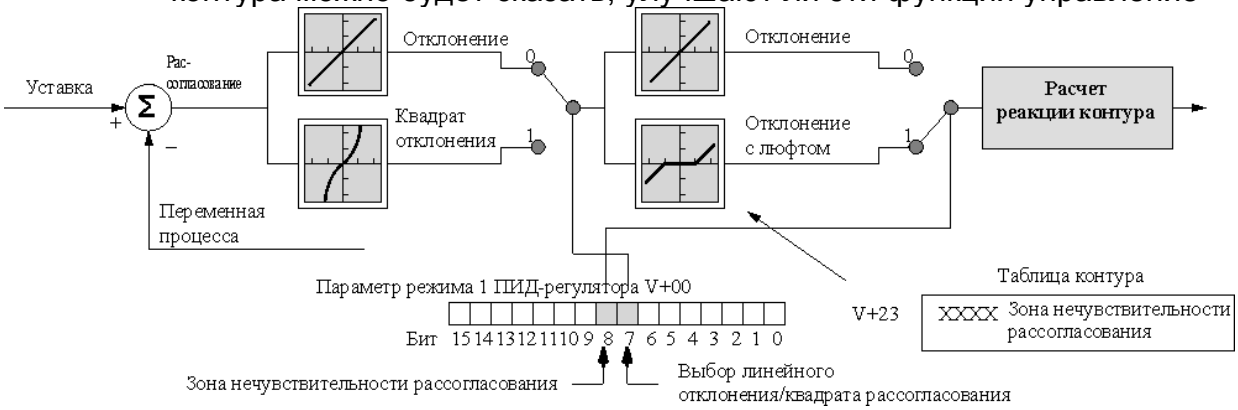
Выбор обычного/инвертированного выхода используется для настройки контуров с прямым/обратным действием. Этот выбор, в конце концов, определяется направлением отклика переменной процесса на изменение управляющего выхода в определенном направлении. Подробно контуры с прямым/обратным действием описаны в 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 вычитания: 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 addr+00 режима 1 ПИД-регулятора позволяет выбрать линейную или квадратичную состовляющую 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, химическое управление pH) лучшие результаты дают нелинейные контроллеры. Другим приложением является управление промежуточным резервуаром, для которого требуется сглаженный сигнал управляющего выхода.

Зона нечувствительности рассогласования (Error Deadband). При выборе этой опции 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.

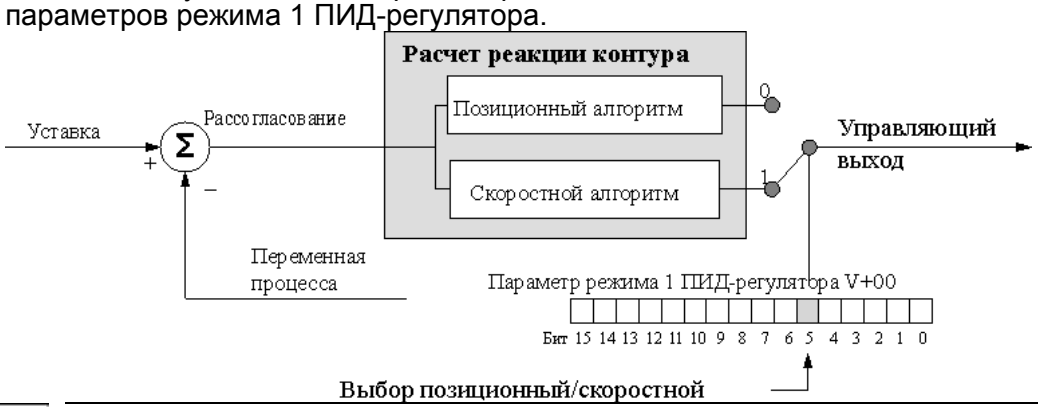
Алгоритмы ПИД-регулирования

Пропорционально-интегрально-дифференциальный (ПИД) алгоритм регулирования широко используется в управлении процессами. ПИД-метод управления хорошо адаптируется к электронным решениям, реализованы они на аналоговых или цифровых (процессорных) компонентах. Процессор DL06 реализует уравнение ПИД-регулирования цифровым образом, программно решая основное уравнение. Модули ввода/вывода служат только для преобразования электронных сигналов в цифровую форму (или наоборот).

DL06 обеспечивает 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. Управляющий выход рассчитывается так, чтобы он представлял скорость изменения значения PV, которое стремится быть равным 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) DirectSOFT32. Или используйте для выбора алгоритма, как показано ниже, бит 5 слова addr+00 параметров режима 1 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 Mn:

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

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

$Mx_o = M_o$

$Mx^n = K_i * e_n + Mx_{n-1}$

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

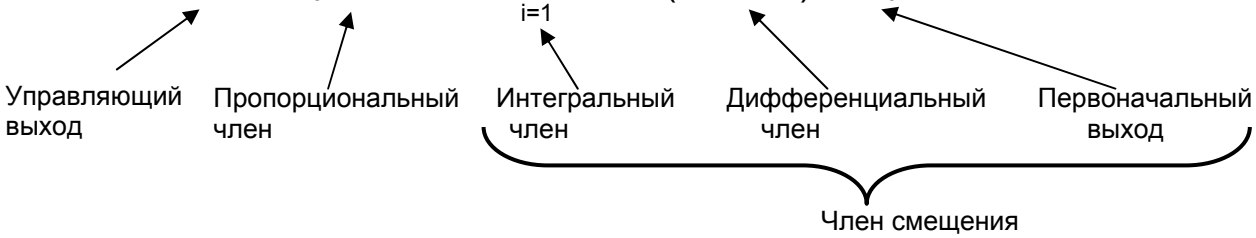
$M_n = K_c * e_n + K_r * (e_n - e_{n-1}) + Mx_n$…..Выход для времени опроса “n”

Ниже перечислены переменные позиционного алгоритма и связанные с ним переменные:

Ts = период опроса
Kc = пропорциональный коэффициент усиления
Ki = Kc * (Ts/Ti) коэффициент интегральной составляющей
Kr = Kc * (Td/Ts) коэффициент дифференциальной составляющей
Ti = время интегрирования
Td = время дифференциирования
SPn = уставка для опроса «n» (значение SP)
PVn = переменная процесса для опроса «n» (PV)
en = SPn - PVn = рассогласование для отсчета «n»
M0 = управляющий выход для опроса «0»
Mn = управляющий выход для опроса «n»

Анализ этих уравнений можно найти в большинстве хороших книг по управлению процессами. На первый взгляд, мы можем, как показано ниже, выделить части позиционного ПИД-алгоритма, соответствующие П-, И- и Д-членам, а также смещению(рабочей точке).

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



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

Скоростной алгоритм

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

Ниже перечислены переменные алгоритма в форме скорости и связанные с ним переменные:

Ts = период опроса
Kc = пропорциональный коэффициент усиления
Ki = Kc * (Ts/Ti) коэффициент интегральной составляющей
Kr = Kc * (Td/Ts) коэффициент дифференциальной составляющей
Ti = время интегрирования
Td = время дифференциирования
SPn = уставка для опроса «n» (значение SP)
PVn = переменная процесса для опроса «n» (PV)
en = SPn - PVn = рассогласование для опроса «n»
Mn = управляющий выход для опроса «n»

Окончательные уравнения для скоростного алгоритма ПИД-уравнения выглядят так:

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

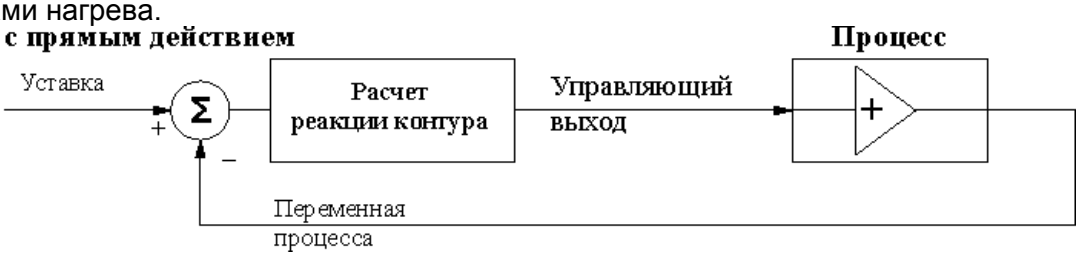
$$\Delta M_n = K_c * (e_n - e_{n-1}) + K_i * e_n + K_r * (e_{n-2} * e_{n-1} + e_{n-2})$$

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, которое мы называем «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.

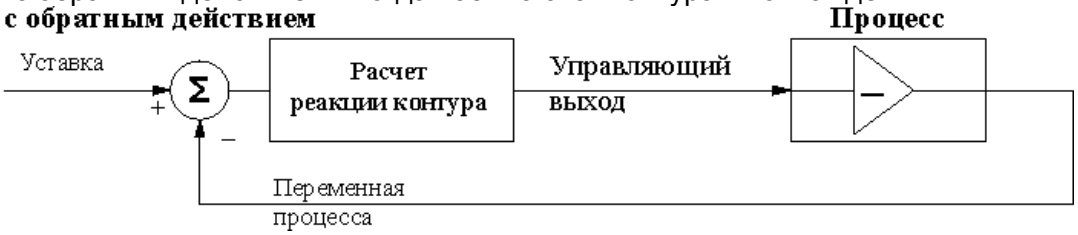
В большинстве случаев контуры процессов, например, температурные контуры, будут контурами с прямым действием. Рост подаваемого тепла приводит к росту PV (температуры). Соответственно, контуры с прямым действием иногда называются контурами нагрева.

Контур с прямым действием



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). Соответственно, контуры с обратным действием иногда называются контурами охлаждения.

Контур с обратным действием

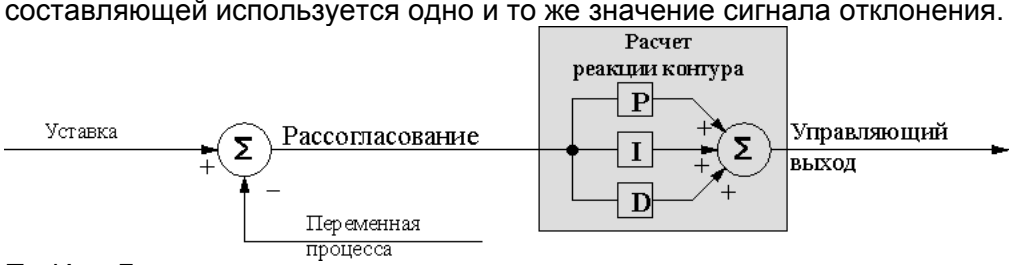


Для конкретного контура важно знать тип действия (прямое или обратное)! Если объект управления не является температурой, ответ не очевиден. Для контура управления потоком цепь позиционирования клапана может быть легко подключена и настроена как с прямым, так и с обратным действием. Простым способом найти 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 (конфигурирование управляющего выхода описано 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 настройке.

П-, И- и Д- составляющие контура

Вспомните позиционную и скоростную формы уравнения контура с ПИД-регулятором. В уравнениях обычно показаны 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 отклонения.

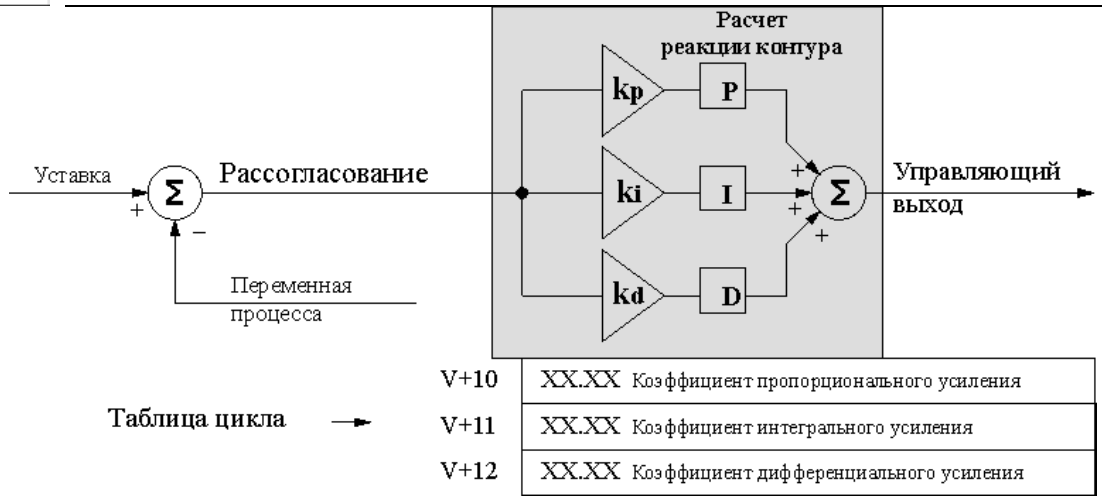


П-, И- и Д- составляющие при управлении играют следующие роли:

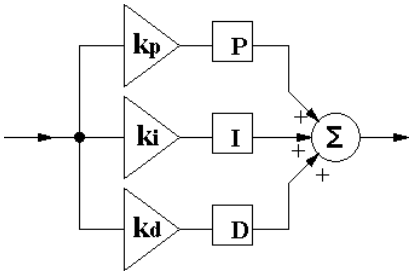
- **Пропорциональная**. Пропорциональная составляющая просто обеспечивает отклик, пропорциональный текущей величине рассогласования. Контроллер контура вычисляет значение пропорциональной составляющей при каждом расчете реакции ПИД-регулятора. Когда рассогласование равно нулю, пропорциональная составляющая 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 управляющий выход, требующийся для сохранения текущей позиции PV.
- **Дифференциальная**. Дифференциальная (Rate) составляющая отвечает за 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 программируемые значения коэффициентов усиления: k_p, k_i и k_d, соответственно. Значения размещаются в показанных ячейках таблицы контура. Цель процесса настройки контура (описываемого ниже) состоит в том, чтобы получить значения коэффициентов усиления, приводящие к хорошей общей эффективности контура.

Примечание. Пропорциональный коэффициент усиления в терминологии контура с ПИД-регулятором также называется просто «коэффициентом усиления».



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.



V+10	XX.XX	Коэффициент П-усиления	–
V+11	XX.XX	Коэффициент И-усиления	0=секунды, 1=минуты
V+12	XX.XX	Коэффициент Д-усиления	секунды

Параметры режима 2 ПИД-регулятора V+01

Бит 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Выбор единиц

В окне просмотра тренда пакета DirectSOFT32 можно задать значения коэффициентов усиления и единиц в реальном времени, во время работы контура. Это обычно делается только в процессе настройки контура.

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 усилением.

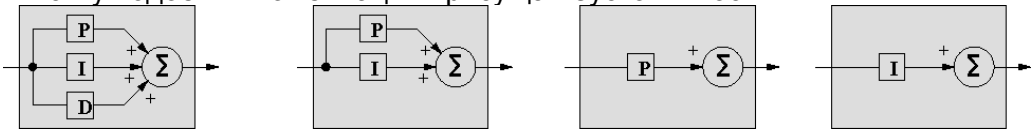
Интегральный коэффициент усиления. Диапазон значений от 0001 до 9998, но внутри значения используются как xx.xx. Задание «0000» или «9999»обнуляет интегральный коэффициент усиления, удаляя интегральную составляющую из уравнения ПИД-регулятора. Это позволяет подстроиться под приложения, для которых требуются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

Дифференциальный коэффициент усиления. 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 усилением. Имеется необязательная возможность ограничения дифференциального коэффициента усиления, обсуждаемая в следующем разделе.

Примечание. Очень важно знать, как правильно увеличивать и уменьшать коэффициенты усиления. Пропорциональный и дифференциальный коэффициенты 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.

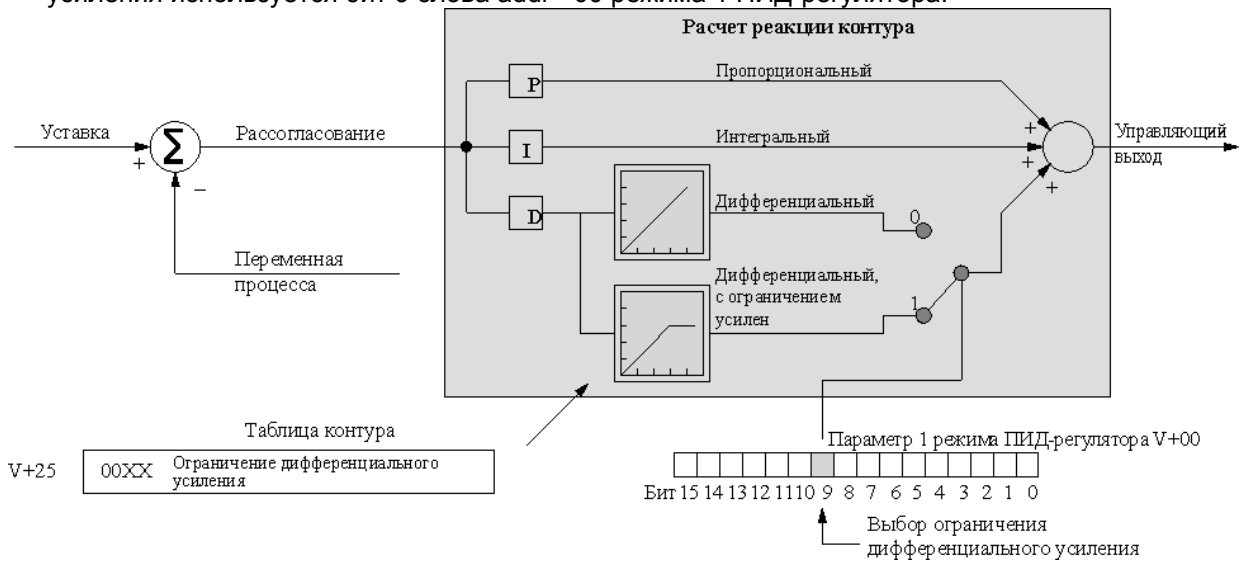
Использование подмножества управляющих элементов ПИД-регулятора

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



Ограничение дифференциального коэффициента усиления

Дифференциальная составляющая отличается тем, что имеет дополнительную возможность ограничения усиления. Это обусловлено тем, что дифференциальная составляющая плохо 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 addr +00 режима 1 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 важным компонентом значения управляющего выхода является составляющая смещения (Bias Term), 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

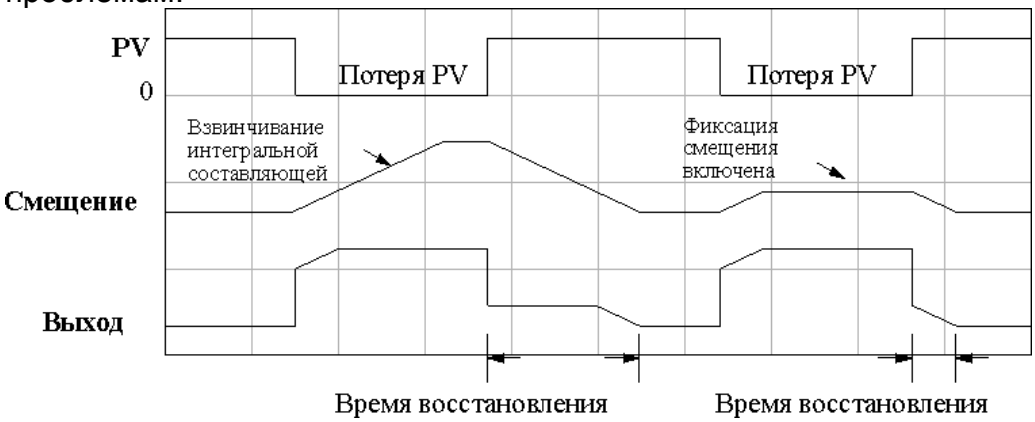
Член смещения

Если после двух или более периодов отсчетов отклонение (e_n) сводится к нулю, пропорциональная и дифференциальная составляющие обнуляются. Составляющая смещения равна сумме интегральной составляющей и начального выхода (M_o). Она представляет собой устойчивую, постоянную часть значения управляющего выхода и аналогична компоненту 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). Если сигнал PV пропадает, и значение PV становится равным нулю. Последствия этого серьезного сбоя — выход контура усиливается 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 достигает некоторых пределов. 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), или устанавливая бит 10 слова параметров 1 режима 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, а 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

Возможно, это самый важный этап управления процессом с замкнутым контуром. 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 ПИД-регулятора, автоматически рассчитывающий коэффициенты усиления (автоматическая настройка). У каждого опытного инженера есть свой любимый метод, и DL06 может подстроиться под любые запросы. Использование автоматической настройки может устранить большую часть ошибок ручного подхода применения методов 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.

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 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 данные PV, попадающие в ячейку addr+02 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 (возможно, 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 идеальную 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.

Начиная со следующей страницы, 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 DirectSOFT32.

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 процесса и добавит путаницу в процесс настройки контура.

- Настройте коэффициенты усиления, 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 DirectSOFT32 или ручного программатора, и т.д.

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

Предупреждение: Если 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 тумблера аварийной остановки, который управляет питанием исполнительного механизма контура, чтобы при необходимости отключить его.

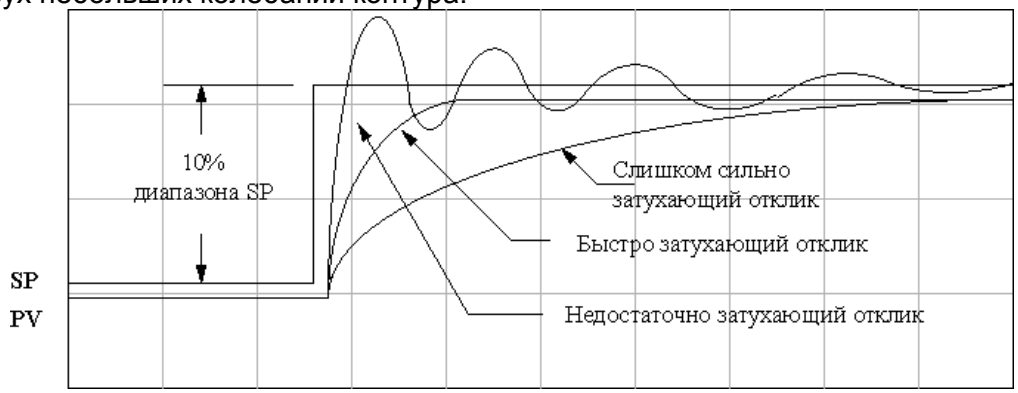
- В этот момент должно выполняться SP = PV благодаря функции безударного перехода. Немного увеличьте SP, создавая отклонение. Если активен только пропорциональный коэффициент усиления, и составляющая смещения равна 0, можно легко проверить значение управляющего выхода:

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 данного действия должно выполняться PV = SP или их значения должны быть очень близки.

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, поэтому постепенно увеличивайте их, сосредоточившись в первую очередь на коэффициенте 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 одной-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 осцилляций контура. Дифференциальное воздействие стремится в какой-то степени ослабить пропорциональный отклик, поэтому настраивайте их совместно.

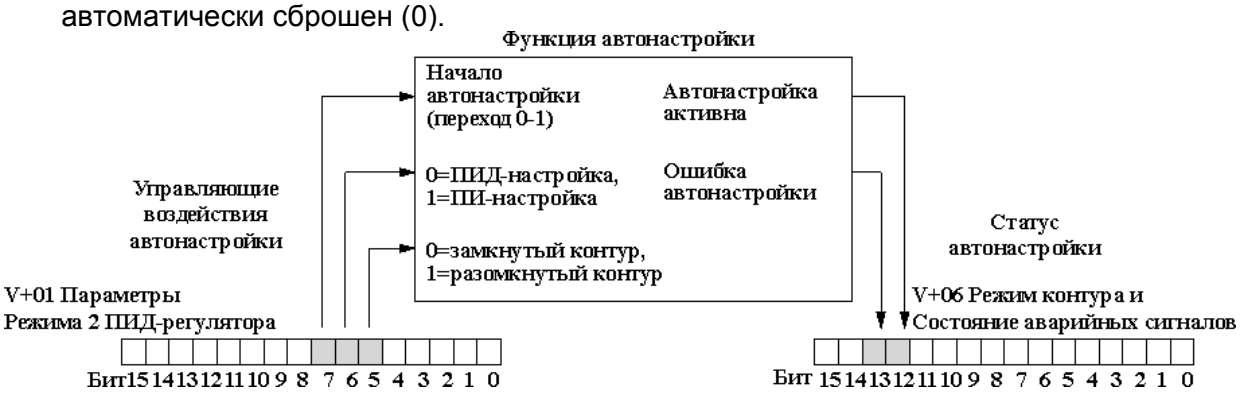
Процедура автонастройки

Автонастройка запускается из DirectSOFT32. Вы можете использовать автонастройку, чтобы установить начальные значения параметров ПИД-регулятора (автонастройка, не действует непрерывно). Всякий раз, когда происходит существенное изменение в динамике контура (обрабатываемой массы, размер испольнительного механизма, и т.д.), понадобится повторно выполнить процедуру настройки, чтобы получить новые коэффициенты усиления, требуемые для оптимального управления.

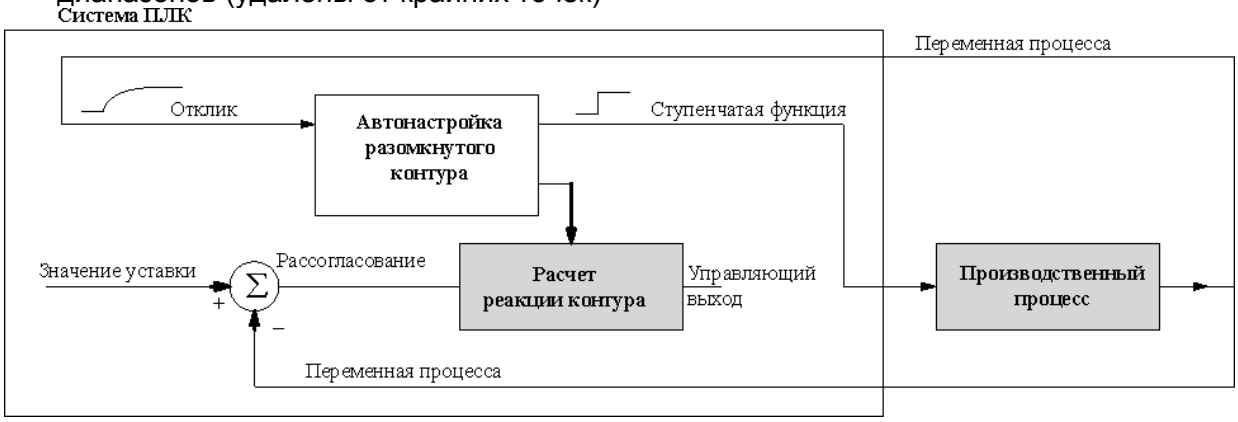
Предупреждение. Изменения, которые влияют на настройку контура, должны выполняться только уполномоченным персоналом, в совершенстве знакомым со всеми аспектами процесса. Применение процедур автонастройки контура оказывает влияние на процесс, в том числе вызывая большие изменения значения управляющего выхода. Чтобы минимизировать опасность травмирования персонала или повреждения оборудования, убедитесь, что вы тщательно проанализировали последствия любых изменений. Автонастройка в DL06 не заменяет необходимость знания 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 addr+01 режима 2 ПИД-регулятора, как показано ниже. При использовании функции автонастройки пакета DirectSOFT32 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 между ПИД- и ПИ-настройкой и между настройкой с разомкнутым и замкнутым контуром. При выборе ПИ-настройки процедура автонастройки приравнивает дифференциальный коэффициент 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)

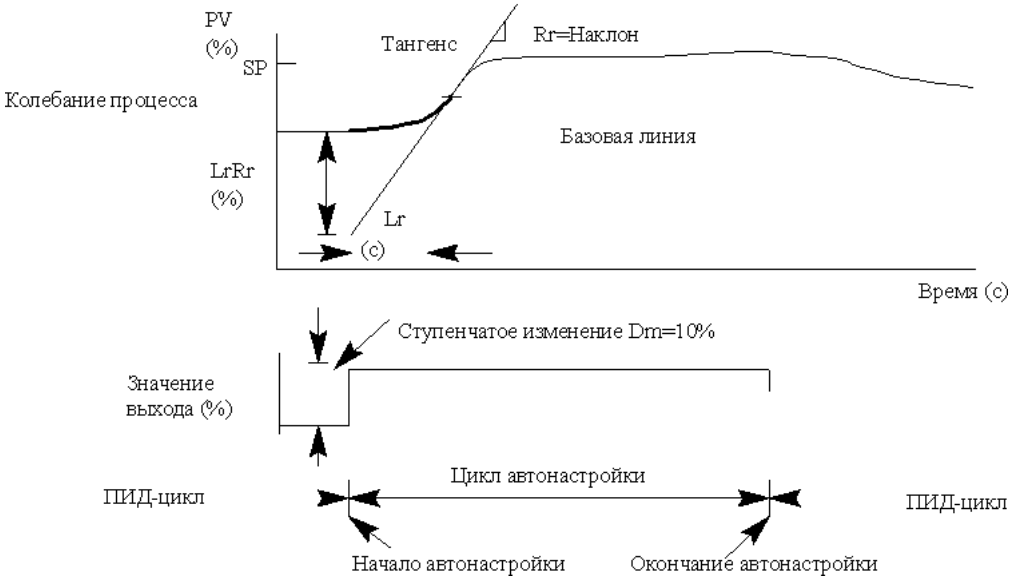


Примечание. По теории в данном случае значение SP не важно, так как контур не замкнут. Однако программное обеспечение ПЛК требует, чтобы перед началом цикла автонастройки значение SP отстояло от значения PV не менее чем на 205 единиц счета (на 205 и более единиц счета ниже SP для контуров с прямым действием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

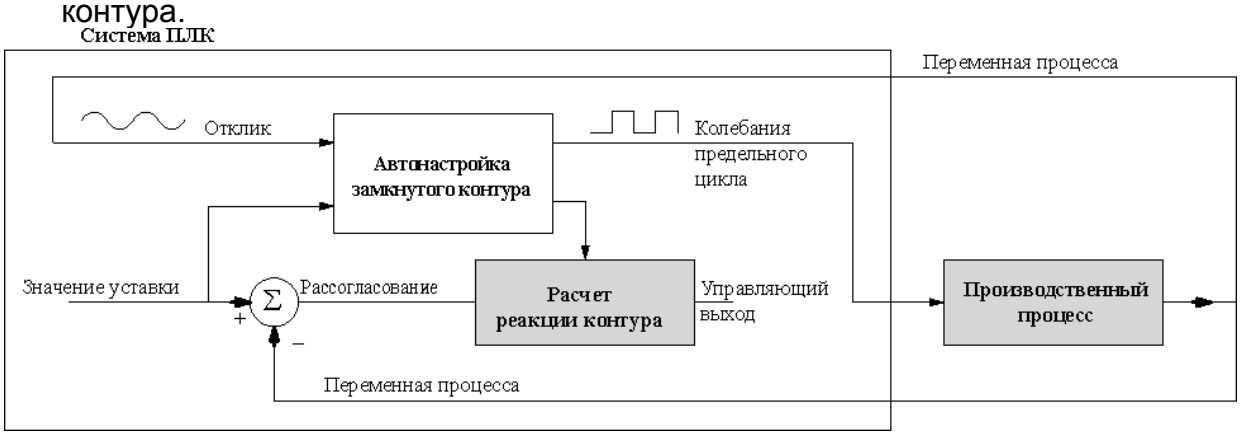


*В начале автонастройки происходит ступенчатое изменение выхода на 10%
*В ходе автонастройки выход контроллера достигает положительного предела диапазона. Автонастройка прекращается и в слове Аварийных сигналов устанавливается бит Ошибка автонастройки.
*Если изменение PV меньше 2%, выход меняется на 20%.
По завершении наблюдений за настройкой контура контроллер контура 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 DirectSOFT32. 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.

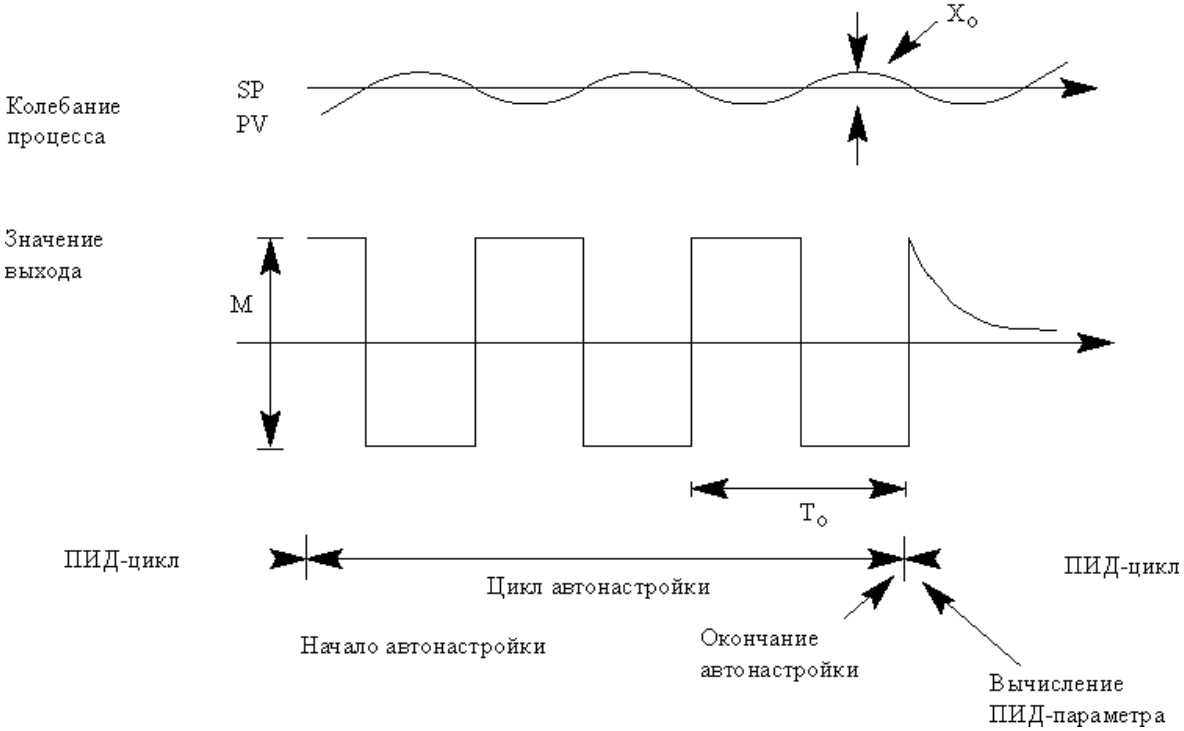
Автонастройка замкнутого контура. На приведенной ниже схеме показана работа контроллера контура в течение цикла автонастройки замкнутого контура.



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 повторяется три полных цикла.

Цикл автонастройки замкнутого контура: Метод предельного цикла



*Mmax = верхний предел значения выхода Mmin = 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:

Kpc = 4M / (π * Xo) Tpc = 0	
M = амплитуда выхода	
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 установлен бит ошибки автонастройки (бит 13 слова Режима контура и статуса аварийных сигналов addr+06), пожалуйста, проверьте, что значения PV и SP находятся в пределах 5% полного диапазона изменений, как требует функция автонастройки. Этот бит также устанавливается, если при использовании метода замкнутого контура выход выходит за пределы диапазона.

Примечание: если переменная колеблется сильно, вам, вероятно, необходим встроенный аналоговый фильтр (см. стр. 8-47) или необходимо создать программный фильтр (см. стр. 8-48).

Настройка каскадных контуров

При настройке каскадных контуров нам понадобится устранить каскадную связь и выполнить независимую настройку контуров с помощью одной из ранее описанных процедур.

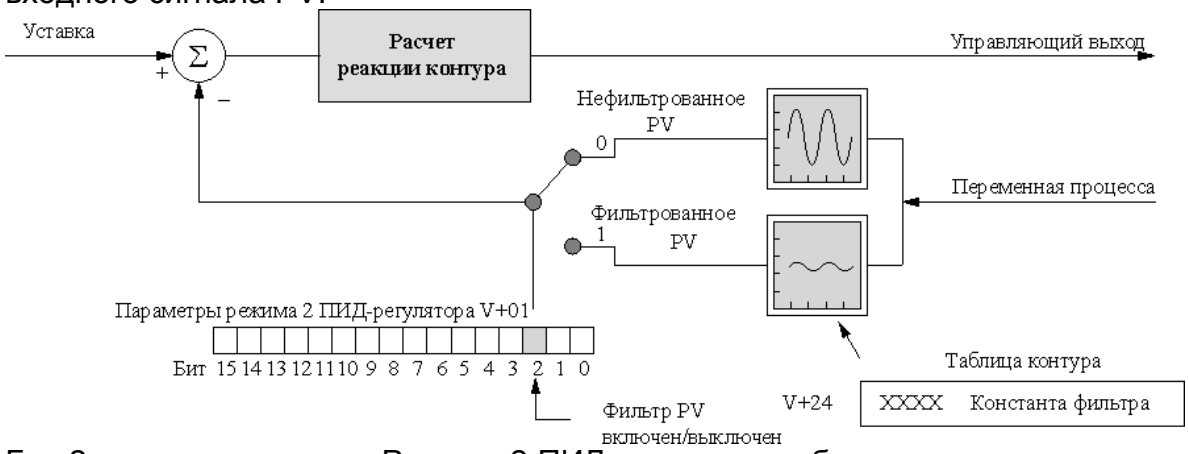
1. Если автонастройка не используется, то с помощью метода, описанного выше в этой главе, найдите частоту опроса для подчиненного контура. Затем установите частоту опроса главного цикла в 10 раз меньше частоты подчиненного цикла. Используйте эти значения в качестве начальной точки.
2. Сначала настройте подчиненный контур. Оставьте главный контур в Ручном режиме и создайте изменения SP для подчиненного цикла вручную, как описано в процедуре настройки контура.
3. Убедитесь, что в Автоматическом режиме будет быстро затухать отклик подчиненного контура на 10% изменения SP. На этом настройка подчиненного контура заканчивается.
4. На этом этапе необходимо перевести подчиненный контур в Каскадный режим, а затем главный контур — в Автоматический режим. Мы будем настраивать главный контур, рассматривая подчиненный контур как последовательный компонент общего процесса. Следовательно, при настройке главного контура не нужно возвращаться к настройке подчиненного контура.
5. Настройте главный цикл, следуя стандартной процедуре настройки, описанной в данном разделе. Отклик PV главного контура в действительности является полным откликом каскада контуров.

Аналоговая фильтрация переменной

Зашумленный сигнал PV затрудняет настройку и может быть причиной более критических значений управляющего выхода, чем требуется, так как выход реагирует на минимумы и максимумы PV. Для придания контуру большей устойчивости существуют два эквивалентных метода фильтрации входа переменной PV: использование встроенного аналогового фильтра или создание программного фильтра.

Встроенный аналоговый фильтр DL06

DL06 имеет встроенный низкочастотный фильтр первого порядка для PV, который можно использовать при автонастройке методом замкнутого контура. Как показано на следующем рисунке, мы настоятельно рекомендуем использовать этот фильтр во время автонастройки. После автонастройки фильтр можно отключить или продолжать использовать для зашумленного входного сигнала PV.



Бит 2 слова параметров Режима 2 ПИД-регулятора обеспечивает управление включением/выключением низкочастотного фильтра PV (0=выключен, 1=включен). Частота спада (сглаживание) однополюсного низкочастотного фильтра задается с помощью регистра addr+24 таблицы параметров цикла, константы фильтра. Эта константа задается в формате BCD, с подразумеваемой десятичной точкой 00X.X следующим образом:

- Константа фильтра может принимать значения от 000.1 до 001.0.
- *DirectSOFT32* преобразует значения выше действующего диапазона к 001.0 и значения ниже этого диапазона к 000.1
- Установка значения 000.0 или от 001.1 до 999.9 полностью отключает фильтр.
- Значения, близкие к 001.0, дают высокие частоты спада, а значения, близкие к 000.1, дают низкие частоты спада.

Настоятельно рекомендуется использовать в качестве интерфейса автонастройки пакет *DirectSOFT32*. Длительность цикла автонастройки будет зависеть от инерционности процесса. Медленное изменение PV приведет к увеличению длительности цикла автонастройки.

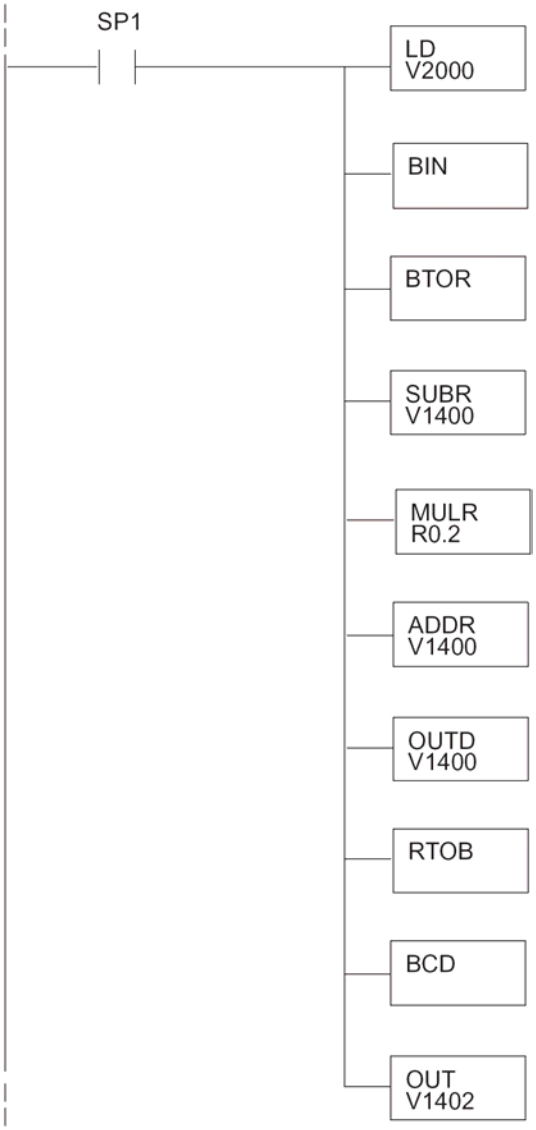
По окончании автонастройки значения пропорционального, интегрального и дифференциального коэффициентов усиления в ячейках таблицы контура addr+10, addr+11 и addr+12, соответственно, будут автоматически обновлены. Кроме того, автоматически обновляется период отсчетов в addr+07. Правильность результатов 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

Создание аналогового 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 необходимую релейную логику. Не забудьте изменить адреса ячеек памяти в соответствии с вашим приложением.

Фильтрация может внести одну тысячную долю погрешности из-за округления. Если ваш процесс не допускает такой погрешности - не используйте фильтрацию. Из-за ошибки округления, вы не сможете использовать ноль и конец шкалы, как точки аварийной сигнализации. Кроме того, уменьшение коэффициента 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

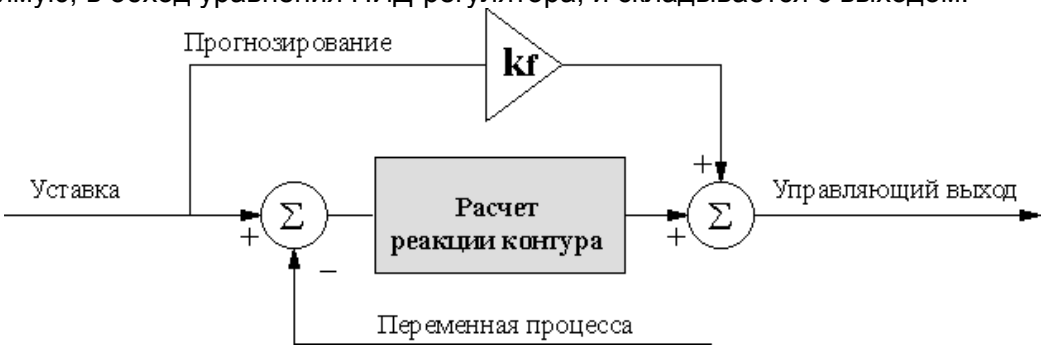
Преобразование вещественного числа в аккумуляторе в двоичное значение, и сохранение результата в аккумуляторе

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 PID контуре

Управление с упреждением

Управление с упреждением (по возмущению) – Feedforward Control представляет собой улучшение стандартного управления в замкнутом контуре. Оно подходит для уменьшения влияния, поддающегося количественному определению, и предсказуемого возмущения или резкого изменения уставки. DL06 поддерживает 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.

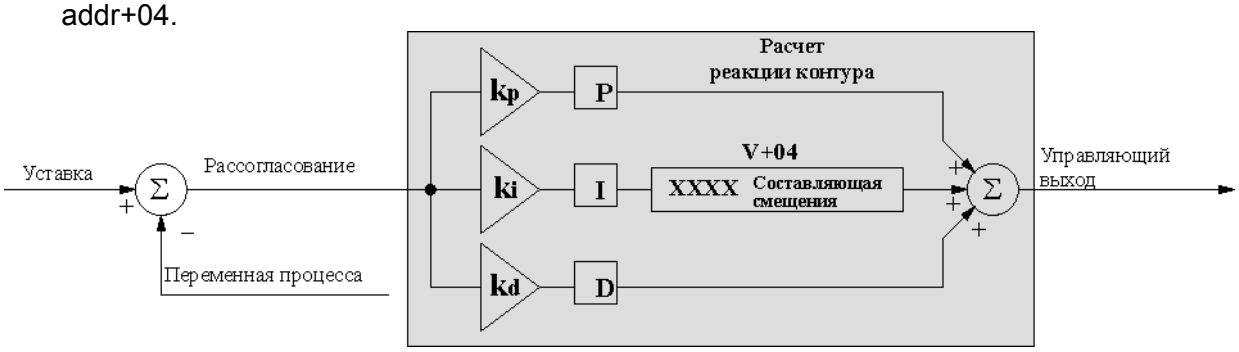


В предыдущем разделе при рассмотрении составляющей смещения, говорилось, что значение этой составляющей задает для управляющего выхода «рабочую область» или рабочую точку. При колебании рассогласования вокруг нулевого значения, величина выхода колеблется вокруг значения смещения. Теперь при изменении уставки 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 пошагово увеличиваться/уменьшаться до исчезновения рассогласования, и так будет обнаружена новая рабочая точка.

Предположим, что нам известно о резком изменении уставки (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 уставки рабочая точка (значение смещения), можно искусственно прямо изменить выход (что и является упреждением). Упреждение обеспечивает следующие преимущества:

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

В контроллере контура DL06, 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 управления 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 по возмущению

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.

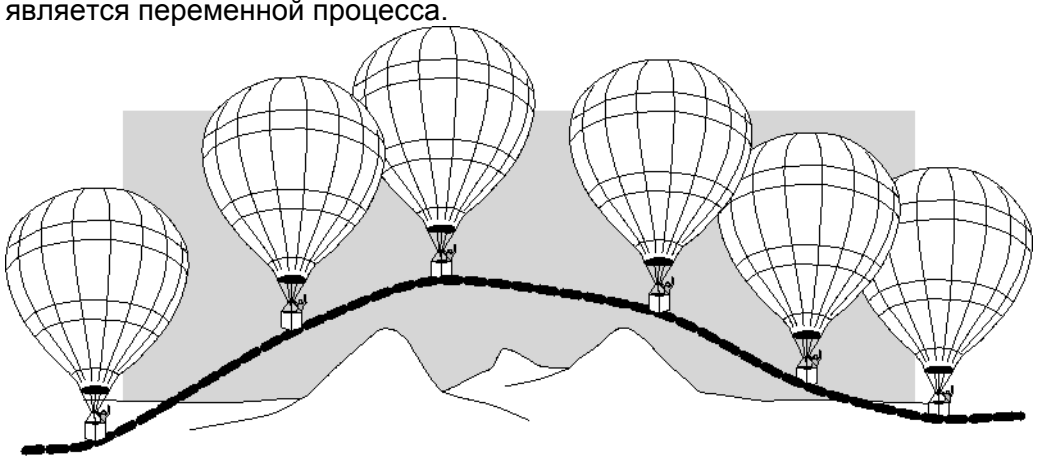
Сначала запишем величину изменения смещения, создаваемого контроллером контура при открытии или закрытии двери. Затем напишем программу, контролирующую положение концевого выключателя двери печи. При открытии двери наша программа считывает текущее значение смещения из addr+04, добавляет нужное изменение 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.

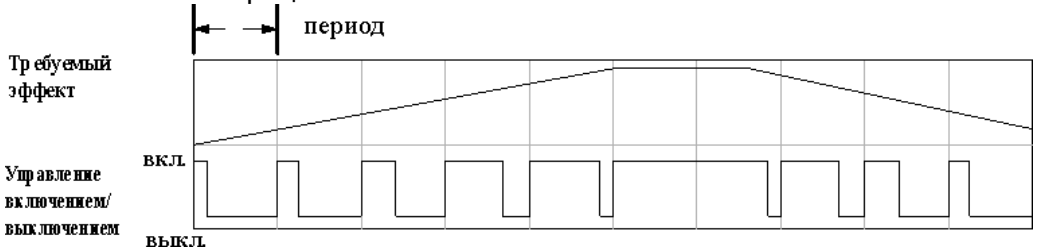
Широтно-импульсное управление

Контроллер контура ПИД-регулирования DL06 создает гладкий сигнал управляющего выхода 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.



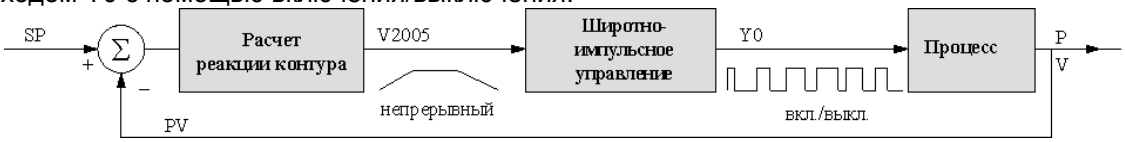
Широтно-импульсное управление является приближением непрерывного управления за счет особенностей рабочего цикла — отношения времени включения к времени выключения. На следующем рисунке приведен пример того, как рабочий цикл обеспечивает приближение к непрерывному уровню при усреднении с помощью большой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



Если бы мы нарисовали времена включения/выключения горелки шара с горячим воздухом, мы, вероятно, обнаружили бы, что этот процесс очень похожим образом связан с температурой и высотой шара.

Пример программы с дискретным управлением

В следующем примере программа реализует функцию дискретного управления (On/OFF Control). Программа преобразует непрерывный выход в V2005 в управление дискретным выходом Y0 с помощью включения/выключения.

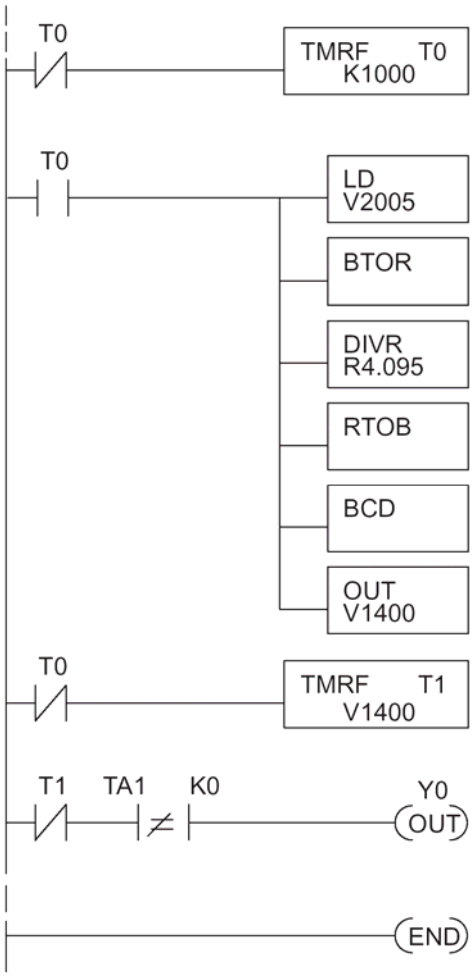


Программа данного примера для управления с помощью включения/выключения использует два таймера. Используются следующие предположения, которые могут меняться для конкретного приложения:

- 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.

Коэффициент заполнения временной диаграммы Y0 соответствует значению управляющего выхода в V2005 с периодом 10 с.

Примечание: Некоторые процессы изменяются слишком быстро в течение времени одного цикла управления. Рассмотрите скорость вашего процесса, когда Вы выбираете этот метод управления. Используйте непрерывное управление для процессов, которые изменяются слишком быстро за период одного цикла управления.



Использует для основной временной развертки быстрый таймер (с разрешением 0.01 с). K1000 обеспечивает предварительно установленное значение 10 с. Нормально-закрытый контакт T0 реализует автоматический сброс. T0 каждые 10 секунд истинно для одного сканирования.

В начале 10-секундного периода T0 истинно. Из ячейки V+05, V2005, таблицы контура загружается двоичное значение управляющего выхода.

Преобразует значение аккумулятора в вещественное число,

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

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

Выводит результат в V1400. Это произвольно заданная ячейка, содержащая уставку для второго таймера.

Второй таймер также подсчитывает тики 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 подается на исполнительный механизм, нагреватель, и т.п. контура.

Обмотка END обозначает окончание основной программы

Каскадное управление

Введение

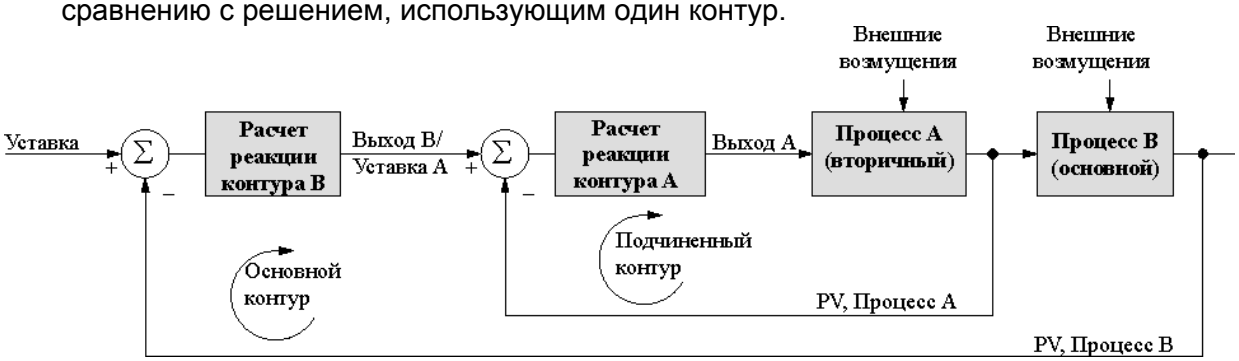
Каскадные контуры представляют собой усовершенствованный метод управления, в определенных ситуациях более выгодный, чем управление с помощью отдельных контуров. Как следует из названия, каскад означает, что один контур подключен к другому. Помимо ручного (разомкнутый контур) и автоматического (замкнутый контур) режимов DL06 также обеспечивает каскадный режим работы ПИД-контуров.

Примечание. Каскадные контуры — это более сложный метод управления. Поэтому их использование рекомендуется только для управления процессом опытными инженерами.

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 источники 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 уставка для подчиненного контура создается автоматически, с помощью выхода основного контура. После программирования и отладки каскадного управления предстоит работать только с первоначальными уставкой 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.

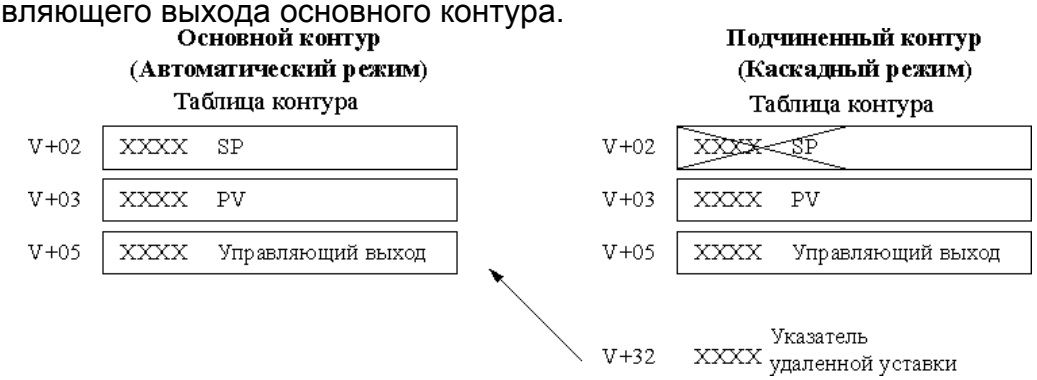
Каскадные контуры в процессоре DL06

Используя термин «каскадные контуры», необходимо учитывать важное замечание. Реально только подчиненный цикл 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 DL06 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 полярность.

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 второстепенный контур будет игнорировать собственный локальный регистр уставки SP (addr+02), считывая в качестве своей уставки значение управляющего выхода основного контура.



При использовании для просмотра значения SP подчиненного контура окна Просмотр ПИД-регулятора (PID View) DirectSOFT32 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.

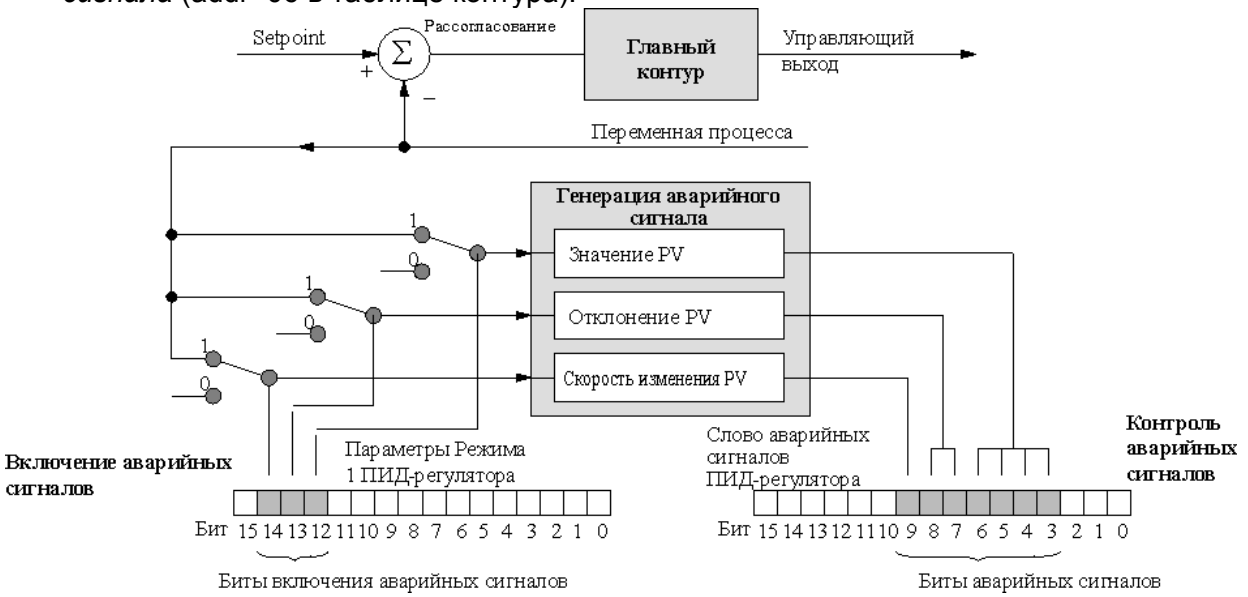
Аварийные сигналы управления процесса

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 неисправность не будет устранена.

Процессор DL06 обладает развитым набором функций аварийных сигналов для каждого контура:

- **Аварийные сигналы абсолютного значения PV.** Отслеживает 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. Аварийные сигналы генерируются, 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

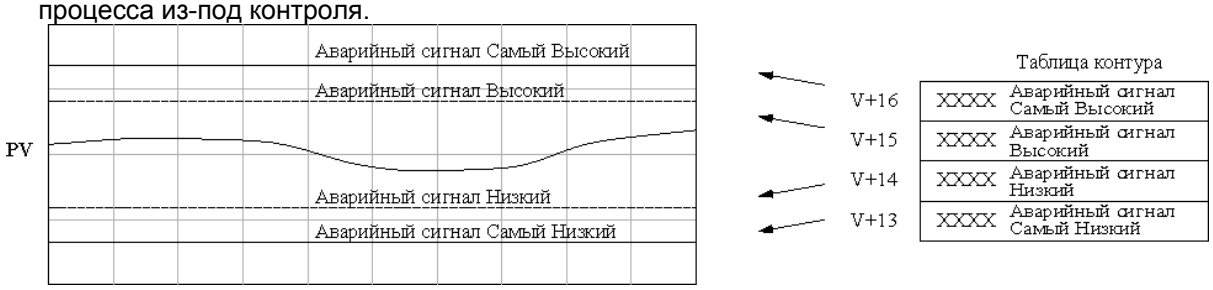
Пороги сигналов являются полностью программируемыми, и каждый тип аварийного сигнала может включаться и контролироваться независимо. На следующей схеме показана функция контроля PV. Аварийные сигналы включают/выключают биты 12, 13 и 14 слова addr+00 параметров Режима 1 ПИД-регулятора в таблице параметров контура. Диалоговые окна настройки Просмотр ПИД-регулятора (PID View) в DirectSOFT32 позволяют легко программировать, включать и контролировать аварийные сигналы. Программа может контролировать статус аварийного сигнала, проверяя биты 3-9 слова *Режим ПИД-регулятора* и слово *Статус аварийного сигнала* (addr+06 в таблице контура).



В отличие от расчета ПИД-коэффициентов аварийные сигналы работают всегда, когда процессор находится в рабочем режиме (RUN). Контур может находиться в ручном, автоматическом или каскадном режиме, и аварийные сигналы будут работать, если установлены соответствующие биты (см. 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 Самый Высокий (High-High Alarm и Самый Низкий (Low-Low Alarm). 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 при помощи команд бит-из-слова. Кроме того, аварийные сигналы ПИД-регулятора можно контролировать, используя DirectSOFT32.

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 колеблятся 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 рассогласования PV индицируются двумя битами слова Режим ПИД-регулятора и слова Статус аварийных сигналов в таблице контура, как показано справа. Настоятельно рекомендуется контролировать их в программе.

Контролировать эти биты легко с помощью команд бит-из-слова. Кроме того, аварийные сигналы ПИД-регулятора можно контролировать, используя DirectSOFT32

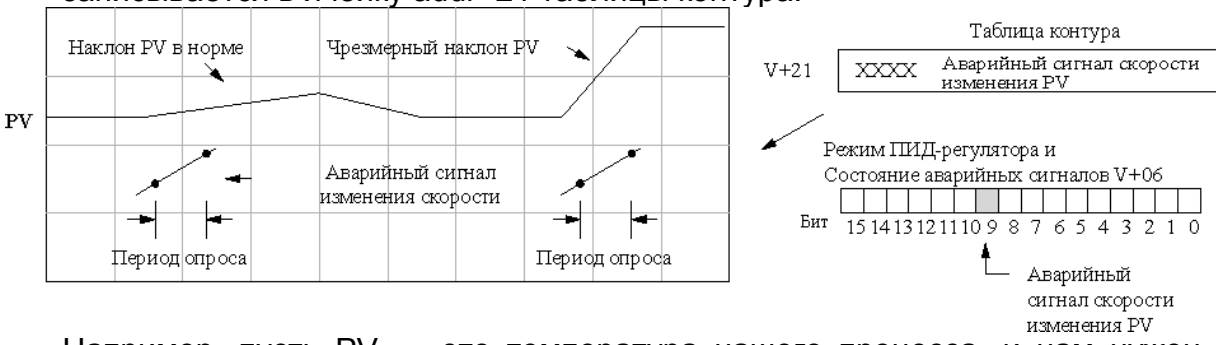
Аварийный сигнал рассогласования PV может включаться и отключаться независимо от других аварийных сигналов PV с помощью бита 13 слова addr+00 параметров режима 1 ПИД-регулятора.

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

Аварийный сигнал скорости изменения PV

Мощным средством раннего оповещения о нарушениях в технологическом процессе является контроль скорости изменения переменной PV. Инерция большинства технологических процессов велика, и значения PV меняются медленно. Относительно быстрое изменение PV является результатом разрыва 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 DL06 позволяет, как показано ниже, программировать 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 частоту 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:

$$\frac{\text{Аварийное значение}}{\text{Скорости смещения}} = \frac{15 \text{ градусов}}{1 \text{ минута}} = \frac{10 \text{ значений/градус}}{30 \text{опросов/минута}} = \frac{150}{30} = 5 \text{ значений/опрос}$$

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 addr+00 параметров режима 1.

Функция гистерезиса (задержки) сигнала, обсуждаемая ниже, не влияет на Аварийный сигнал скорости изменения.

Гистерезис (Hysteresys) аварийных сигналов PV

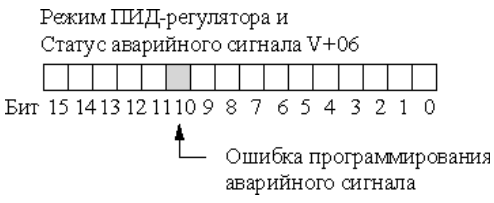
Аварийные сигналы абсолютного значения PV и рассогласования PV программируются с помощью пороговых значений. Когда абсолютное значение или рассогласование превышает порог, устанавливается бит статуса аварийного сигнала. Реальные сигналы PV подвержены некоторому шуму, который приводит к некоторым колебаниям 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 аварийных сигналов PV.

Значение гистерезиса аварийных сигналов PV принимает значения от 1 до 200 (шестнадцатеричное). При использовании аварийного сигнала рассогласования 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

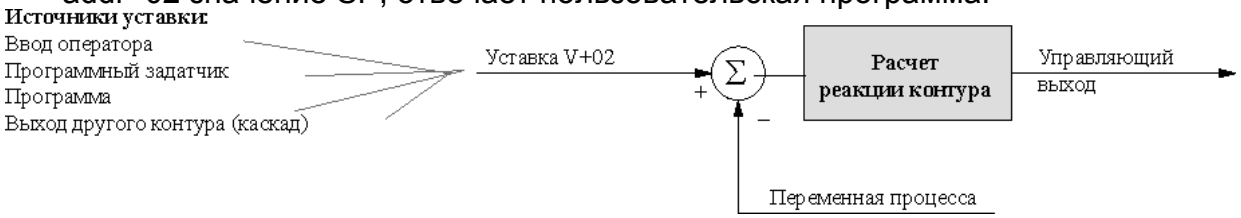
Требования к аварийному сигналу рассогласования PV:

Желтый < красный

Программный задатчик

Введение

При описании работы контура отмечалось, что уставка 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.

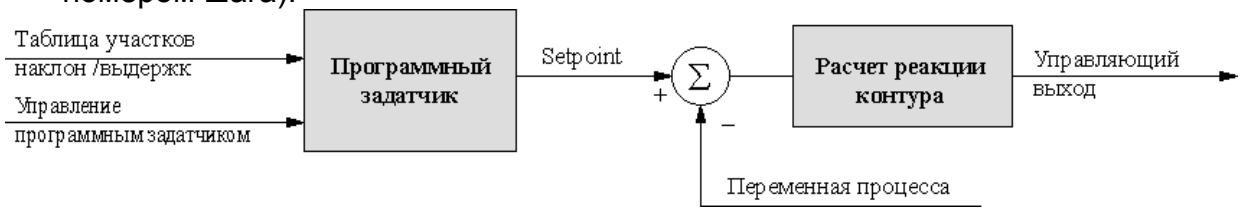


Если SP процесса меняется редко или допускает ступенчатые изменения, использование программного задатчика, возможно, не потребуется. Однако для некоторых процессов потребуется точно управлять изменениями значения SP. Программный задатчик может значительно снизить объем программирования, нужный для подобных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

В управлении процессами термины «наклонный участок» («ramp») и «участок выдержки» («soak») имеют специальные значения. На рисунке справа уставка возрастает на наклонном участке и остается постоянной на участке выдержки.

Сложные профили SP могут быть созданы с помощью задания последовательности сегментов. Наклон линейных участков задается в единицах SP в секунду. Длительность участка выдержки задается и 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 изменения сигнала, содержащую значения, которые будут определять профиль сигнала наклон/выдержка. По мере необходимости контур считывает эти значения из таблицы при каждом расчете ПИД-коэффициентов. За управление программным задатчиком отвечают биты в специальном слове таблицы контура, в реальном времени управляющие возможностью запускать/останавливать программный задатчик. Программа может следить за состоянием профиля сигнала (текущим номером шага).



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 в режиме RUN ПЛК включают Start (запуск), Hold (остановка), Resume (продолжение) и Jog (толчок).
- Контроль за работой 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.

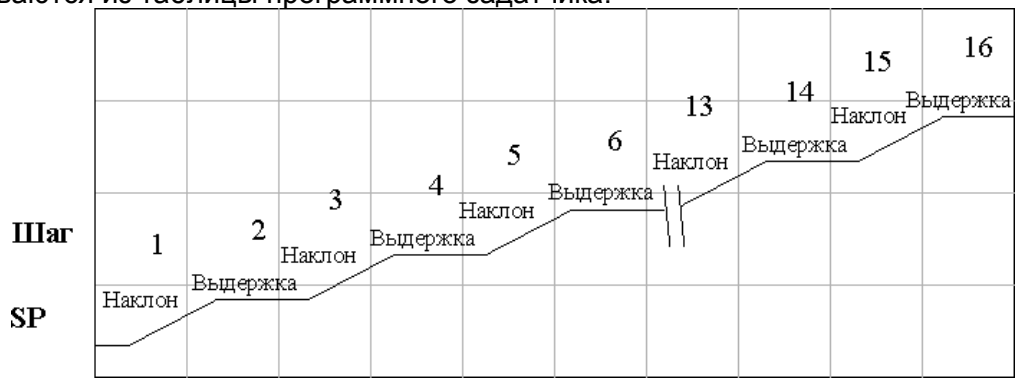
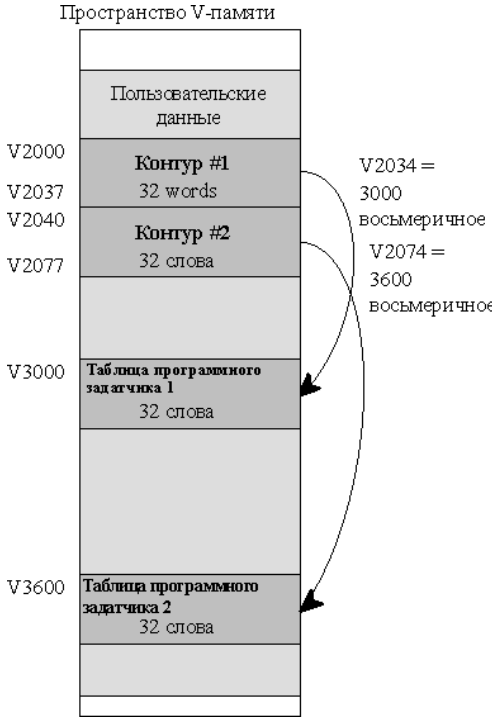


Таблица программного задатчика

Параметры, определяющие профиль SP, 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 располагающейся 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 DirectSOFT32, включающий специальный редактор для этой таблицы. Для определения пары участков наклон/выдержка требуется задать четыре параметра, как показано на 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.

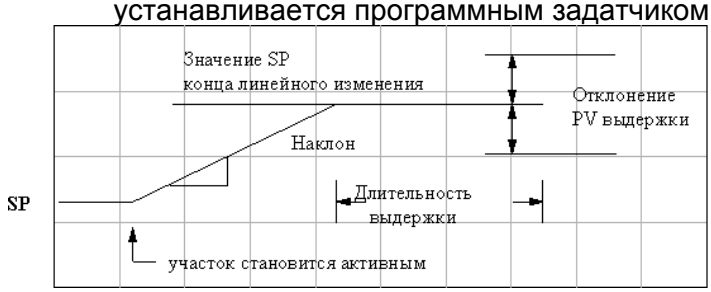


Таблица программного задатчика

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
+ 00	1	Значение SP окончания участка наклона
+ 01	1	Крутизна наклона
+ 02	2	Длительность участка выдержки
+ 03	2	Отклонение PV на участке выдержки
+ 04	3	Значение SP окончания участка наклона
+ 05	3	Крутизна наклона
+ 06	4	Длительность участка выдержки
+ 07	4	Отклонение PV на участке выдержки
+ 10	5	Значение SP окончания участка наклона
+ 11	5	Крутизна наклона
+ 12	6	Длительность участка выдержки
+ 13	6	Отклонение PV на участке выдержки
+ 14	7	Значение SP окончания участка наклона
+ 15	7	Крутизна наклона
+ 16	8	Длительность участка выдержки
+ 17	8	Отклонение PV на участке выдержки

Смещение адреса	Шаг	Описание
+ 20	9	Значение SP окончания участка наклона
+ 21	9	Крутизна наклона
+ 22	10	Длительность участка выдержки
+ 23	10	Отклонение PV на участке выдержки
+ 24	11	Значение SP окончания участка наклона
+ 25	11	Крутизна наклона
+ 26	12	Длительность участка выдержки
+ 27	12	Отклонение PV на участке выдержки
+ 30	13	Значение SP окончания участка наклона
+ 31	13	Крутизна наклона
+ 32	14	Длительность участка выдержки
+ 33	14	Отклонение PV на участке выдержки
+ 34	15	Значение SP окончания участка наклона
+ 35	15	Крутизна наклона
+ 36	16	Длительность участка выдержки
+ 37	16	Отклонение PV на участке выдержки

Флаги таблицы программного задатчика

Для многих приложений все 16 участков не нужны. В таком случае заполните неиспользуемые шаги таблицы нулями. Программный задатчик закончит свою работу, как только обнаружит крутизну участка наклона = 0. Описания отдельных битов слова Флагов таблицы программного задатчик (Addr+33) приведены в следующей таблице.

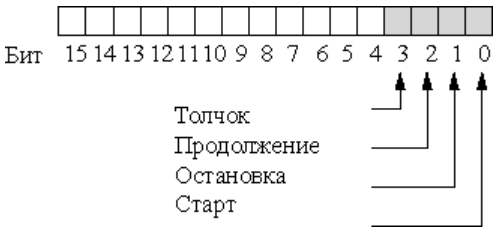
Бит	Описание бита флагов программного задатчикаRamp / Soak Flag Bit Description	Чтение/ запись	Бит = 0	Бит = 1
0	Начать работу программного задатчика	запись	–	0⇒1 Старт
1	Приостановить профиль программного задатчика	запись	–	0⇒1 Остановка
2	Продолжить профиль программного задатчика	запись	–	0⇒1Продолжение
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 (шестнадцатеричное)	

Включение программного задатчика

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

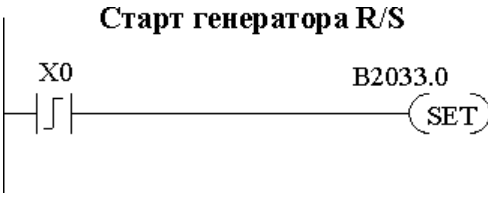
Средства управления программным задатчиком

Четыре основных управляющих воздействия на программный задатчик реализуются с помощью битов 0 - 3 слова параметров сигнала наклон/выдержка в таблице параметров контура. DirectSOFT32 непосредственно управляет этими битами, используя диалог параметров программного задатчика. Однако в ходе выполнения программы эти биты должны управляться программно. Рекомендуем использовать команды типа - бит из слова.



Для выполнения нужной функции 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 используется команда Set Bit-jf Word (бит из слова)

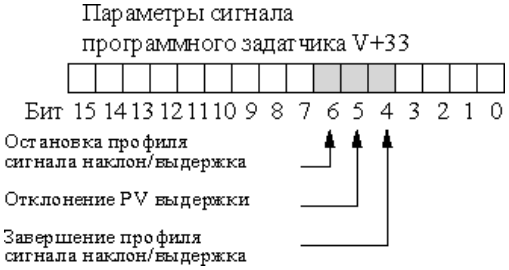


В нормальном состоянии все биты управления программного задатчика равны 0. Программа в каждый момент времени должна устанавливать не более одного бита управления.

- **Старт (Start)**. Переход 0 - 1 запустит программный задатчик. Процессор 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 (Hold)**. Переход 0 - 1 остановит программный задатчик в текущем состоянии, при этом фиксируется значение SP.
- **Продолжение (Resume)**. Переход 0 - 1 инициирует продолжение работы программного задатчика, если он был в состоянии Остановки. Значение SP будут равно своему предыдущему значению.
- **Толчок (Jog)**. Переход 0 - 1 заставляет программный задатчик прервать текущий участок (шаг) и перейти к следующему участку.

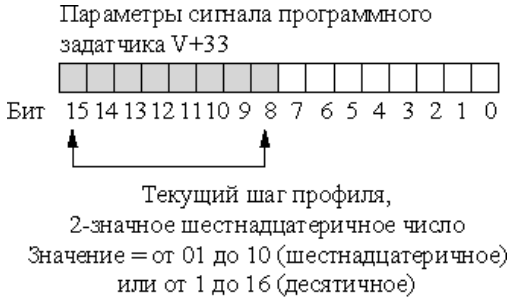
Мониторинг профиля сигнала наклон/ выдержка

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



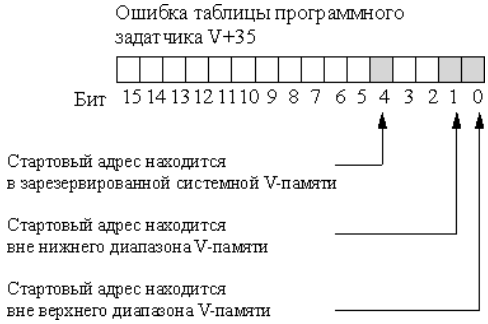
- **Завершение профиля программного задатчика**. Равен 1, если программа выполнила последний запрограммированный шаг.
- **Отклонение PV на участке выдержки**. Равен 1, если ошибка (SP-PV) превышает значение, заданное в таблице программного задатчика.
- **Остановка профиля программного задатчика.** Равен 1, если профиль активен, но в данный момент приостановлен.

Номер текущего шага хранится в верхних 8 битах слова addr+33 параметров программного сзадатчика. 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 DirectSOFT32, который автоматически выполняет проверку диапазонов V-памяти.



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 DirectSOFT32, так как в нем профиль сигнала выводится на экран. Не забудьте выбрать достаточно медленную временную развертку, чтобы полностью вывести пары участков сигнала наклон/выдержка в окне формы сигналов.

Советы по поиску неисправностей

В. Контур не переходит в автоматический режим.

О. 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.

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

О. 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.

В. Управляющий выход не равен нулю, но он неправильный.

О. 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 DirectSOFT32 выводит значения SP, PV, смещения и управляющего выхода в десятичном (BCD) формате, преобразуя их в бинарный формат перед обновлением таблицы контура.

В. Программный задатчик не работает после установки бита запуска.

О. 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 переходит к фиксированному участку, при этом создается иллюзия, что первое линейное изменение не работает.

Контур находится в каскадном режиме и пытается получить удаленное 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 начале вашей программы, затем переводя ПЛК в рабочий режим и вручную запуская программный задатчик.

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

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

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

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

В. Кажется, что уставка контура изменяется сама по себе.

О. 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.

В. Значения SP и PV, вводимые из DirectSOFT32, работают правильно, но эти же значения, записываемые программно, работают неправильно.

О. Окно Просмотр ПИД-регулятора (PID View) в DirectSOFT32 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 DirectSOFT32 выполняет необходимые преобразования. При использовании 12-битового униполярного формата значения в таблице лежат в диапазоне от 0 до FFF.

В. Контур кажется неустойчивым и не поддающимся настройке независимо от того, какие коэффициенты усиления используются.

О. 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 велико запаздывание. Это означает, что PV медленно реагирует на изменения управляющего выхода. Причиной этого может быть слишком большое «расстояние» между исполнительным механизмом и датчиком PV, или недостаточна пропускная способность регулирующего клапана.

Может существовать возмущение процесса, которое контур не может преодолеть. Убедитесь, что значение PV устойчиво при постоянном SP.

Словарь терминов ПИД-регулирования

Автоматический режим - Режим работы контура, в котором контур выполняет расчет ПИД-алгоритма и изменяет управляющий выход.

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.

Составляющая смещения(Bias Term) - В позиционной форме уравнения ПИД-регулятора это сумма интегрального члена и начального значения выхода.

Безударный переход - Метод изменения режима работы контура, позволяющий избежать резкого изменения уровня управляющего выхода за счет искусственного приравнивания в момент изменения режима значений уставки (SP) и переменной (PV), или составляющей смещения и управляющего выхода.

Каскадные контура - Каскадный контур получает значение уставки с выхода другого контура. Каскадные контуры связаны соотношением главный/подчиненный и работают совместно, управляя одной переменной - PV.

Каскадный режим - Режим работы контура, в котором контур получает значение задания - SP с выхода другого контура.

Непрерывное управление - Управление процессом, использующее в качестве управляющего выхода непрерывный (аналоговый) сигнал.

Контур прямого действия - Контур, в котором PV возрастает с ростом управляющего выхода. Другими словами, коэффициент усиления процесса является положительным.

Рассогласование (Error) - Разность значений SP и PV, рассогласование =SP-PV

Зона нечувствительности рассогласование - Необязательная возможность, которая делает контур нечувствительным к небольшим отклонениям. Размер зоны нечувствительности задается пользователем.

Квадрат отклонения - Необязательная функция, которая умножает отклонение само на себя, но сохраняет первоначальный знак. Она уменьшает влияние малых отклонений, увеличивая влияние больших.

Предварение, управление по возмущению (Feedforward) - Метод оптимизации управляющей реакции контура, когда изменение уставки или возмущение, известно и имеет поддающееся вычислению влияние на смещение.

Управляющий выход - Численный результат решения уравнения ПИД-регулятора, выдаваемый контуром с целью достижения нулевого текущего рассогласования.

Дифференциальный коэффициент усиления (Derivative Gain) - Константа, определяющая величину дифференциальной составляющей ПИД-регулятора, соответствующую текущему рассогласованию.

Интегральный коэффициент усиления (Integral Gain) - Константа, определяющая величину интегральной составляющей ПИД-регулятора, соответствующую текущему рассогласованию.

Главный контур - В каскадном управлении это контур, генерирующий уставку - SP для каскадного контура.

Ручной режим - Режим работы контура, в котором расчет ПИД-регулятора прекращается. Оператор вручную управляет контуром, напрямую записывая значение управляющего выхода.

Подчиненный контур - В каскадном управлении это контур, получающий значение уставки - SP от главного контура.

Дискретное управление (On/Off Control) - Простой метод управления процессом с помощью включения/выключения подачи энергии в систему. Масса процесса усредняет влияние включения/выключения на переменную PV. Непрерывный выход DL250 преобразуется в управление включением/выключением с помощью небольшой программы

Контур ПИД-регулирования - Математический метод управления замкнутым контуром, включающий сумму трех составляющих, использующих значения пропорционального, интегрального и дифференциального отклонений. Коэффициенты усиления трех составляющих постоянны и независимы, позволяя оптимизировать (настраивать) контур для конкретной физической системы.
Позиционный алгоритм - Управляющий выход рассчитывается в соответствии с расположением (позицией) PV относительно SP
Процесс - Производственная процедура, увеличивающая стоимость сырья. В частности управление процессом связано с внесением в ходе процесса в сырье химических изменений.
Переменная процесса (PV) - Количественная мера физического свойства материала, которая влияет на качество окончательного продукта и требует контроля и управления.
Пропорциональный коэффициент усиления (Proportional Gain) - Константа, определяющая величину пропорциональной составляющей ПИД-регулятора, соответствующую текущему рассогласованию.
Аварийный сигнал абсолютного значения PV - Программируемый аварийный сигнал, сравнивающий значение PV с пороговыми значениями аварийного сигнала.
Аварийный сигнал отклонения PV - Программируемый аварийный сигнал, сравнивающий разность значений SP и PV с пороговым значением рассогласования.
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 последовательностей уставок -SP.
Скорость(Rate) - Также называемая дифференциатором, составляющая скорости реагирует на скорость изменения рассогласования.
Удаленная уставка (Remote Setpoint) - Ячейка, из которой контур считывает свою уставку, если он является подчиненным контуром в каскадной топологии контуров.
Интегратор(Reset) - Интегральная составляющая добавляет каждую вычисленную ошибку к предыдущей, поддерживая вычисление текущей суммы, называемой смещением.
Взвинчивание (выбег) интегральной составляющей (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.
Контур обратного действия - Контур, в котором PV увеличивается в ответ на уменьшение управляющего выхода. Другими словами, коэффициент усиления процесса отрицателен.
Период опроса (Sampling Time) - Интервал между вычислениями ПИД-регулятора. Метод, с помощью которого процессор управляет процессом, называется квазианалоговым управлением, так как расчеты выполняется периодически.
Уставка (SP) - Требуемое значение переменной процесса. Уставка (SP) — это сигнал, подаваемый на вход контроллера контура при работе замкнутого контура.
Отклонение PV на участке выдержки - Отклонение PV на участке выдержки — это мера разности между SP и PV на участке выдержки профиля программного задатчика.
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)
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.
Скоростной ПИД - алгоритм - Управляющий выход рассчитывается так, чтобы представлять изменение (скорость) PV, чтобы стать равной SP.

Глава 9

9. Обслуживание и поиск неисправностей

В данной главе…

Обслуживание технических средств

Нормальное обслуживание

DL06 это контроллер не требующий регулярного обслуживания; только нескольких периодических проверок раз в один-два месяца, для проверки состояния системы по следующим аспектам:

- Температура воздуха в шкафу должна быть в пределах рабочего диапазона температур.
- Воздушные фильтры. Если в шкафу установлены фильтры, очистите их, а при необходимости замените
- Плавкие предохранители и автоматы — проверьте сохранность.
- Вентилляционные отверстия в DL06. Проверьте и очистите. Если корпус нуждается в чистке, обесточьте контроллер и протрите влажной тряпкой, не допуская попадания воды внутрь корпуса. Не используйте моющие средства.

Диагностика

Диагностика

Система на базе DL06 выполняет несколько программ диагностики при каждом сканировании Процессора. Диагностика разработана таким образом, чтобы обнаруживать отказы и сбои различного вида в ПЛК. Выделяют два основных вида ошибок: *критические и некритические*

Критические ошибки

Критические ошибки (Fatal Error) — это ошибки, которые связаны с риском того, что система не будет правильно или безопасно функционировать. Если Процессор находится в Рабочем Режиме, когда 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 Процессор не позволит войти в Рабочий Режим до тех пор, пока ошибка не будет исправлена.
Несколько примеров Критических ошибок:

- Отказ источника питания
- Ошибка четности или сбой Процессора.
- Некоторые программные ошибки.

Некритические ошибки

Некритические ошибки Non-fatal Error) — это такие ошибки, которые требуют внимания, но не приведут к некорректным действиям. Они не вызвают переход из Рабочего режима в Программный режим, также как не препятствуют переходу Процессора в Рабочий режим. Есть специальные реле, которые позволяют прикладной программе обнаружить появление некритической ошибки. Прикладная программа может затем правильно отключить систему или при необходимости переключить Процессор в Программный режим.
Примеры некритических ошибок:
• Определенные программные ошибки - программируемые устройства известят Вас относительно ошибки, если она произойдёт в оперативном режиме.
• DirectSOFT32 предоставляет код ошибки и сообщение об ошибке.
• Ручной программатор показывает код ошибки и короткое описание ошибки.
В приложении B данного руководства приведен полный перечень сообщений об ошибках, упорядоченных по их коду. Во многих из этих сообщений имеются указания на дополнительные ячейки V-памяти, в которых находится дополнительная информация об ошибках. Такой памятью является V-память и специальные реле (SP) (обратитеськ Приложению D).

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 диагностики
Определяемый пользователем	Код ошибки, используемый с командой FAULT	V7751
Системные ошибки	Код критической ошибки	V7755
	Код существенной ошибки	V7756
	Код менее существенной ошибки	V7757
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 сканирования, мс	V7775
	Минимальное время сканирования, мс	V7776
	Максимальное время 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.

Реле состояния Процессора			SP51	Сторожевой таймер цикла
SP11	Принудительный рабочий режим		SP52	Синтаксическая ошибка
SP12	Терминальный рабочий режим		SP53	Невозможно разрешить логику
SP13	Тестовый рабочий режим		SP54	Ошибка связи
SP15	Тестовый режим STOP		SP56	Переполнение в табличной команде
SP16	Терминальный программный режим		**Реле состояния аккумулятора**	
SP17	Принудительный STOP		SP60	Акк. меньше заданного значения
SP20	Команда STOP выполнена		SP61	Акк. равен заданному значению
SP22	Прерывание разрешено		SP62	Акк. больше заданного значения
Реле контроля системы			SP63	Результат в Акк. равен нулю
SP36	Использована подмена битов		SP64	Произошел заем части
SP37	Ошибка времени цикла		SP65	Произошло заимствование
SP40	Критическая ошибка		SP66	Произошел перенос части
SP41	Некритическая ошибка		SP67	Произошел перенос
SP42	Ошибка диагностики		SP70	Результат отрицательный (по знаку)
SP44	Ошибка программной памяти		SP71	Ошибка ссылки на указатель
SP45	Ошибка Ввода/Вывода		SP73	Переполнение
SP46	Ошибка связи		SP75	Данные не в формате BCD
SP50	Команда FAULT выполнена		SP76	Загрузка нуля

Коды системных ошибок DL06

Эти ошибки могут формироваться Процессором или Ручным Программатором в зависимости от фактической ошибки. В приложении В 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)
E104	Сбой при записи
E151	Неправильная команда
E311	Ошибка связи 1
E312	Ошибка связи 2
E313	Ошибка связи 3
E316	Ошибка связи 6
E320	Тайм-аут
E321	Ошибка связи
E360	Тайм-аут внешнего порта ручного программатора
E501	Неверный ввод
E502	Неверный адрес
E503	Неверный оператор
E504	Неверная ссылка/значение
E505	Неправильная команда
E506	Неправильная операция
E520	Неверная операция - ЦПУ в Рабочем режиме
E521	Неверная операция - ЦПУ в режиме работы Теста
E523	Неверная операция - ЦПУ в режиме Тест Программы
E524	Неверная операция - ЦПУ в режиме Программирования

Код ошибки	Описание
E525	Переключатель Режимов 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
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		Код ошибки	Описание
E4**	Нет программ в Процессоре		E438	Неправильный адрес IRT
E401	Отсутствие оператора END		E440	Неправильный адрес данных
E402	Отсутствие оператора LBL			
E403	Отсутствие оператора RET		E441	ACON/NCON
			E451	Неверное MLS/MLR
E404	Отсутствие оператора FOR		E453	Отсутствие Таймера/Счетчика
E405	Отсутствие оператора NEXT		E454	Неверный TMRA
			E455	Неверный CNT
E406	Отсутствие оператора IRT		E456	Неверный SR
E412	SBR/LBL > 64		E461	Переполнение стека
E421	Двойная ссылка на стадию		E462	Потеря значимости стека
E422	Двойная ссылка на SBR/ LBL		E463	Логическая ошибка
E423	Вложенные циклы		E464	Цепь LL не закончена
E431	Неправильный адрес ISG/SG		E471	Двойная ссылка на обмотку
E433	Неправильный адрес SBR		E472	Двойная ссылка на TMR
E434	Неправильный адрес RTC		E473	Двойная ссылка на CNT
E435	Неправильный адрес RT		E499	Неправильная запись текста в команде печати
E436	Неправильный адрес INT			
E437	Неправильный адрес IRTC			

Индикаторы состояния процессора

На передней панели микроконтроллера DL06 расположены индикаторы, с помощью которых вы можете диагностировать неисправности в системе. В нормальном рабочем режиме горят только индикаторы PWR (Питание) и RUN (Работа). В приведенной ниже таблице приводится краткий перечень возможных неисправностей.

Состояние индикатора	Возможная неисправность
PWR (Зеленый,выключен)	1. Неправильное напряжение в системе 2. Неисправен источник питания контроллера
RUN (Зеленый,выключен)	1. Ошибка в программе процессора 2. Процессор находится в режиме Программирования
ЦПУ (Красный,включен)	1. Действие электрических помех 2. Внутренняя неисправность Процессора
ЦПУ (Индикатор мигает красным цветом)	Напряжение у батареи низкое (смотрите стр. 4-8)

Индикатор PWR (Питание)

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

1. Питание блока неправильное или неприложено.
2. Источник питания блока поврежден.
3. Отключен источник питания другого компонента(ов).

Если напряжение источника питания ненадлежащее, то ПЛК не может нормально функционировать или вообще не сможет работать. Применяйте следующие указания при устранении неисправности.

ВНИМАНИЕ. 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. Проверьте все внешние предохранители и автоматы.
3. Проверьте внешние подключения на наличие контакта. Если вы используете отдельные клеммные блоки для подключения, убедитесь в их целостности.
4. 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.
5. Если все соединения правильны и питание на входе соответствует требуемым техническим условиям, то источник питания DL06 неисправен.

Лучший способ проверки Процессора это замена его другим исправным Процессором, чтобы убедиться в наличии неисправности. Съемные клеммники DL06 позволяют относительно легко это сделать. 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.

Индикатор RUN (Работа)

Если Процессор не может быть переведен 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).

Оба программирующих устройства, Ручной Программатор и DirectSOFT32, возвращают сообщение с описанием возникшей неисправности. Существуют AUX функции, зависящие от вида ошибки, которые помогут 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 (ЦПУ)

Если индикатор CPU включен, то в Процессоре 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 Процессор 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.

Если индикатор Процессора мигает, то Напряжение у батареи низкое (смотрите стр. 4-8).

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).
- Устройство, подсоединенное к порту, неправильно посылает данные или в устройстве дополнительно выполняется другое приложение.
- Существует большая разность потенциалов заземления двух соединенных устройств.
- Электрические помехи вызывают периодические ошибки.
- В Процессоре неисправный коммуникационный порт, Процессор должен быть заменен.

При появлении ошибки связи с *DirectSOFT32*, читайте руководство по *DirectSOFT32*. Там есть глава поиска неисправностей, содержание которой поможет в диагностировании проблем с настройками портов связи, адресами и др.

Поиск неисправностей в модулях ввода/вывода

Проверяемые причины

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

- Ошибка конфигурации Высокоскоростного Ввода/Вывода
- Перегоревший предохранитель в вашем механизме (DL06 не имеет внутренних предохранителей Ввода - вывода)
- Плохой контакт со съемным клеммником
- Отказ внешнего источника питания 24В постоянного тока
- Входные или выходные цепи ПЛК вышли из строя

Несколько быстрых шагов

При поиске неисправности в DL06 имеется несколько обстоятельств, которые вам следует учитывать. Они помогут быстро найти и исправить неисправности в подсистеме Ввода/Вывода.

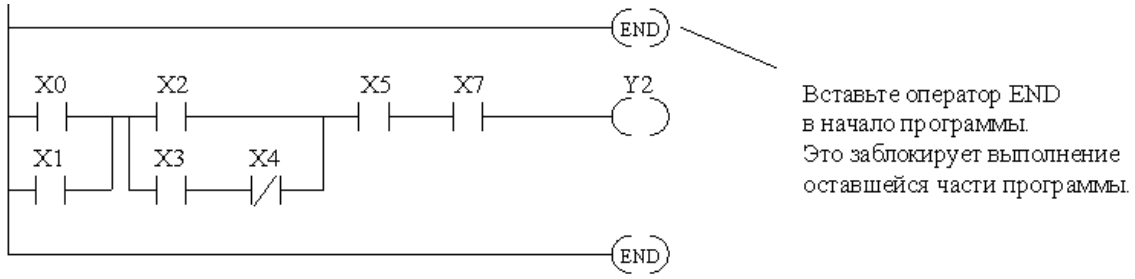
- Ошибки конфигурации высокоскоростного ввода/вывода чаще проявляются при написании программы. Если неисправны входы/выходы X0-X2 и Y0-Y1, то проверьте параметры перечисленные в главе 3, используемые при настройке высокоскоростного ввода/вывода.
- В выходных модулях нельзя автоматически обнаружить выходные точки с коротким замыканием или с обрывом. Если вы полагаете, что одна или несколько точек выходного модуля неисправны, то измерьте перепад напряжений между общим проводом и предполагаемой точкой. Напоминаем, что при использовании Цифрового Вольтметра необходимо учитывать ток утечки из выходного устройства, такого как семистор или транзистор. Отключенная точка может дать такой же ток, как включенная точка при отключенной нагрузке.
- Индикаторы состояния точек Ввода/Вывода в модулях являются индикаторами на логической стороне. Это значит, что светодиод, указывающий состояние «включено» или «выключено», отражает это состояние по отношению к Процессору. Индикатор состояния на выходном модуле может нормально функционировать, в то время как выходное устройство (например, семистор 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 сопротивление параллельно входу или выходу цепи. Величина этого сопротивления будет зависеть от тока утечки и используемого напряжения, но обычно сопротивление 10-20 Ком будет работать. Убедитесь, что резистор для вашего приложения правильно подобран.
- Самый легкий способ определения неисправного модуля состоит в замене его, если вы имеете резерв. Однако если другое устройство вызвало сбой в этом модуле, то оно может привести к такому же сбою 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.

В Процессорах серий DL06 выходные точки могут находиться в состоянии «включено» или «выключено». Если вы хотите провести независимую от прикладной программы проверку Ввода/Вывода используйте следующую процедуру:

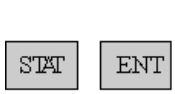
Шаг	Действие
1	Используйте ручной программатор или DirectSOFT32 для связи с ПЛК.
2	Измените режим на Программный.
3	Перейдите в адрес 0.
4	Поставьте в адресе 0 оператор END (При этом программа будет выполняться только до адреса 0 и не сможет включать или выключать точки Ввода/Вывода).
5	Измените режим на Рабочий.
6	Используйте ручной программатор 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 форсирование Вводов/Выводов 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.

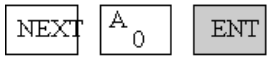
Работа с клавишами ручного программатора при тестировании выходной точки

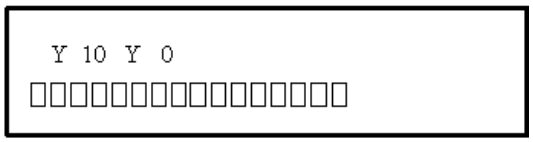


При пустом экране дисплея нажмите следующие клавиши

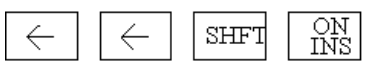


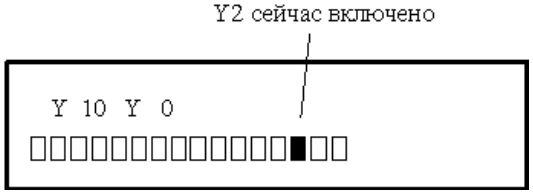
Используйте клавиши PREV или NEXT для выбора типа данных Y





Используйте клавиши со стрелками для выбора тестируемой точки, затем используйте ON и OFF для изменения состояния





Поиск и устранение помех

Проблемы электрических помех

Помехи являются одной из самых трудных проблем при диагностике. Электрические помехи могут вводиться в систему разными способами, их можно разделить на две категории, введенные и наведенные. Иногда трудно определить, как помехи вводятся 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 «земля» недоступна, подведите заземляющий провод 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 вводиться в систему через источник питания контроллера или модули Ввода/Вывода. Установка развязывающего трансформатора для всех источников переменного тока может решить эту проблему. Источниками постоянного тока должны быть хорошо заземленные источники питания высокого качества. Импульсные источники питания постоянного тока обычно генерируют больше помех, чем линейные 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

Процессоры DL06 предоставляют несколько возможностей для отладки Вашей программы до и после запуска производственных машин. В данном разделе рассматриваются следующие полезные для Вас темы.

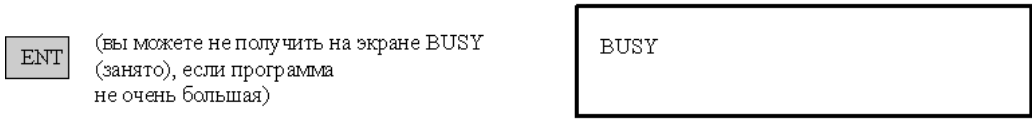
- Проверка синтаксиса программ
- Проверка дублированных ссылок
- Специальные команды
- Редактирование в рабочем режиме (RUN TIME EDIT)
- Форсирование точек ввода/вывода

Проверка синтаксиса

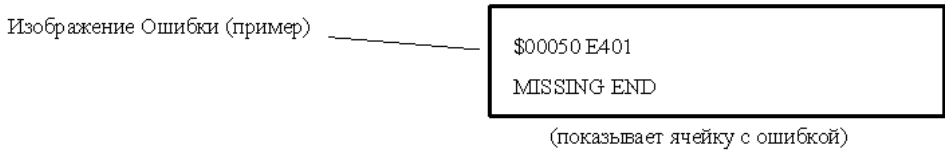
Хотя Ручной Программатор и DirectSOFT32 обеспечивают проверку ошибок при вводе программы, вы можете отдельно проверять измененные программы. Оба этих устройства для программирования предусматривают возможность проверки синтаксиса программ. Например, вы можете использовать вспомогательную функцию AUX 21, CHECK PROGRAMM, для проверки синтаксиса программы с Ручного Программатора, либо вы можете воспользоваться опцией меню Diagnostics (Диагностика) ПЛК в DirectSOFT32. Эта проверка позволяет найти разнообразные программные ошибки. На следующем примере показывается, как применять проверку синтаксиса с Ручного Программатора.

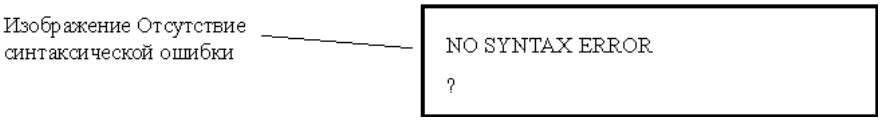
Применить AUX 21 для проверки синтаксиса

Выбрать проверку синтаксиса (выбор по умолчанию)



Появится одна из двух картинок на экране





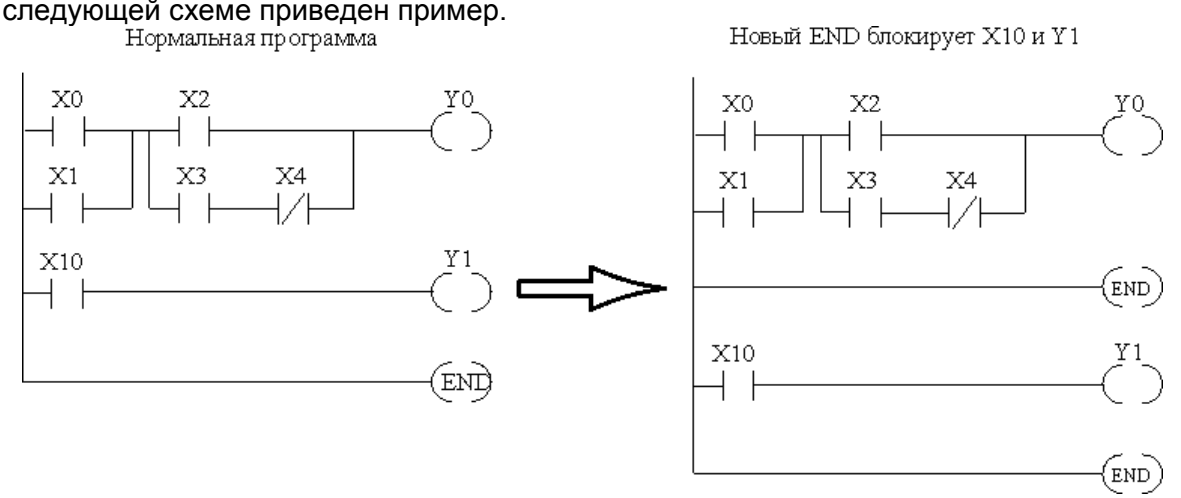
Обратитесь к разделу Кодов Ошибок с полным списком кодов программных ошибок. Если вы получили ошибку, нажмите CLR, и на Ручном Программаторе отобразится команда с ошибкой. Исправьте ошибку и продолжайте Проверку Синтаксиса, пока не появится сообщение NO SYNTAX ERROR (синтаксических ошибок нет).

Специальные команды

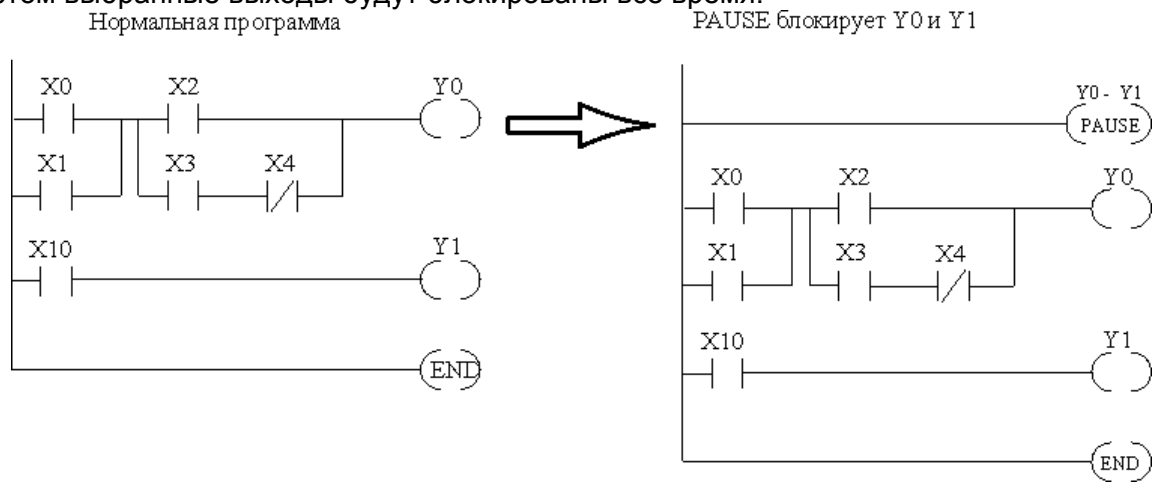
Существует несколько команд, которые могут помочь вам при отладке программы во время операций после запуска машин.

- END
- PAUSE
- STOP

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

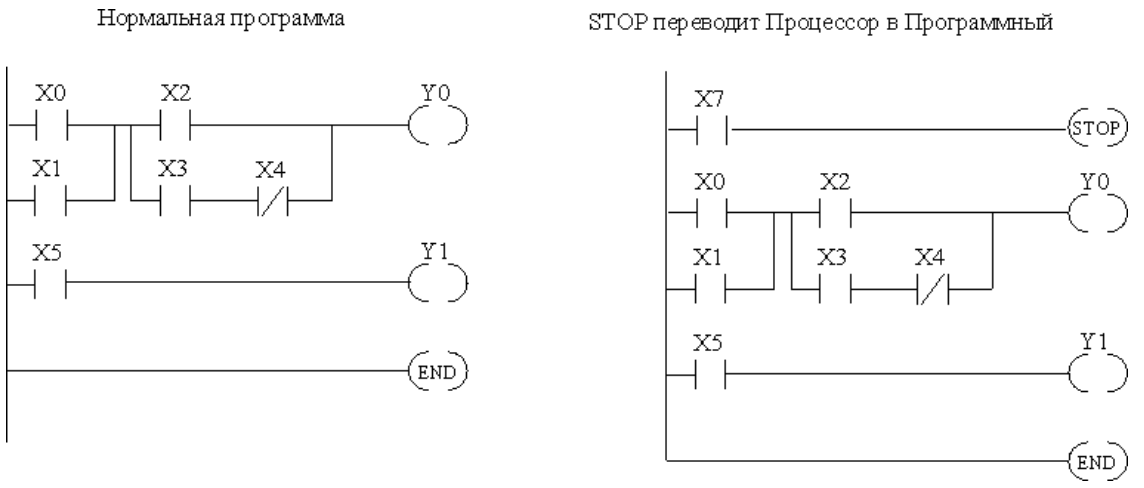


Команда PAUSE. Эта команда позволяет быстро установить режим, в котором входы (или другие логические переменные) функционируют при блокировании выбранных выходов. При этом регистр отображения выходов обновляется, но выходные состояния не записываются в модули. Например, вы можете сделать эту блокировку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



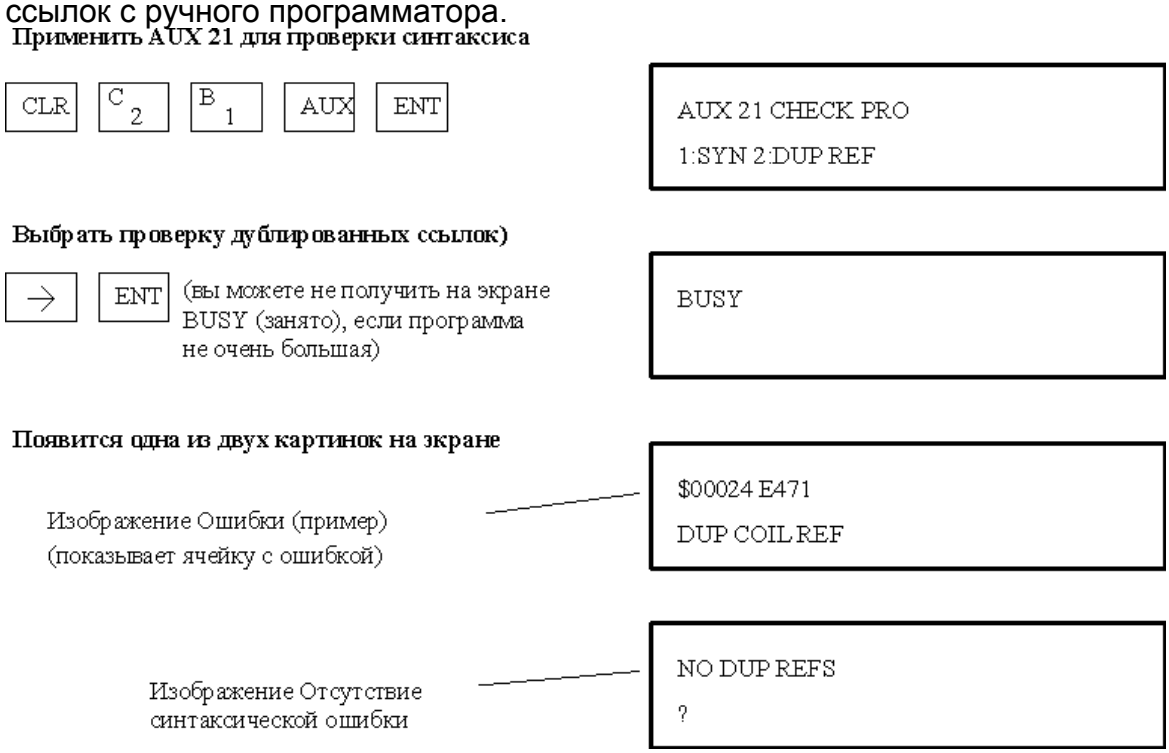
Команда STOP. Иногда после запуска машин вам необходимо быстро отключить все выходы и вернуться в Программный Режим. Для этого Вы можете использовать команду STOP. При выполнении этой команды Процессор автоматически выйдет из Рабочего Режима и перейдет в Программный Режим. Напоминаем, что все выходы в Программном режиме отключаются. На следующей схеме приведен пример условия, при котором Процессор возвращается в Программный Режим.

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



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 Diagnostics (Диагностика) ПЛК в DirectSOFT32. Эта проверка позволяет найти разнообразные программные ошибки. 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.

Редактирование в рабочем режиме

Процессоры DL06 дают вам возможность делать изменения в прикладной программе в Рабочем Режиме. Такое редактирование не является «безударным». Сканирование Процессора мгновенно прерывается (а выходы сохраняются в текущем состоянии), пока изменение в программе не будет завершено. Это означает, что, если выходы отключены, то они останутся отключенными до тех пор, пока не закончится изменение программы. Если выходы включены, они останутся включенными.

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:

1. Если новая команда содержит синтаксическую ошибку, то Процессор не перейдет в Рабочий Режим.
2. 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.
3. 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.

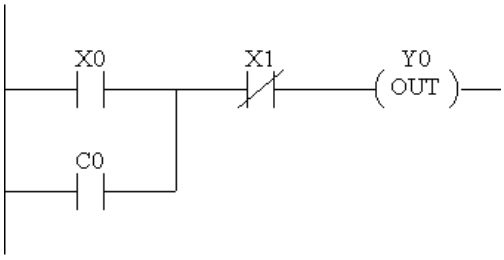
Не все команды могут редактироваться в сеансе Редактирования в Рабочем Режиме. В следующем списке приведены команды, которые могут редактироваться.

Обозначение	Описание
TMR	Таймер
TMRF	Быстрый таймер
TMRA	Аккумулирующий таймер
TMRAF	Аккумулирующий быстрый таймер
CNT	Счетчик
UDC	Счетчик Вверх / Вниз
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-равно
ORE, ORNE	ИЛИ, если равно; ИЛИ, если НЕ-равно
STR, STRN	Сохранить, если больше или равно; Сохранить, если меньше
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 (константы)
ADDD	Сложение Двойное (констант)
SUBD	Вычитание Двойное (констант)
MUL	Умножение (констант)
DIV	Деление (констант)
CMPD	Сравнить Аккумулятор (констант)
ANDD	И Аккумулятора (констант)
ORD	ИЛИ Аккумулятора (констант)
XORD	Исключающее ИЛИ Аккумулятора (констант)
LDF	Загрузка дискретных точек в Аккумулятора
OUTF	Вывод из Аккумулятора дискретных точек
SHFR	Сдвиг Аккумулятора вправо
SHFL	Сдвиг Аккумулятора влево
NCON	Числовая константа

Пример редактирования в рабочем режиме

Мы будем использовать показанную на рисунке программную логику для описания того, как происходит процесс редактирования в рабочем режиме. В этом примере мы будем изменяить X0 на C10. Предполагается, что Процессор 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 Процессор DL06 обрабатывает запросы при таком форсировании.

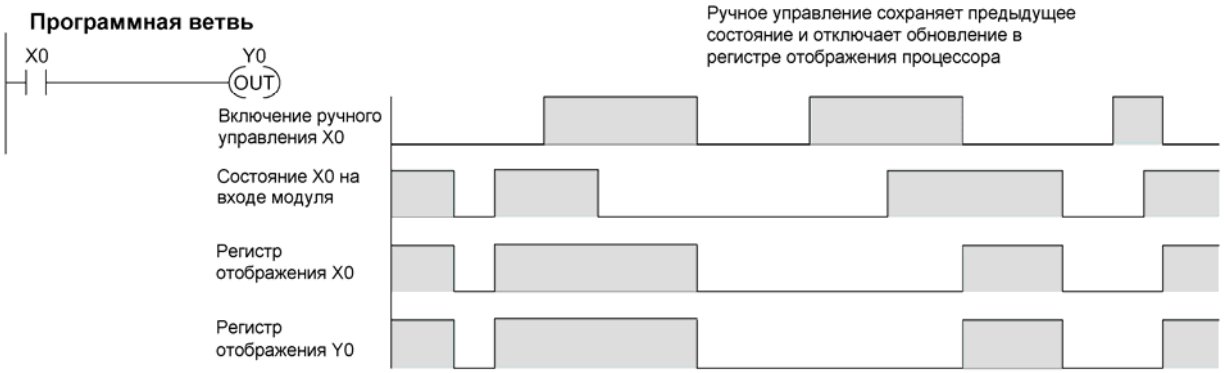
ПРЕДУПРЕЖДЕНИЕ. Только квалифицированные лица, знающие в полном объеме все аспекты приложения, могут изменять программу. Тщательно проанализируйте последствия любых изменений, чтобы минимизировать риск нанесения вреда персоналу или повреждения оборудования.

Существует два типа форсирования доступного для процессора DL06. (Глава 3 дает детальное описание того, как процессор обрабатывает каждый тип запроса форсирования).

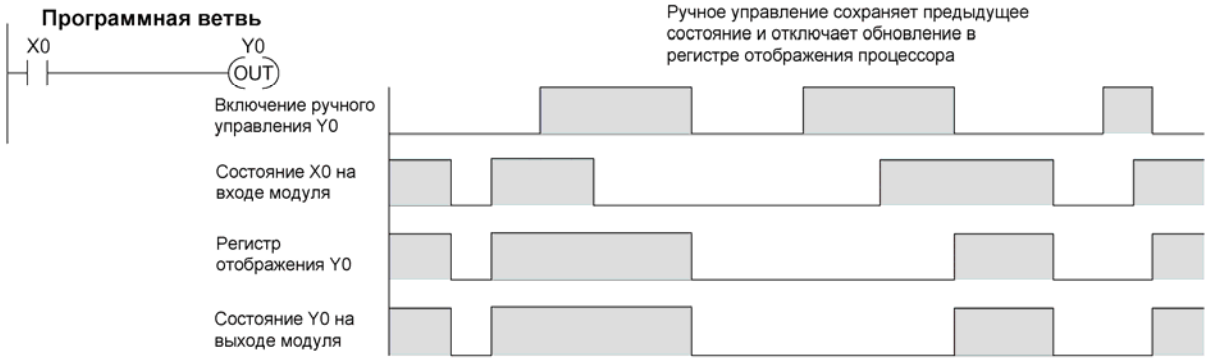
- **Нормальное форсирования:** Этот тип форсирования может временно изменять состояние бита дискретного сигнала. Например, вы желаете включить вход, хотя реально его нет. Это даст вам возможность изменить состояние точки, которое хранится в регистре отображения. Это значение будет действовать до тех пор, пока ячейка регистра отображения не будет обновлена при следующем сканировании. Это полезно, главным образом, при тестировании, когда вам необходимо включить какой-то бит, чтобы запустить другое событие.
- **Подмена бита (Bit Override):** Подмена Бита может быть разрешена для конкретных битов, командой AUX 59 на непосредственно подключенном к контроллеру Ручном Программаторе или через пункт меню Bit Override в DirectSOFT32. Вы можете использовать подмену бита 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».

Имеется преимущество доступное при использовании подмены бита. 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 обслужит 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.

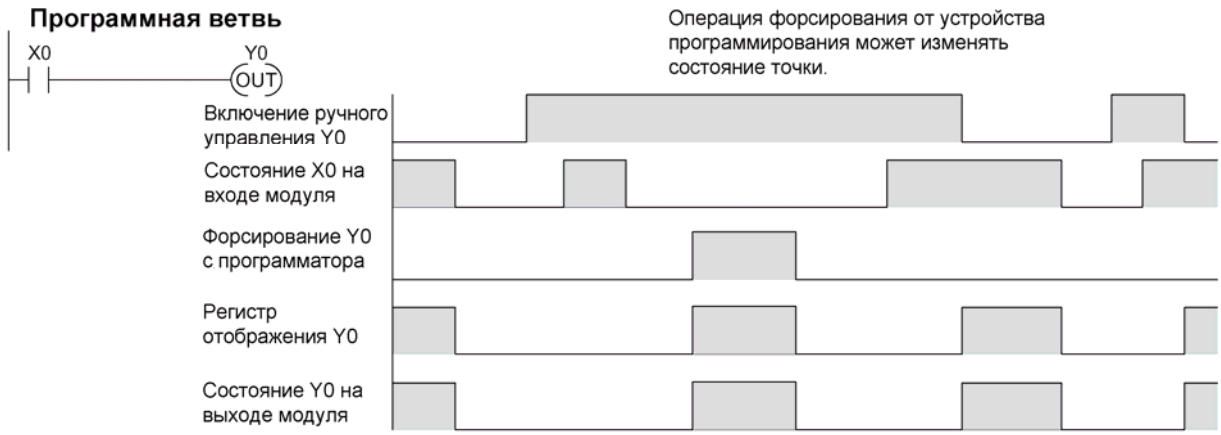
Следующие рисунки показывают, как работает подмена бита для точек ввода/вывода. Пример использует простой цепь релейной программы, но данная идея одинакова для любого битового типа памяти.



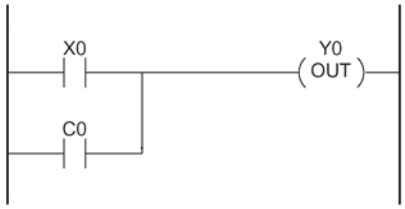
Следующая схема показывает, как работает подмена бита для точки вывода. Заметьте, что ручное управление битом поддерживает выход в текущем состоянии. Если выход включен, когда включается подмена бита, то выход остается включенным. Если он выключен, то выход остается выключенным.



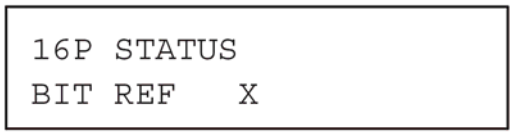
Следующая схема показывает, как Вы можете использовать устройство программирования в комбинации с подменой бита, чтобы изменить состояние бита. Помните, подмена бита только отключает изменения от процессора. Вы можете использовать устройство программирования, чтобы форсированием изменить состояние бита. 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.



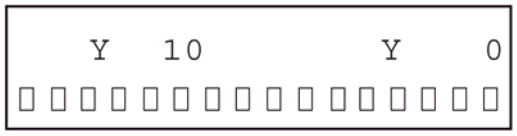
Следующие диаграммы показывают краткий пример того, как Вы могли бы использовать ручной программатор, чтобы форсировать точку ввода/вывода. Помните, если Вы используете операцию подмены бита, то процессор сохранит форсированное значение, пока Вы не отключите подмену бита или пока Вы не удалите форсирование. Регистр отображения не будет обновляться в соответствии с состоянием входа модуля. Значение от прикладной программы, также, не будет использоваться, чтобы изменить регистр отображения выхода. Пример предполагает, что Вы перевели процессор в рабочий режим.



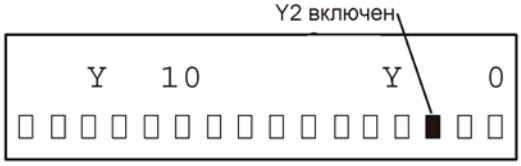
Очистите дисплей и введите следующую последовательность клавиш.



Используйте клавиши PREV или NEXT для выбора типа данных Y. (Как только увидите Y, нажмите 0 для перехода к выбору бита Y0.)

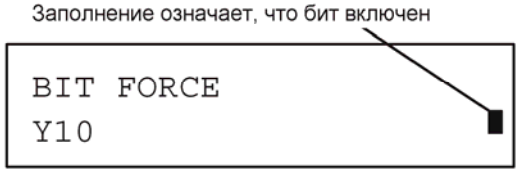


Используйте клавиши со стрелками, чтобы выбрать необходимую точку, затем используйте ON и OFF для изменения состояния.

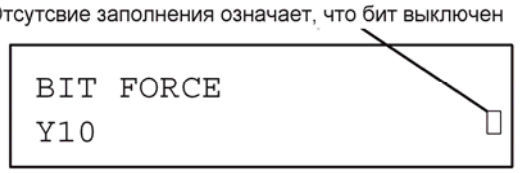


Нормальное форсирование с прямым доступом

Очистите дисплей и введите следующую последовательность клавиш для форсирования включения Y10. Заполнение, указывает, что бит включен.

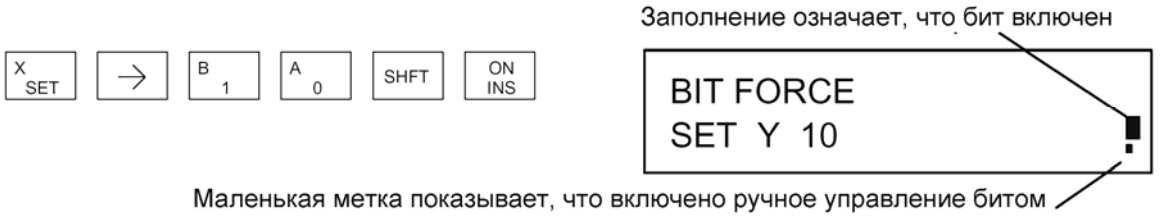


Очистите дисплей и введите следующую последовательность клавиш для форсирования выключения Y10. Отсутствие заполнения, указывает, что бит выключен.



Форсирование бита с подменой

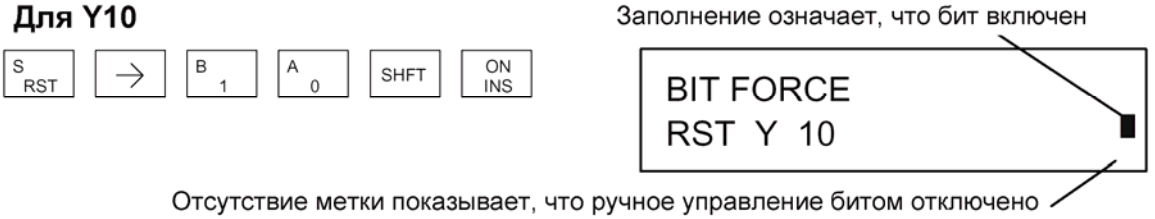
Очистите дисплей и введите следующую последовательность клавиш для форсирования состояния включения Y10.



Обратите внимание, в этой точке Вы можете использовать клавиши PREV и NEXT, чтобы перейти в смежные ячейки памяти и использовать клавишу 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

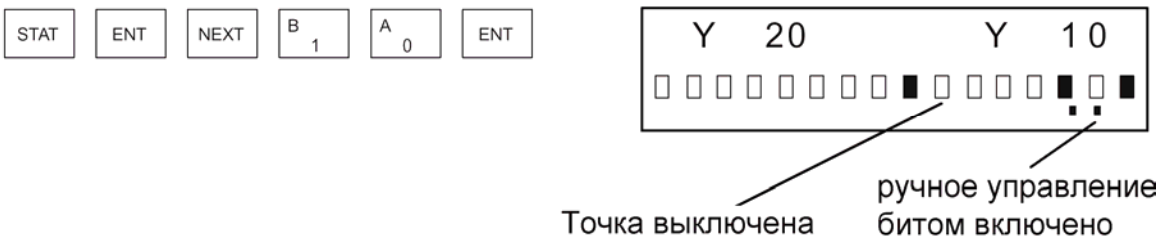


Подобно примеру выше, Вы можете использовать клавиши PREV и NEXT, чтобы перейти в смежные ячейки памяти и использовать клавишу SHFT OFF для выключения режима подмены бита.

Индикаторы подмены бита

Индикаторы подмены бита (Bit Override) могут отображатся на дисплее состояния ручного программатора. Ниже дана последовательность клавиш, чтобы вызвать отображение состояния для выходов Y10 - Y20.

Очистите дисплей и введите следующую последовательность клавиш для отображения состояния выходов Y10 – Y20.



Глава 10

10. Жидкокристаллическая панель

В данной главе…

Введение

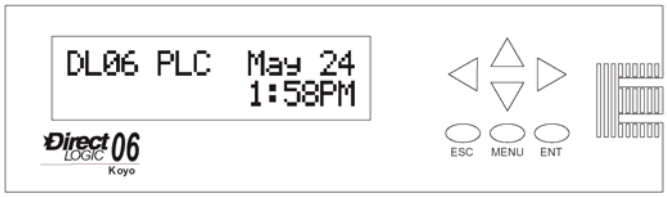
Жидкокристаллическая панель DL06 имеет дисплей на два ряда по 16 символов, и размещается непосредственно на лицевой панели микроконтроллера DL06. Жидкокристаллический экран с задней подсветкой легко читается практически при любом освещении.

Имеется несколько способов взаимодействия с LCD-панелью:

- Встроенная клавиатура
- Команды релейной логики для управления LCD-панелью
- Использование команд релейной логики для записи изменения бита состояния в указанной ячейке памяти.

Семь функциональных клавиш на передней стороне LCD-панели дают пользователю доступ к установке часов и календаря, значению данных ячеек V-памяти и состоянию ввода/вывода и др.. Индивидуальная авторизация по паролю позволяет:

- Изменение настроек часов/календаря или формата отображения.
- Просмотр и изменение значений V-памяти (включая и значения из двойного слова)

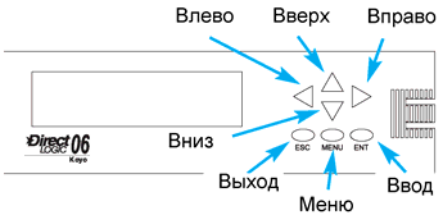


- Форсирование отдельных бит в состояние вкл./выкл. (до 16 бит на экран)
- Просмотр истории кодов ошибок
- Установка или изменение пароля
- Включение/выключение задней подсветки или зуммера

Потенциальные возможности использования LCD-панели для DL06 очень широки. Оператор может изменять значения для введения данных настроек или выбора времени работы механизма для производства различных изделий. Обслуживающий персонал может использовать интерфейс в шкафу управления, чтобы идентифицировать ошибки в работе механизма. Сообщения на LCD-панели могут быть запрограммированны для различных аварий или событий. LCD-панель может удовлетворять и многие другие потребности интерфейса оператора.

Клавиатура

LCD-панель имеет семь клавиш, которые можно использовать для навигации по иерархическому меню. Каждый показываемый экран имеет определенный набор активных клавиш, связанных с ним. Все другие клавиши (не связанные с текущим экраном) бездействуют.

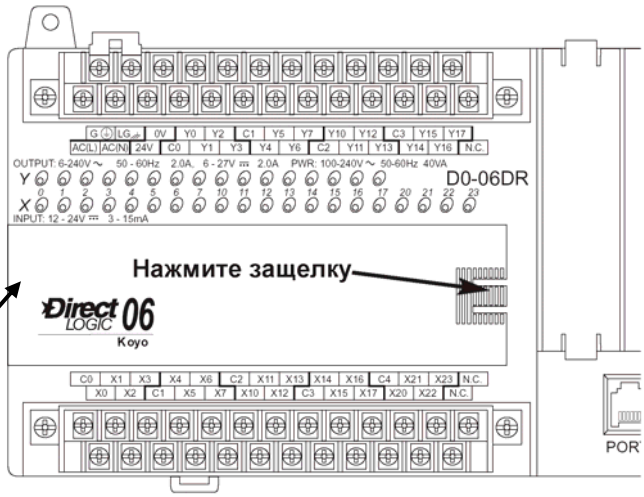


Функции клавиш		
Имя	**Надпись**	**Функция**
Стрелка вверх		Передвигает вверх или увеличивает значение
Стрелка вниз		Передвигает вниз или уменьшает значение
Стрелка влево		Передвигает в следующую цифру влево
Стрелка вправо		Передвигает в следующую цифру вправо
Выход	ESC	Возвращает к предыдущему экрану, или уровню в иерархии меню
Меню	MENU	Открывает список главного меню или выбирает подменю
Ввод	ENT	Входит внутрь пункта меню или сохраняет новое значение

Установка

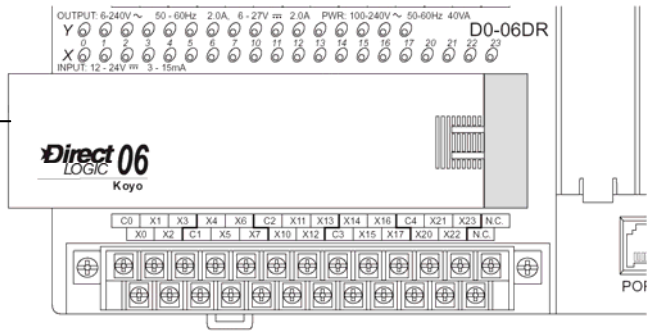
LCD-панель устанавливается в любую модель контроллера DL06 очень легко.

Удалите пластиковую крышку (расположенную между клеммами водов и выходов). Нажмите защелку, чтобы освободить ее, и сдвиньте крышку влево приблизительно на 1 см.



Пластиковая крышка

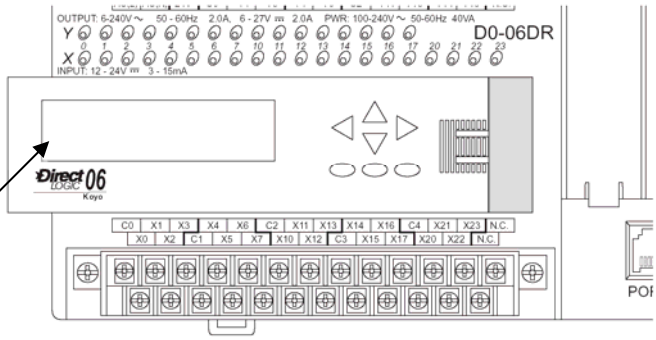
Теперь крышка должна подняться прямо из отверстия на лицевой панели DL06.



Сдвиньте и поднимите крышку

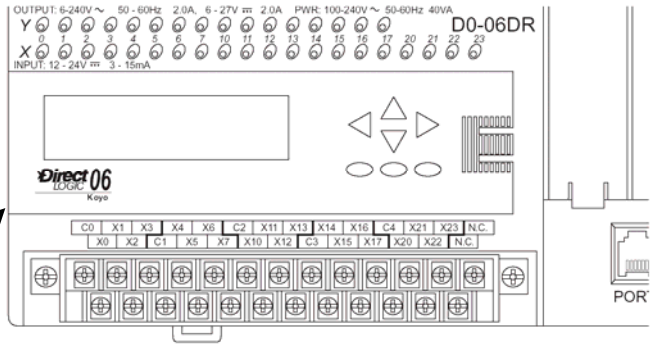
ПРЕДУПРЕЖДЕНИЕ: Перед установкой или удалением LCD-панели обязательно отключите питание контроллера.

Разместите LCD-панель сверху открытого отверстия, но сместив влево приблизительно на 1 см. Панель должна легко вставать в отверстие.



Разместите LCD-панель поверх открытого разъема

Сдвиньте панель вправо, так чтобы левая сторона панели совпадала с левой стороной контроллера. Фиксатор панели щелкнет при блокировке панели на месте установке.



Двигайте LCD-панель, пока не щелкнет фиксатор.

Приоритет дисплея

LCD-панель покажет одно из следующих сообщений (если питание не отключено от микроконтроллера):

- Экран по умолчанию (определяемый пользователем или заводской установкой по умолчанию)
- Выбор меню
- Сообщение из релейной программы
- Сообщение об ошибке

Встроенная вспомогательная клавиатура позволяет Вам осуществлять навигацию через эти экраны сообщений.

При включении питания обычно показывается сообщение по умолчанию. Сообщение по умолчанию установливается на заводе при изготовлении, но может быть изменено пользователем. Загрузка пользовательского сообщения по умолчанию описана ниже в этой главе.

Если происходит системная ошибка, то сообщение об ошибке прерывает показ сообщения по умолчанию (или другого текущего экрана), и отображает соответствующий код ошибки для диагностических целей.

Навигация по меню

Начиная с экрана по умолчанию, каждый раз при нажатии клавиши MENU будет показан список для перехода к следующему пункту меню. Клавишами стрелок вверх или вниз можно просматривать список меню (в направлении, обозначенном стрелкой), но *Вы должны первоначально нажать клавишу MENU (в сообщении по умолчанию), чтобы активизировать клавиши стрелок вверх и вниз.*

Существует семь встроенных пунктов меню. Некоторые из пунктов меню имеют подменю. Меню и подменю описаны в этой главе. Каждый выбор меню требует, чтобы Вы нажали клавишу ENT для просмотра или изменения установок или значений в пределах выбранного пункта главного меню.

Семь пунктов меню

Нажатие и удержание клавишу MENU, заставит панель пролистать следующие пункты меню:

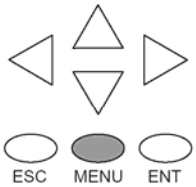
- M1 : Информация о ПЛК
- M2 : Конфигурация Системы
- M3 : Монитор
- M4 : Чтение/запись Календаря
- M5 : Чтение/запись Пароля
- M6 : Чтение истории ошибок
- M7 : Тест LCD-панели и ее настройки

В этой главе мы используем иллюстрации вспомогательной клавиатуры LCD-панели и область экрана, чтобы показать, как пройти через иерархию меню. Пример ниже показывает заводское сообщение по умолчанию, как Экран 1(Screen 1), и экран входа в главное меню, как Экран 2 (Screen 2).

Иллюстрация клавиатуры между экранами сообщений показывает, что нажатие клавиши MENU вызывает переход от Экрана 1 к Экрану 2. Этот тип представления используется повсюду в этой главе. Внутри иерархии меню, нажатие на клавишу ESC возвращает изображение предыдущего экрана.

Экран 1 – заводская установка

Нажатие на закрашенную клавишу приводит к переходу от экрана 1 к экрану 2

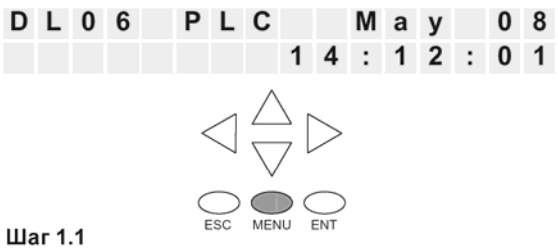


Экран 2

Проверка типа ПЛК, версии фирменного ПО, использования памяти, и т.д.

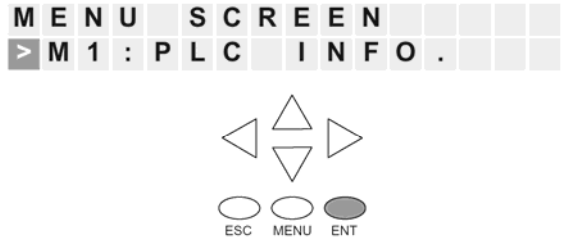
Menu 1, M1:PLC INFO.
В экране по умолчанию нажмите клавишу MENU один раз чтобы перейти к пункту меню 1 информация о ПЛК (PLC INFO)

Экран по умолчанию



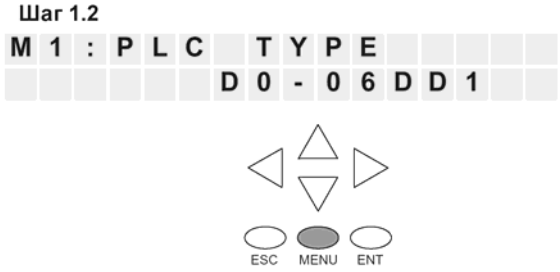
Нажмите ENT, чтобы войти в этот пункт меню. Первый экран внутри пункта меню «информация о ПЛК» - тип контроллера (M1:PLC TYPE). Этот пункт показывает номер модели контроллера.

Шаг 1.1

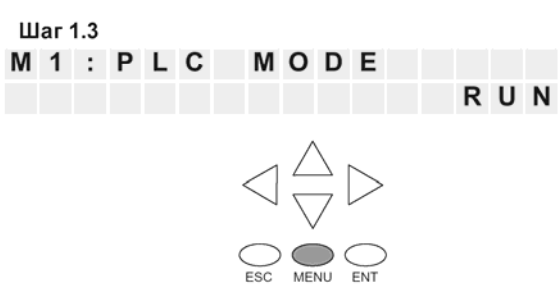


Нажмите снова клавишу MENU чтобы последовательно перейти к пункту режим ПЛК (PLC MODE). Режим ПЛК может быть любым из следующих: Рабочий (RUN), Останов (STOP) для режимов останова или программирования, Тестовый останов или работа (TEST-STOP) / (TEST-RUN). Вы можете включить DL06 в тестовый рабочий режим из режима тестового останова.

Шаг 1.2



Шаг 1.3

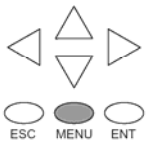


Примечание: примеры экранов меню, показанные в этой секции предполагают, что возможности ограничения доступа паролем не активирована. Если включена защита паролем, то пользователю необходимо будет вводить пароль при работе с панелью. Пользователи без пароля будут иметь доступ к ограниченному числу экранов.

Нажмите MENU еще раз для перехода к проверке версии фирменного ПО (FIRMWARE REV).

Шаг 1.4

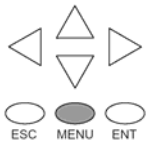
```
M1:FIRMWARE  REV.
      V1.000
```



Нажмите MENU еще раз для перехода к проверке использования памяти (LADDER MEMORY USED). Будет показано число используемых слов и общее число доступных слов памяти в ПЛК.

Шаг 1.5

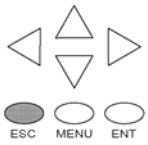
```
M1:LADDER MEMORY
USED   21 /  7680
```



Нажмите MENU снова для перехода к проверке режима работы с паролем (LADDER PASSWORD, ACTIVATED OR NOT ACTIVATED). Это последний экран в пункте меню PLC INFO.

Шаг 1.6

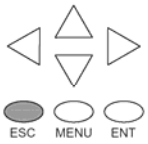
```
M1:LADDER PASSWD
   NOT ACTIVATED
```



Нажмите ESC для выхода из пункта M1 и возврата в главное меню.

Возврат к шагу 1.1

```
MENU SCREEN
>M1:PLC  INFO.
```



Нажмите ESC Еще раз для возврата в сообщение по умолчанию.

Экран по умолчанию

```
DL06  PLC   May 08
        14:22:11
```

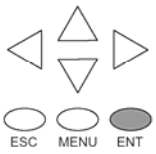
Проверка дополнительных слотов

Menu 2, M2:SYSTEM CFG

Из экрана по умолчанию дважды нажмите клавишу MENU для входа в пункт меню 2 «Конфигурация системы» (M2: SYSTEM CFG).

Шаг 2.1

```
>M1:PLC  INFO.
>M2:SYSTEM  CFG.
```



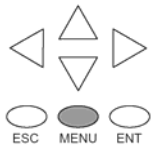
Нажмите ENT для входа в пункт меню «Конфигурация системы»

Шаг 2.2

```
M2:OPTION  SLOT  1
  D0-DEVNETS
```

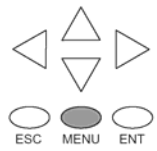
Примечание: Это только пример и он не представляет содержание дополнительных слотов вашего контроллера.

Нажмите MENU четыре раза и просмотрите в цикле содержимое всех четырех дополнительных слотов. Во второй строке показывается номер модели дополнительного модуля или признак, что разъем является пустым.



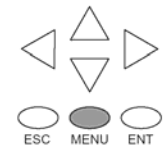
Шаг 2.3

```
M2:OPTION  SLOT  2
  EMPTY I/O SLOT
```



Шаг 2.4

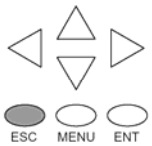
```
M2:OPTION  SLOT  3
  F0-04AD-1
```



Дважды нажмите ESC для возврата в сообщение по умолчанию.

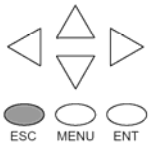
Шаг 2.5

```
M2:OPTION SLOT 4
  EMPTY I/O SLOT
```



Возврат к шагу 2.1

```
>M1:PLC INFO...
>M2:SYSTEM CFG..
```



Возврат к экрану по умолчанию

```
DL06 PLC May 08
      14:57:21
```

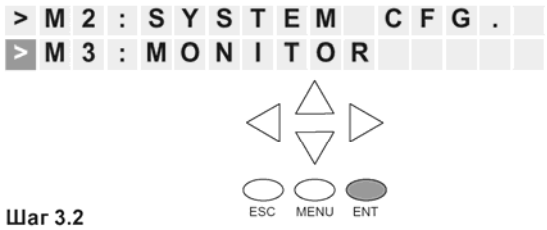
Просмотр и изменение значений данных

Menu 3, M3:MONITOR

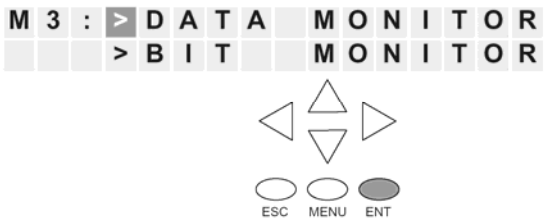
Из сообщения по умолчанию нажмите три раза клавишу MENU для выбора пункта меню Монитор (M3:MONITOR).

Подменю пункта M3:MONITOR содержат монитор данных и монитор битов. Монитор данных позволяет Вам просматривать содержание регистров памяти или указателей, и увидеть их содержание. Формат по умолчанию – BCD (двоичнокодированный-десятичный) / HEX(шестнадцатиричный), но формат числа может быть изменен на десятичный, установкой бита 8 ячейки V7742. Пожалуйста обратитесь к Карте Памяти DL06 за информацией о диапазонах памяти.

Шаг 3.1



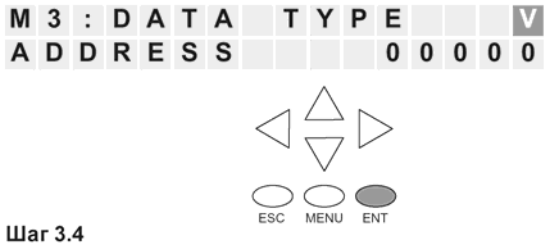
Шаг 3.2



Монитор данных

Тип данных (Data type) = V для ячеек V-памяти или P для указателей.
Нажмите MENU для изменения типа данных или нажмите ENT, чтобы определить тот регистр, который Вы хотите отобразить или изменить.

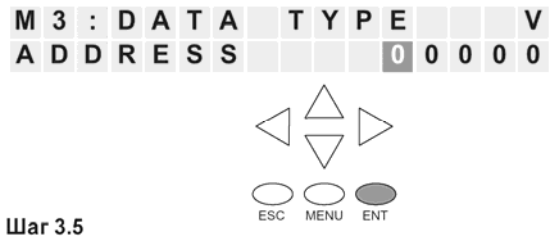
Шаг 3.3



Значения V-памяти

Используйте стрелки вправо или влево, чтобы переместить курсор в позицию, которую Вы хотите изменить. Используйте стрелки вверх или вниз, чтобы изменить цифру. Адрес V-памяти выражен восьмиричным числом, так что Вы не увидите цифр 8 и 9.
Этот экран позволяет Вам рассматривать две смежных ячеки V-памяти в формате BCD. Младшее слово - справа. Нажатие на клавишу ENT делает возможным изменить значение в младшем слове. На этом уровне иерархии меню, Вы можете также использовать стрелки вверх и вниз для перехода к другим ячейкам памяти.

Шаг 3.4



Шаг 3.5

Значения данных на этом экране будут представлены в виде четырех цифр в формате BCD/HEX, если бит 8 ячейки V7742 не установлен. Установка бита 8 ячейки V7742 изменяет формат данных на десятичный (пять цифр).

Используйте стрелки вправо или влево, чтобы переместить курсор в позицию, которую Вы хотите изменить. Используйте стрелки вверх или вниз, чтобы изменить цифру. Значение ячейки V-памяти выражено как BCD число, так что Вы будете видеть значения цифр в диапазоне: 0 - F. Формат данных может быть изменен на десятичный, установкой бита 8 ячейки V7742.

Шаг 3.6

```
M3:DATA   V    0
  CHG= 0000   0000
```

Шаг 3.7

```
M3:DATA   V    0
  CHG= AF06   0000
```

Шаг 3.8

```
M3:V    1  V    0
VAL  0000     AF06
```

Возврат к шагу 3.1

```
M3:DATA  TYPE    V
ADDRESS     0000 0
```

Нажмите ESC пять раз для возвращения в экран по умолчанию.

Возврат к экрану по умолчанию

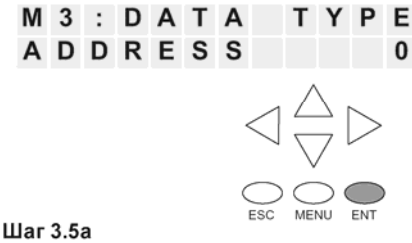
```
DL06 PLC   May 08
          15:02:13
```

Значения указателей

Нажмите ESC дважды для возврата на шаг 3.3, в котором курсор расположен на выборе параметра V, как показано на рисунке справа. Используйте стрелки вверх или вниз, чтобы изменить V на P. Теперь, будет показана информация об указателе (Pointer).

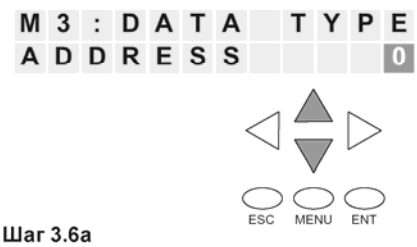
Возврат к шагу 3.3

Шаг 3.4a

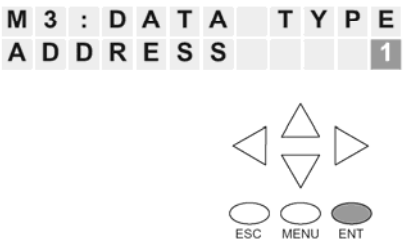


Используйте кнопки - стрелки вверх или вниз, чтобы изменить значение текущей цифры. Используйте стрелки влево или вправо, чтобы перейти от одной цифры к другой.

Шаг 3.5a



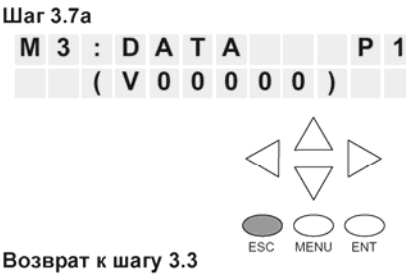
Шаг 3.6a



В Шаге 3. 7a стрелки вверх и вниз могут использоваться в цикле изменения адреса данных. Каждый раз при нажатиии на стрелку вверх или вниз, адрес увеличивается или уменьшается на одно 16-битное слово (адреса выражены в восьмиричном формате).

Чтобы перейти с монитора данных на монитор битов, нажмите ESC три раза, чтобы возвратиться на шаг 3.2 (пять раз, чтобы возвратиться экрану по умолчанию).

Шаг 3.7a



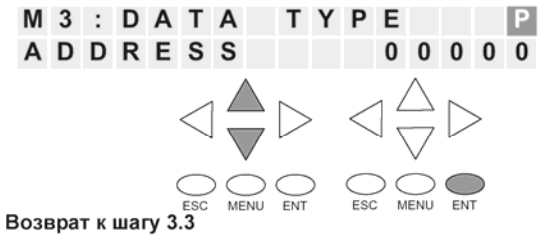
Возврат к шагу 3.3

Монитор битов

Состояние бита

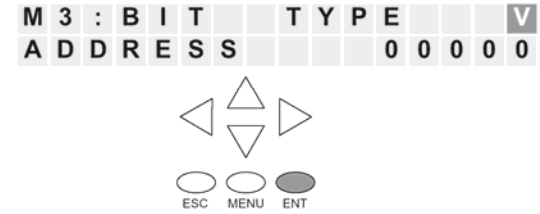
Из шага 3.3, нажмите стрелки вверх или вниз, затем ENT. Вы сможете увидеть один из одиннадцати битовых типов данных. Тип данных, который появляется на панели - это тип данных, к которому обращались в прошлый раз. Показанный адрес – это, также, адрес, к которому обращались в прошлый раз.

Возврат к шагу 3.3

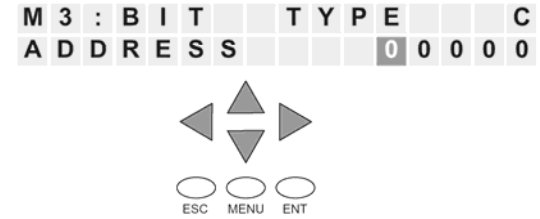


Нажмите ENT для изменения адреса.

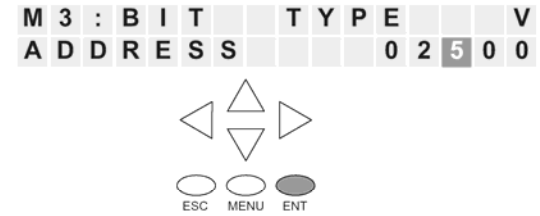
Возврат к шагу 3.3



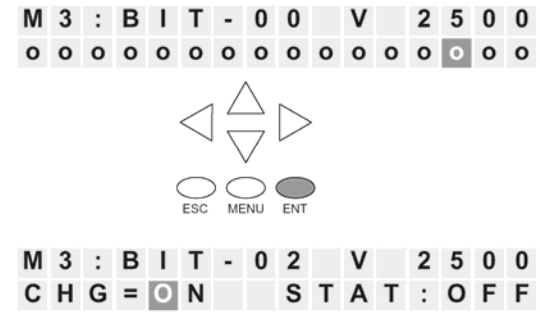
Используйте стрелки, чтобы изменить адрес по мере необходимости.



Нажмите ENT, чтобы увидеть выбранные биты.



Используйте стрелки влево и вправо, чтобы выбрать бит, чье состояние Вы хотите изменить. Нажмите ENT один раз, чтобы увидеть экран изменения состояния. Нажмите ENT снова, чтобы изменить состояние с ВЫКЛ на ВКЛ или с ВКЛ на ВЫКЛ.

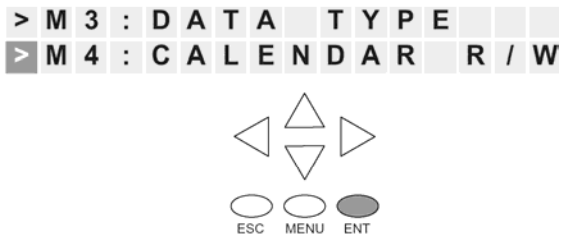


Изменение даты и времени

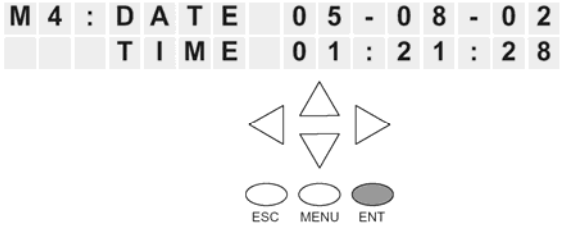
Menu 4, M4 : CALENDAR R/W

Из экрана по умолчанию нажмите MENU четыре раза для перехода к шагу 4.1.

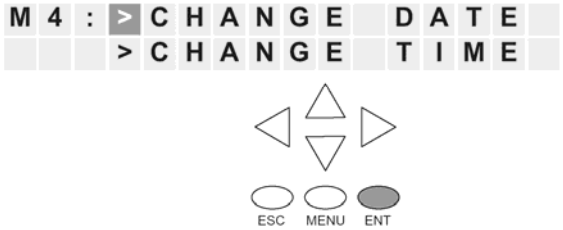
Шаг 4.1



Шаг 4.2

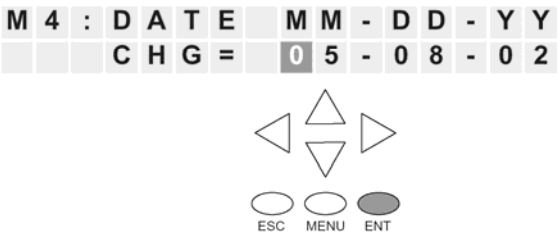


Шаг 4.3



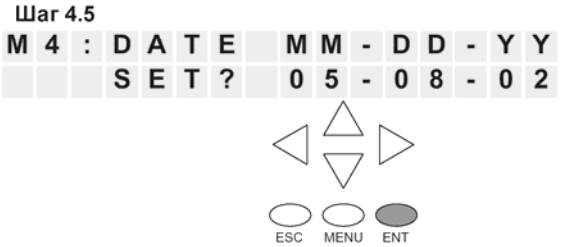
В Шаге 4.4, используйте стрелки вверх и вниз, чтобы изменить значение месяца, дня, или года. Используйте стрелки влево и вправо, чтобы передвигаться между различными цифрами в дате. После ввода необходимых изменений, нажмите ENT, чтобы зарегистрировать изменения.

Шаг 4.4



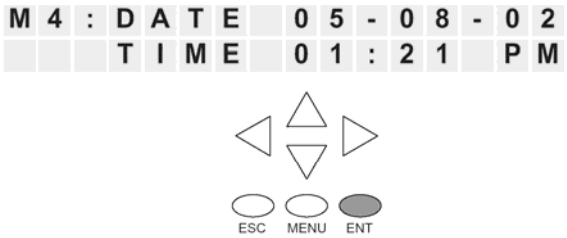
Контроллер спросит, хотите ли Вы установить дату в указанное значение. Нажмите ENT снова, если дата правильна. Вы автоматически возвратитесь на шаг 4.2, и новая дата будет показана.

Шаг 4.5

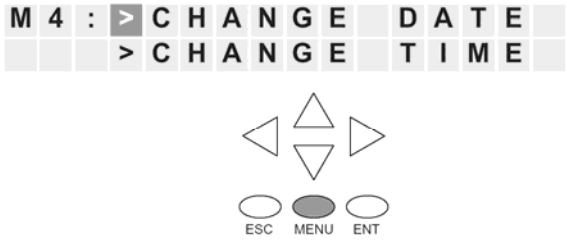


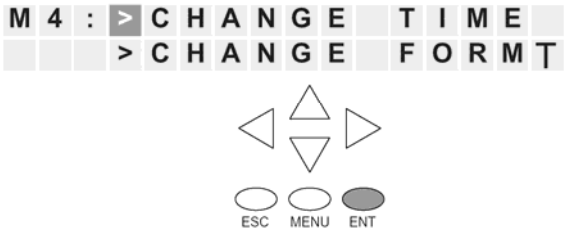
Возврат к шагу 4.2

Чтобы изменить время или формат даты / времени, нажмите ENT снова в Шаге 4.2.

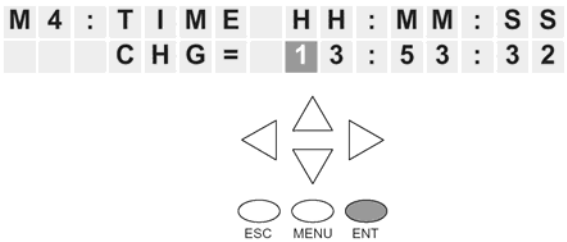


Используйте стрелки вверх или вниз или клавишу МЕНЮ, чтобы просмотреть пункты подменю. Далее в нашем примере, мы изменим установку времени.

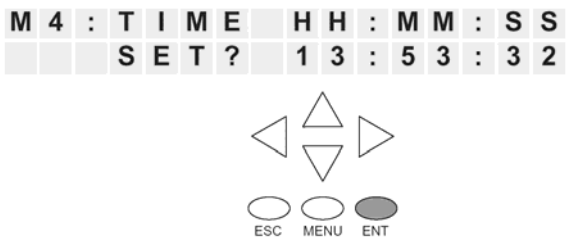




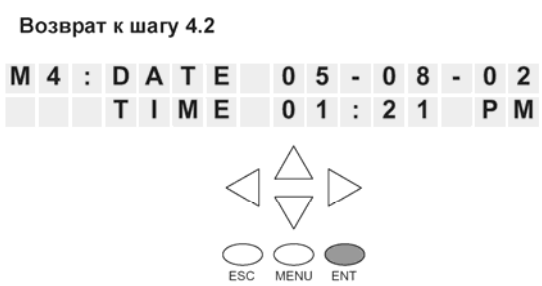
В Шаге 4.4, используйте стрелки вверх и вниз, чтобы изменить значение часов, минут, или секунд. Используйте стрелки влево и вправо, для перехода между цифрами времени. После ввода необходимых изменений, нажмите ENT, чтобы зарегистрировать изменения.



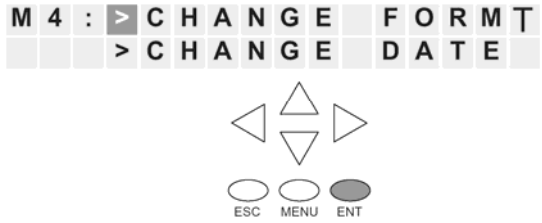
Контроллер спросит, хотите ли Вы установить время в набранное значение. Нажмите ENT снова, если время правильное. Вы автоматически возвратитесь на шаг 4.2 и новое значение времени будет показана.



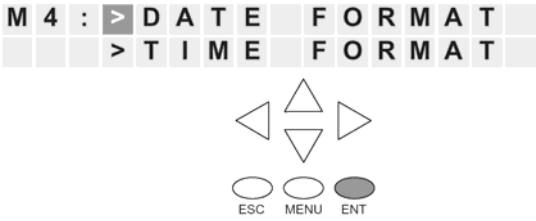
Если Вы хотите изменить формат для даты или времени, вернитесь на шаг 4.2 и нажмите ENT.



Нажмите ENT, MENU, MENU, чтобы перейти в пункт меню для изменения формата даты или времени. Нажмите ENT снова, чтобы войти в ячейку изменения формата.

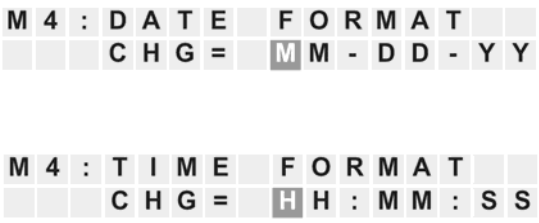


Нажмите снова ENT, чтобы войти в ячейку изменения формата даты, или нажмите МЕНЮ, ENT, чтобы изменить формат времени.



В Шаге 4.4, используйте стрелки вверх и вниз, чтобы просмотреть форматы даты.
Существуют следующие варианты:

- MM-DD-YY(Американский формат)
- DD-MM-YY (Европейский формат)
- YY-MM-DD (Азиатский формат)



Нажмите ENT, чтобы сохранить изменение формата.
Если Вы выбрали изменение формата времени, то существуют следующие варианты:

- HH:MM US (12-часовой 12:00 - 11:59AM/PM американский формат)
- HH:MM AS (12-часовой 00:00 - 11:59AM/PM Азиатский формат)
- HH:MM:SS (24–часовой формат)

Нажмите ENT, чтобы сохранить изменение формата. Нажимайте ESC пока не появится вновь экран по умолчанию.

Переменные Date и Time и их форматы		
_date:us	Американский формат	MM/DD/YY
_date:e	Европейский формат	DD/MM/YY
_date:a	Азиатский формат	YY/MM/DD
_time:12	12-часовой формат	HH:MM AM/PM
_time:24	24-часовой формат	HH:MM:SS

Установка пароля и блокировка

Menu 5, M5 : PASSWORD R/W

LCD-панель имеет собственную защиту паролем отдельную от “пароля релейной программы” для защиты ПЛК. Пароль LCD-панели может использоваться, чтобы предотвратить несанкционированные изменения настроек часов, календаря или значений данных V-памяти. Персонал, вводящий правильный пароля может изменять часы, календарь, значения V-памяти, форсировать биты на включение/отключение, и т.д.

Пароль LCD-панели запрещает неавторизованному персоналу проводить изменения данных в DL06 с клавиатуры LCD-панели. Даже при том, что пароль LCD-панели установлен, пользователь все же может изменять данные в DL06 с помощью DirectSOFT32 или D2-HPP. LCD-панель не поддерживает многоуровневых паролей.

Только пункт меню 5 LCD-панели может изменять пароль.

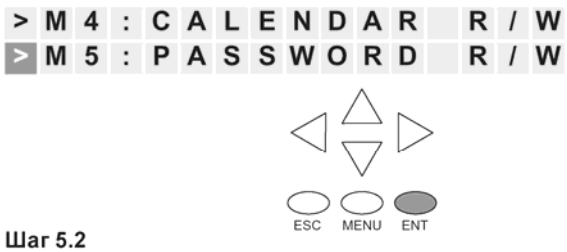
ПРЕДУПРЕЖДЕНИЕ: защита пароля, доступная в DirectSOFT32 или HPP не запрещает изменения данных с LCD-панели. Чтобы закрыть доступ к изменениям с LCD-панели, необходимо использовать пароль самой LCD-панели.

Используйте клавишу MENU, чтобы перейти к пункту меню M5. Нажмите ENT, чтобы перейти к экрану, показанному как Шаг 5.2.

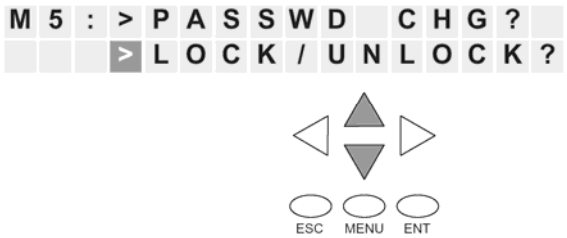
Назначение пароля без блокировки разрешает доступ ко всем возможностям при работе с LCD-панелью.

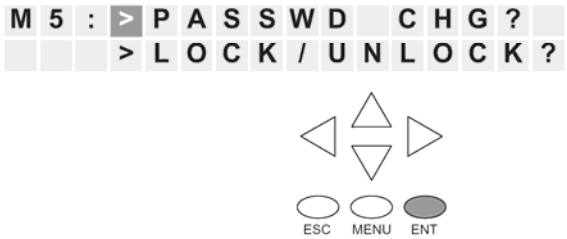
Используйте стрелки вверх или вниз для переключения между PASSWD CHG? и LOCK/UNLOCK? (блокировка/разблокировка) Восемь нолей удаляют пароль. Если пароль равен восьми нулям, то панель не будет заблокирована.

Шаг 5.1

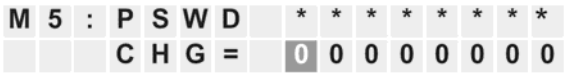


Шаг 5.2

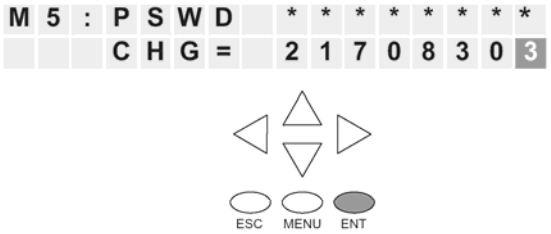


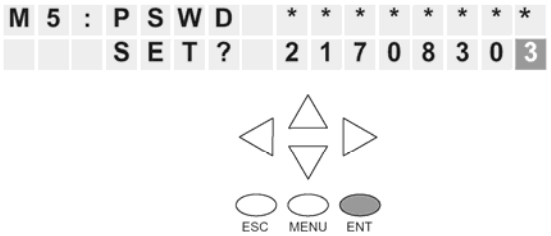


Используйте стрелки вверх или вниз, чтобы выбрать число, а стрелки вправо и влево, чтобы выбрать требуемую позицию.

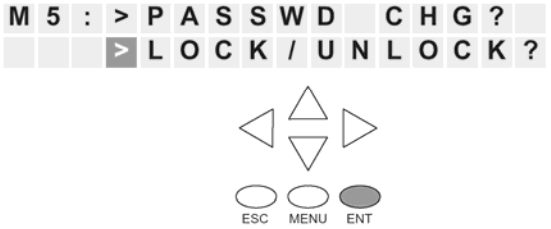


Примечание: важно делать запись пароля там, где он не будет забыт и выдавать пароль только квалифицированному персоналу. Полный доступ к LCD-панели дает доступ на изменение значений данных в памяти ПЛК.

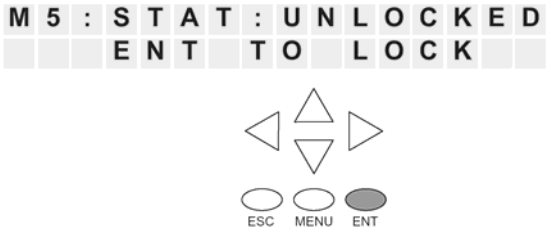




Вохврат к шагу 5.2

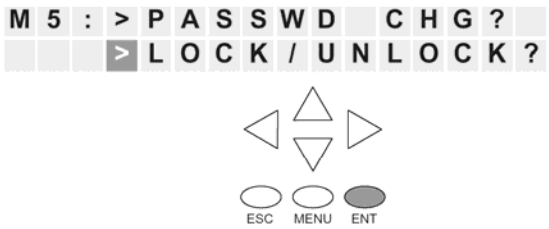


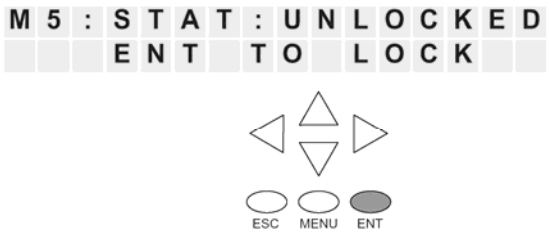
Не возможно заблокировать панель без назначения пароля. Возможно назначить пароль блокировки панели, но в этом случае данные контроллера не будут защищены.
Нажмите ENT в Шаге 5.2, и панель будет заблокирована. Если Вы не желаете блокировать панель, то нажмите ESC.



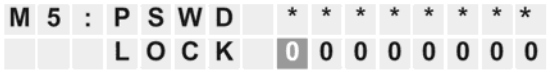
Перед назначением пароля, Вы можете выбрать, “Lock/Unlock”, нажимая ENT в Шаге 5.2.

Возврат к шагу 5.2





Здесь, экран запрашивает у Вас ввод пароля.



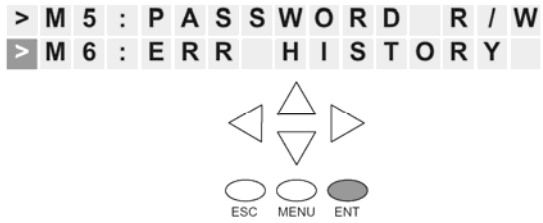
Просмотр истории ошибок

Menu 6, M6 : ERR HISTORY

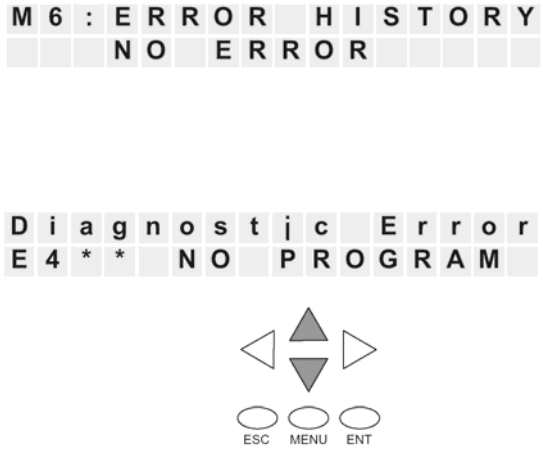
Из экрана по умолчанию, нажмите MENU шесть раз, чтобы перейти к Шагу 6.1.

Экран по умолчанию

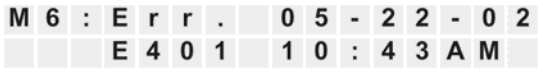
Шаг 6.1



Экран истории ошибок покажет "NO ERROR", если нет отчета об ошибках. Если ошибки происходили, они будут идентифицированы их Кодом Ошибки. Таблица Кодов Ошибок (см. приложение B) объяснит источник возникновения сообщения об ошибке. Будут показаны последние 16 сообщений. Сообщения об ошибке будут смещены при возникновении нового сообщения об ошибке.



Для просмотра прошлых сообщений об ошибках используют стрелку вниз, чтобы просмотреть отчет сообщений об ошибках.



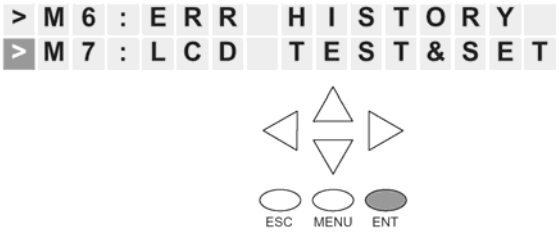
Переключение подсветки и устройства звуковой сигнализации, тест клавиатуры.

Menu 7, M7 : LCD TEST&SET

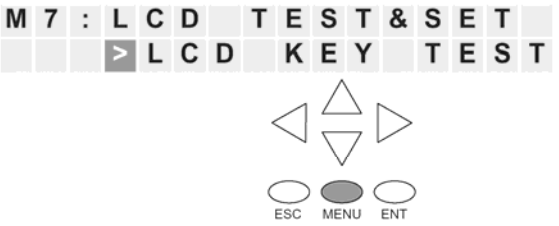
Этот пункт меню дает Вам возможность:

- Проверить каждую клавишу LCD-панели, чтобы знать, что ПЛК получает соответствующий сигнал,
- Включить/Отключить устройство звуковой сигнализации,
- Включить/Отключитьподсветку LCD-панели.

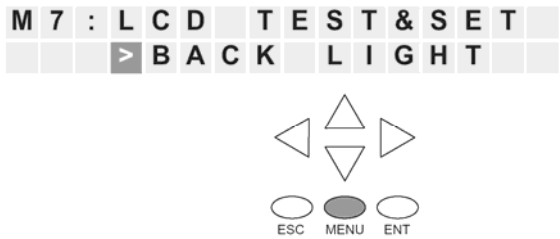
Выберите пункт меню, нажатием на клавишу ENT.



Нажмите ENT, чтобы войти в тест клавиатуры LCD KEY TEST. Все клавиши могут быть проверены на функционирование в этом меню.
Чтобы вернуться в меню, нажмите ESC, дважды или удерживайте ESC нажатой, пока вновь не появиться требуемый уровень меню.



Нажмите ENT, чтобы войти в меню проверки подсветки панели.



Пьезоэлектрический зуммер может быть сконфигурирован так, чтобы обеспечить обратную связь для кнопок.

Память контроллера с информацией для LCD-панели

Диапазоны памяти для хранения текстовых сообщений в DL06:
V400 - V677; V1200 - V7577; V10000 - V17777

Индексы форматов для встроенных данных V-памяти

Несколько форматов данных доступны для представления данных V-памяти на LCD-панели. Варианты показаны в таблице ниже. Двоеточие используется, чтобы отделить, встроенную ячейку V-памяти от индекса формата данных и модификатора.

Формат данных	Модификатор		Пример	Положение символа /Содержимое выхода												
Не указан (16-битное двоичное число в HEX-формате)			V2000 = 0012	**1**	**2**	**3**	**4**									
			V2000	s	s	1	8									
	S	[:S]	V2000:S	1	8											
	C0	[:C0]	V2000:C0	0	0	1	8									
	0	[:0]	V2000:0	s	s	1	8									
:B (4-знаковое BCD)			V2000 = 0012	1	2	3	4									
		[:B]	V2000:B	0	0	1	2									
	S	[:BS]	V2000:BS	1	2											
	C0	[:BC0]	V2000:BC0	0	0	1	2									
	0	[:B0]	V2000:B0	s	s	1	2									
:D (32-битное двоичное)			V2000 = 0000													
			V2001 = 0001	1	2	3	4	5	6	7	8	9	10	11		
		[:D]	V2000:D	s	s	s	s	s	s	6	5	5	3	6		
	S	[:DS]	V2000:DS	6	5	5	3	6								
	C0	[:DC0]	V2000:DC0	0	0	0	0	0	0	6	5	5	3	6		
	0	[:D0]	V2000:D0	s	s	s	s	s	s	6	5	5	3	6		
:DB (8-знаковое BCD)			V2000 = 0000													
			V2001 = 0001	1	2	3	4	5	6	7	8					
		[:DB]	V2000:DB	0	0	0	1	0	0	0	0					
	S	[:DBS]	V2000:DBS	1	0	0	0	0								
	C0	[:DBC0]	V2000:DBC0	0	0	0	1	0	0	0	0					
	0	[:DB0]	V2000:DB0	s	s	s	1	0	0	0	0					
:R (вещественное число с плавающей запятой)			Value = 222.11111													
			V2000 = 1C72													
			V2001 = 435E	1	2	3	4	5	6	7	8	9	10	11	12	13
		[:R]	V2000:R	s	s	s	f	2	2	2	.	1	1	1	1	1
	S	[:RS]	V2000:RS	f	2	2	2	.	1	1	1	1	1			
	C0	[:RC0]	V2000:RC0	f	0	0	0	2	2	2	.	1	1	1	1	1
	0	[:R0]	V2000:R0	s	s	s	f	2	2	2	.	1	1	1	1	1
:E (вещественное число в экспоненциальной форме)			Value = 222.1													
			V2000 = 199A													
			V2001 = 435E	1	2	3	4	5	6	7	8	9	10	11	12	13
		[:E]	V2000:E	s	f	2	.	2	2	1	0	0	E	+	0	2
	S	[:ES]	V2000:ES	f	2	.	2	2	1	0	0	E	+	0	2	
	C0	[:EC0]	V2000:EC0	f	2	.	2	2	1	0	0	E	+	0	2	
	0	[:E0]	V2000:E0	f	2	.	2	2	1	0	0	E	+	0	2	
s = пробел, f = флаг плюс/минус (плюс = нет символа, минус = -)																

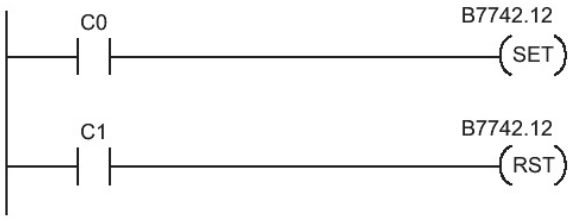
Модификаторы S, C0 и 0 изменяют представление нулевых старших разрядов и пробелов. S удаляет пробелы в старших разрядах и результат перемещает влево. C0 заменяет пробелы в старших разрядах нулями. 0 – модификация C0. Модификатор 0 в версии формата C0 нули в старших разрядах заменяет пробелами.

Резервирование памяти регистров для LCD-панели

Два регистра V-памяти зарезервированы для изменения функций LCD-панели из программы на релейной логике. Ячейка V7742 содержит битовые флаги, которые могут быть установлены в релейной программе. Битовые флаги позволяют изменять форматы данных, управлять подсветкой экрана, и устройством звуковой сигнализации. Все флаги ячейки V7742 определены в таблице на следующей странице.
Другой зарезервированный регистр - V7743. Этот регистр используется, чтобы записать пользовательское сообщение для экрана по умолчанию. Типовая программа для этой цели иллюстрируется ниже в этой главе.

Адрес V-памяти	Содержание
V7742	Различные флаги LCD: • Формат даты и времени • Меню по умолчанию • Формат отображаемых данных • Флаг состояния пароля LCD-панели • Установка подтверждения нажатия клавиши звуковым сигналом включено/отключено • Настройка подсветки экрана: включено/отключено
V7743	Ячейка сообщения по умолчанию (запись 0 в эту ячейку возвращает сообщение по умолчанию к заводской установке)

Следующая часть программы использует команды SET и RST, чтобы переключить из состояния включено в состояние выключено бит 12 ячейки V7742. Когда C0 включен, бит 12 включается. Бит 12 включает устройство звуковой сигнализации LCD-панели. Контакт C1 сбрасывает бит 12 в состояние выключено.



Назначения бит ячейки памяти V7742

Бит	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
V7742	*	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Бит 1, 0	Формат отображения даты (по умолчанию = 00)		
	00, 11	=	Месяц/День/Год (американский формат)
	01	=	День/Месяц/Год (европейский формат)
	10	=	Год/Месяц/День (Азиатский формат)
Бит 3, 2	Формат отображения времени (по умолчанию = 00))		
	00, 11	=	HH:MM:SS (24 –часовой формат)
	01	=	HH:MM PM/AM (12–часовой американский формат 12:00-11:59)
	10	=	HH:MM PM/AM (12-часовой азиатский формат 00:00-11:59)
Бит 6 - 4	Установка меню по умолчанию (по умолчанию = 000)		
	000	=	Последовательность пунктов меню по умолчанию, начало с пункта меню 1
	001	=	начало с пункта меню 1
	010	=	начало с пункта меню 2
	011	=	начало с пункта меню 3
	100	=	начало с пункта меню 4
	101	=	начало с пункта меню 5
	110	=	начало с пункта меню 6
	111	=	начало с пункта меню 7
Бит 8	Формат отображаемых данных (по умолчанию = 0)		
	0	=	BCD/HEX формат (0000 - FFFF)
	1	=	Десятичный формат (00000 - 65535)
Бит 9	Перезапись нового сообщения (по умолчанию = 0)		
	0	=	Новое LCD сообщение очищает оба ряда от предыдущего сообщения
	1	=	Новое LCD сообщение оставляет предыдущее сообщение, переписывая только определенные символы
Бит 11	Флаг состояния пароля LCD-панели (только для чтения)		
	0	=	Пароль разблокирован
	1	=	Пароль заблокирован
Бит 12	Флаг состояния управления устройством звуковой сигнализации (по умолчанию = 0)		
	0	=	Звуковой сигнал отключен
	1	=	Звуковой сигнал включен (LCD подает звуковой сигнал непрерывно, пока этот флаг установлен)
Бит 13	Звуковое подтверждение нажатия клавиш (по умолчанию = 0)		
	0	=	Звуковой сигнал отключен
	1	=	Звуковой сигнал включен (LCD подает звуковой сигнал при нажатии на клавишу)
Бит 14	Флаг настройки подсветки панели (по умолчанию = 1)		
	0	=	Подсветка отключена
	1	=	Подсветка включена
Бит 15	Флаг состояния установки LCD-панели (Только для чтения)		
	0	=	LCD-панель не установлена
	1	=	LCD-панель установлена

Изменение экрана по умолчанию

При включении питания на панели будет отображен экран по умолчанию. Сообщение для экрана по умолчания установливается на заводе, но может быть изменено пользователем. Один метод настройки сообщения по умолчанию использует команду VPRINT. Работа с командой VPRINT описана в Главе 5.

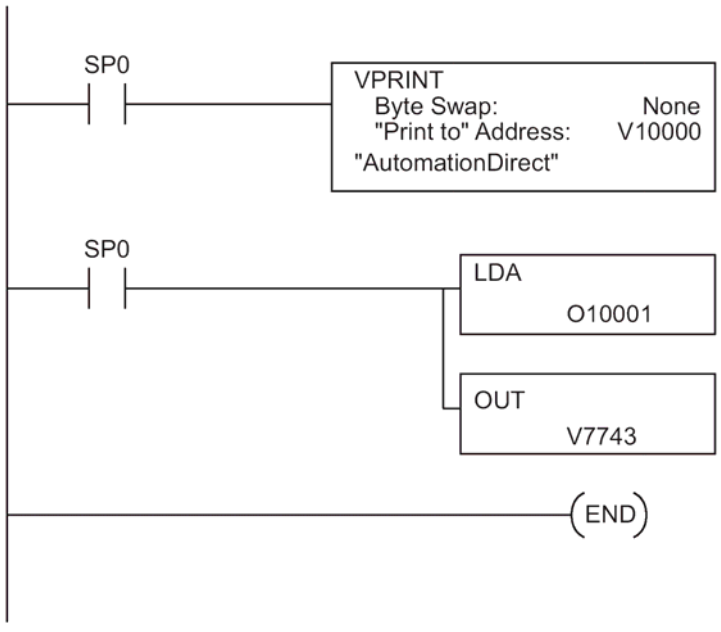
Заводское сообщение по умолчанию

```
DL06 PLC  May 08
     14:20:49
```

Пример программы для установки сообщения по умолчанию

Следующая программа может использоваться, чтобы установить сообщение для экрана по умолчанию. Эта программа использует команду VPRINT, чтобы загрузить ASCII текст в указанную ячейку V-памяти и включить указатель на текущие данные. Команды LDA и OUT используются, чтобы указать на ячейку V-памяти (+1), где расположен текст. Ячейка памяти V7743 зарезервирована для указателя на адрес сообщения по умолчанию.

Примечание: Команда VPRINT добавляет одно слово (2 байта) непечатаемого заголовка к тексту. По этой причине, LDA инструкция указывает на ячейку V-памяти V10001 , а не на V10000.



V10000	00h	16h
V10001	u	A
V10002	o	t
V10003	a	m
V10004	i	t
V10005	n	o
V10006	i	D
V10007	e	r
V10010	t	c
V10011		
V10012		
V10013		
V10014		
V10015		
V10016		
V10017		
V10020		

После выполнения этой программы, нажмите MENU, затем ESC или отключите и включите питание. Новое сообщение по умолчанию будет показано на панели таким, как на рисунке справа. Смотри Меню 4 инструкции для изменения информации о дате и времени.

```
AutomationDirect
```

Примечание: Существует возможность вернуться к заводской установке сообщения по умолчанию записав 0 в ячейку V7743 и сделав цикл питания.

Команда управления LCD-панелью DL06 (LCD)

Из папки проекта DirectSOFT32, используйте Окно «Instructions Browser» (Поиск команд), чтобы найти команду LCD. Когда Вы выберите команду LCD и нажмете на кнопку OK, появится диалоговое окно команды LCD.

Команда LCD вставляется в программу релейной логики через окно настройки, показанное справа. Оно используется, чтобы определить сообщение, которое будет показано в ряду 1 или 2 LCD-панели.

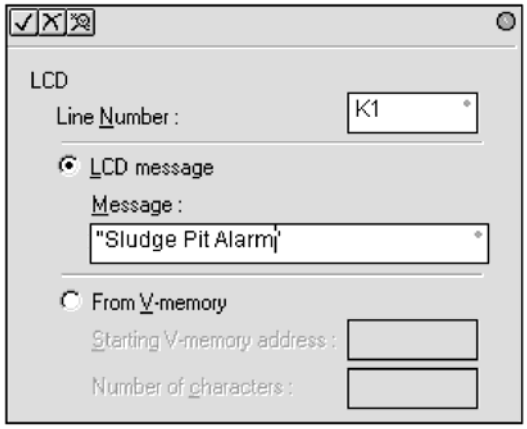
```
Sludge Pit Alarm
Effluent Overflo
```

Источник сообщения

Текст сообщения может иметь два источника. Текст может быть введен непосредственно в теле команды как строка символов (см. рисунок A), или может быть сохранен как ASCII текст в одной из ячеек V-памяти (рисунок B). В последнем случае, необходимо в диалоговом окне указать начальную ячейку V-памяти и длину текста.

Показываемые строки текста могут включать вложенные данные. Любое значение V-памяти или дата и время может быть встроена в показываемый текст.

Рисунок A



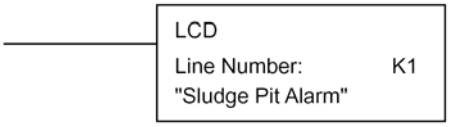
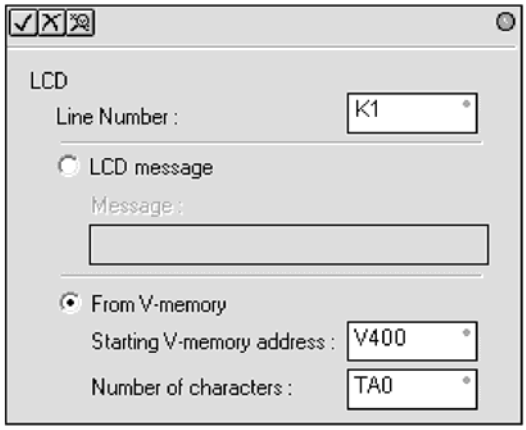
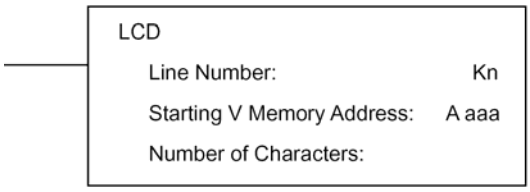


Рисунок B





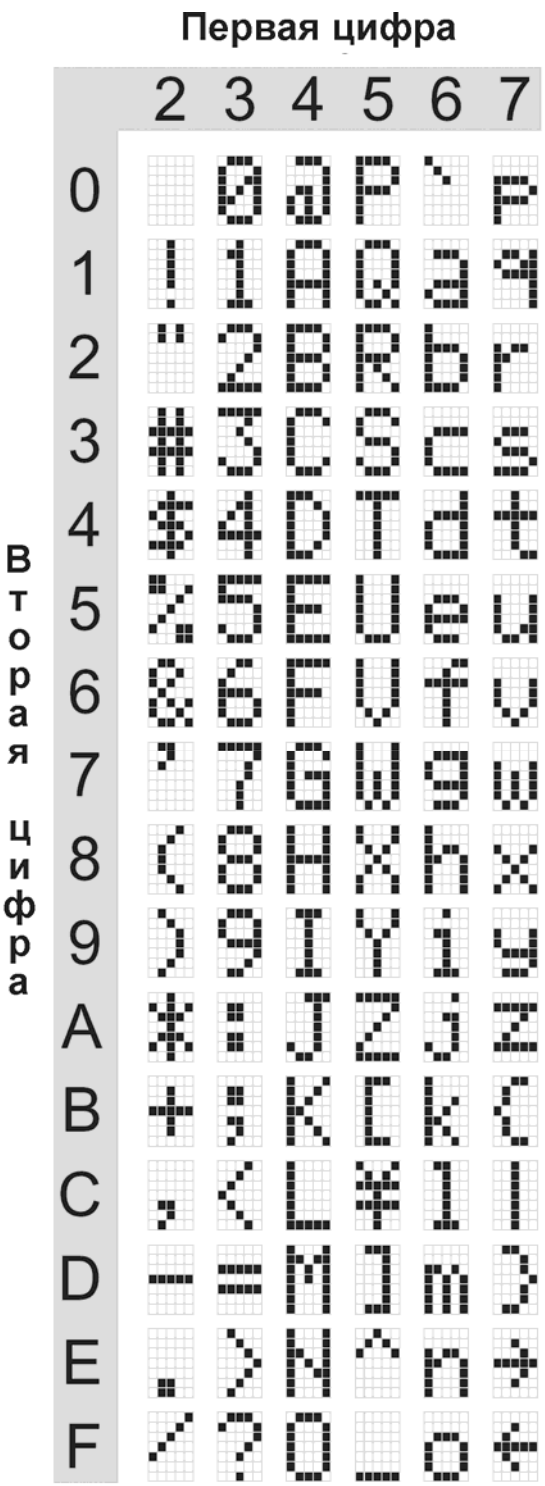
Примечание: Команда LCD поддерживается только в версии 4 пакета программирования DirectSOFT32, или более поздней, и не поддерживается в ручном программаторе D2-HPP.

Коды ASCII-символов

ASCII символы могут быть написаны непосредственно в ячейках V-памяти и затем отображены с использованием команды LCD. Таблица справа показывает BCD/HEX код с двумя цифрами для каждого символа, доступного для отображения.

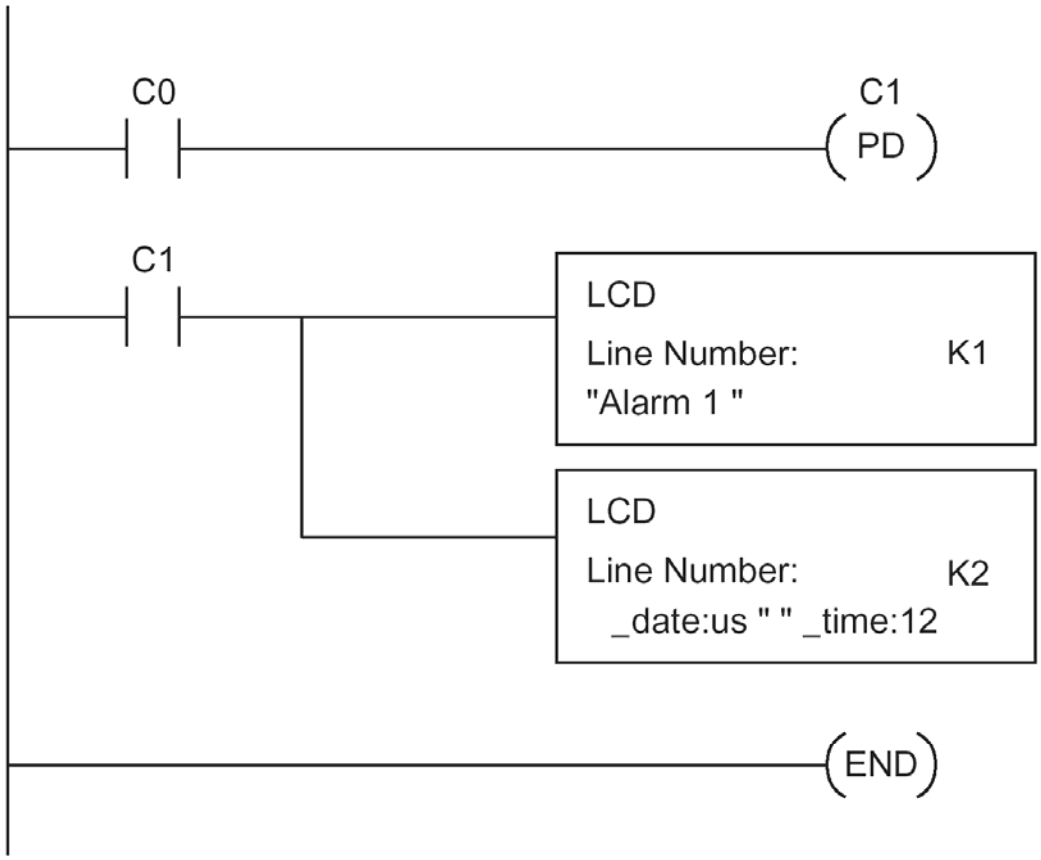
Например:
Чтобы отобразить заглавную букву A, запишите число 41(HEX) в ячейку памяти, указанную для команды LCD.

Коды ASCII-символов (BCD/HEX)



Пример программы: аварийное сообщение со встроенной меткой даты / времени

Следующая программа покажет сообщение, “Alarm 1” и время на линии K1 с датой на линии K2.
Одноразовый положительный импульс (PD), используется так, чтобы показываемые сообщения не блокировали другие сообщения или работу с меню. Нажатие клавиши MENU или ESC заставит этот текст сообщения исчезнуть.

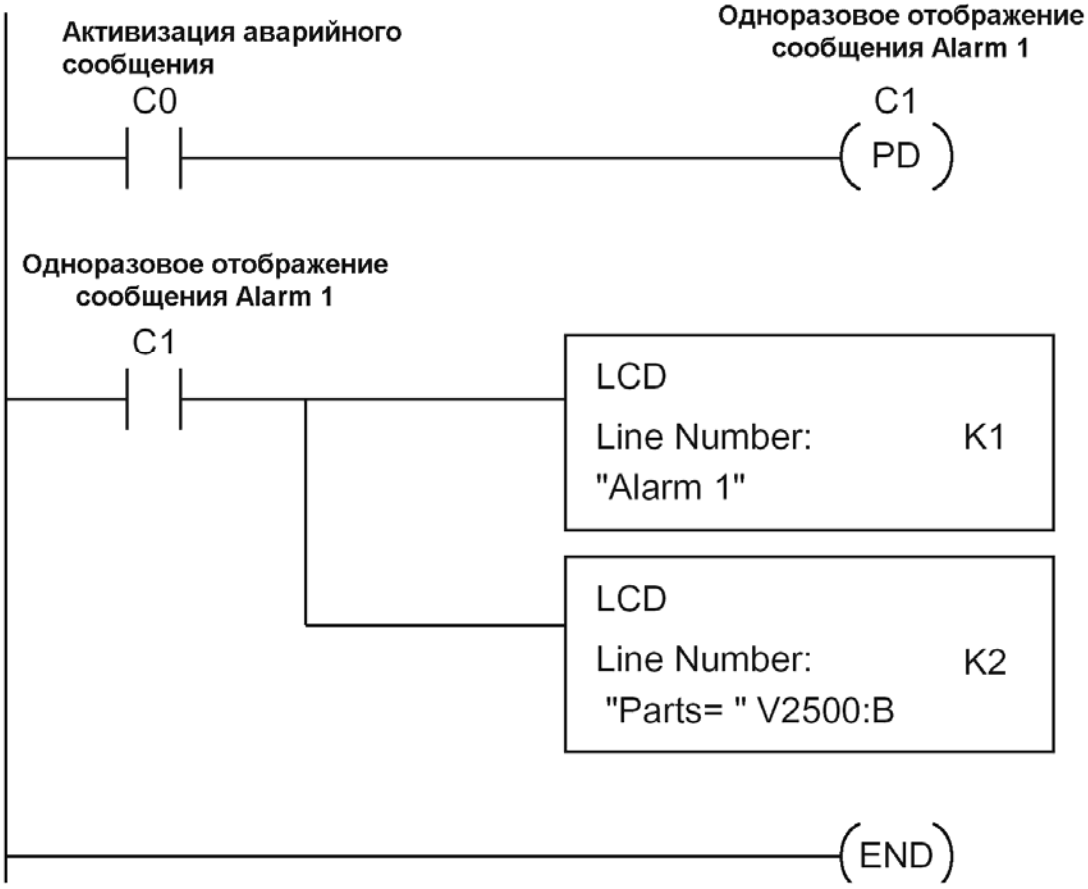


Пример программы: аварийное сообщение со встроенными данными из V-памяти

В этом примере программы, текст аварийного сообщения отображается вместе с содержанием ячейки V2500. Индекс “B” добавлен к ячейке памяти (V2500:B), чтобы данные отображались как число в BCD-формате.

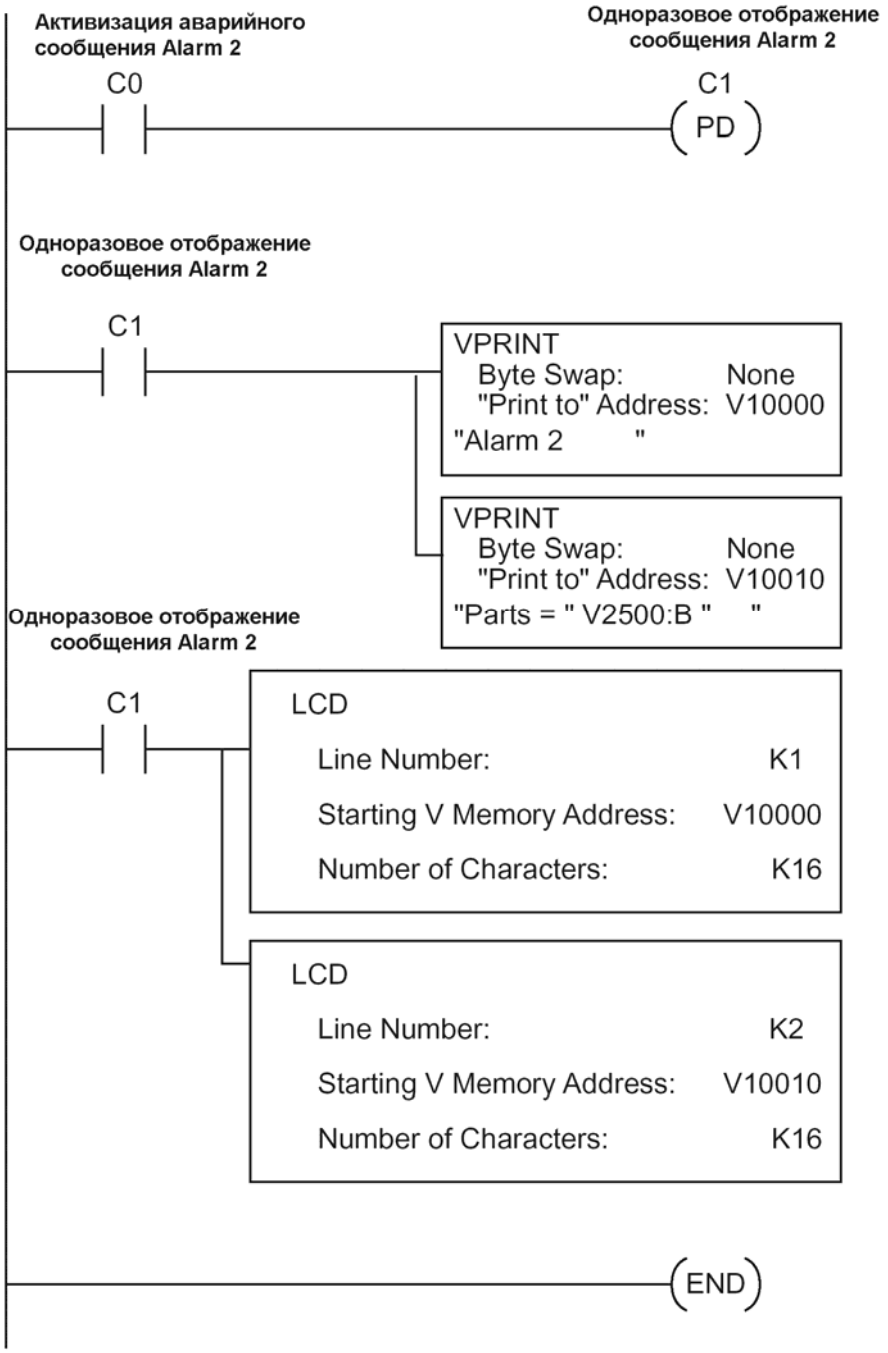
В первом примере, текст аварийного сообщения загружен непосредственно через команду LCD. Во втором примере, текст аварийного сообщения загружен в V-память, и команда LCD использует указатель на этот текст.

Примечание: При использовании команды LCD, для отображения V2000:R, имеется возможность использовать только три символа текста, потому что числа в форматеV2000:R использует 13 символов.



Пример программы: текст аварийного сообщения из V-памяти со встроенными данными из V-памяти

Этот пример программы использует команду VPRINT, чтобы записать ASCII текст (в соответствующей последовательности символов) в ячейки от V10000 до V10010. Команда LCD использует указатель на ячейку V-памяти, где находится текст для отображения на каждой строке панели.



Приложение A

11. Приложение A. Вспомогательные функции

В данной главе…

Введение

Цель вспомогательных функций

Многие задачи настройки Процессора предполагают использование Вспомогательных (AUX) функций. AUX функции выполняют много различных операций, включая очистку программной памяти, отображение времени сканирования, копирование программ в ЭППЗУ и в ручной программатор и др. Они разделены на категории, которые предназначены для различных системных параметров. Вы можете получить доступ к AUX функциям из DirectSOFT32 или с помощью Ручного Программатора D2-HPP. 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 DirectSOFT32. Несмотря на то, что в настоящем Приложении приводится достаточно много примеров AUX функций, вам следует дополнительно использовать документацию по выбранному вами устройству для программирования.

Примечание: Ручной Программатор может иметь дополнительные AUX функции, которые не поддерживаются для Процессорм DL06.

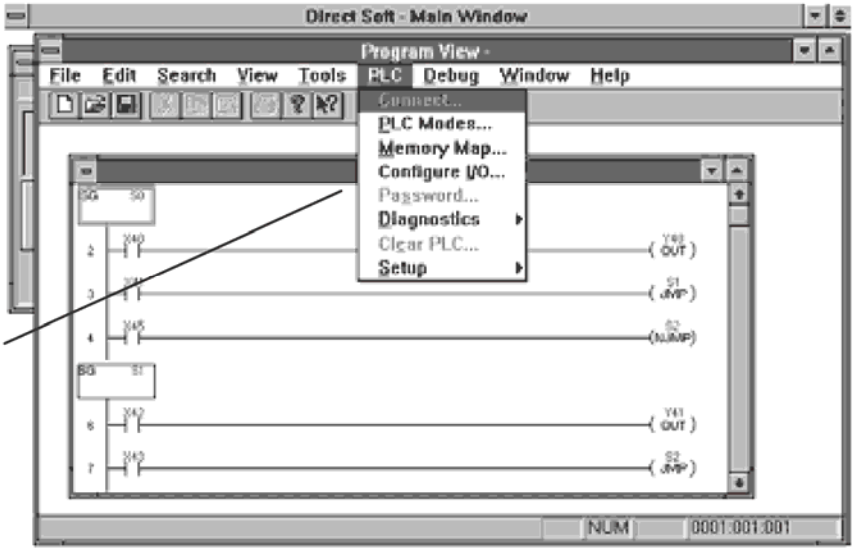
	AUX Функции и их описание	DL06
AUX 2* — Операции RLL		
21	Проверка программы	O
22	Изменение ссылки	O
23	Очистка памяти в заданном диапазоне цепей	O
24	Очистка памяти по всем цепям	O
AUX 3* — Операции с V-Памятью		
31	Очистка V-Памяти	O
AUX 4* — Конфигурация ввода/вывода		
41	Показать конфигурацию ввода/вывода	O
AUX 5* — Конфигурация процессора		
51	Изменить Имя программы	O
53	Показать Время сканирования	O
54	Инициализировать электронный блокнот-ScratchPad	O
55	Установить сторожевой таймер	O
56	Установить связь по порту 2	O
57	Установить области сохранения	O
58	Операции тестирования	O
59	Установка форсирования	O
5B	Конфигурировать высокоскоростной ввод/вывод	O
5D	Настройка скан-цикла	O

	AUX Функции и их описание	DL06
AUX 6* — Конфигурация ручного программатора		
61	Показать номер версии	O
62	Включить/отключить звуковой сигнал	HP
65	Запустить самодиагностику	HP
AUX 7* — Операции ЭППЗУ		
71	Копировать память процессора в ЭППЗУ HPP	HP
72	Записать ЭППЗУ HPP в процессор	HP
73	Сравнить процессор с ЭППЗУ HPP	HP
74	Проверить очистку памяти (ЭППЗУ HPP)	HP
75	Стереть ЭППЗУ HPP	HP
76	Показать тип ЭППЗУ (Процессор и HPP)	HP
AUX 8* — Операции с паролем		
81	Изменить пароль	O
82	Разблокировать процессор	O
83	Заблокировать процессор	O

O - поддерживается
HP - Функции ручного программатора

Доступ к AUX функциям через DirectSOFT32

DirectSOFT32 предоставляет различные опции меню как при диалоговом, так и при автономном режиме программирования. Некоторые из AUX функций могут использоваться только в диалоговом режиме программирования, некоторые — только в автономном, некоторые — и в том, и в другом. На рисунке ниже показан пример меню операции ПЛК в DirectSOFT32.



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

Вспомогательные функции Операции в RLL позволяют Вам исполнять различные действия в релейной программе.

AUX 21 Проверка программы

Как Ручной Программатор, так и DirectSOFT32 автоматически проверяют ошибки при вводе программы. Но могут возникнуть случаи, когда вам необходимо проверить программу, которая уже введена в Процессор. 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 в Приложении В приведен полный списко кодов ошибок. После исправления ошибки продолжайте синтаксическую проверку, пока не появится сообщение «NO SYNTAX ERROR» (синтаксических ошибок нет).

При Проверке Дублированных ссылок проверяется, не используете ли вы ссылку на одно и то же выходное реле 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.

Данная AUX функция доступна в DirectSOFT32 в подменю «PLC/Diagnostics».

AUX 22 — Изменение ссылок

Иногда вам необходимо изменить ссылку на адрес точки Ввода/Вывода или ссылку на управляющее реле. 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 можно выбрать и удалить часть программы. DirectSOFT32 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 чистить программную память. Данная AUX функция доступна в DirectSOFT32 через подменю «PLC/ Clear PLC memory »

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 DirectSOFT32 через подменю «PLC/Clear PLC memory»

AUX 4* — Конфигурация ввода/вывода

AUX 41 — Вывести конфигурацию ввода/вывода

Эта AUX функция позволяет вам вывести на экран текущую конфигурацию ввода/вывода DL06. В DirectSOFT32 выдается та же информация, но в гораздо более удобном виде.

AUX 5* — Конфигурирование процессора

Вы можете использовать эти AUX функции для настройки, просмотра или изменения конфигурации Процессора

AUX 51 — Изменить имя программы

В DL06 программа в Процессоре или программа, хранимая в памяти ЭППЗУ, может иметь имя. (Следует напомнить, что вы не можете иметь несколько программ в ЭППЗУ). Имя программы может быть длиной до восьми символов, в имени можно использовать любой доступный символов (A - Z, 0 - 9). AUX 51 дает вам возможность ввести имя программы. Вы можете выполнять эту операцию в DirectSOFT32 через подменю «PLC/Setup». После ввода имени программы вы можете удалить его только с помощью AUX 54 сбросом системной памяти. Перед использованием AUX 54 убедитесь, что вы поняли все возможные ветвления этой функции.

AUX 53 — Показать время сканирования

Функция AUX 53 отображает текущее, минимальное и максимальное времена сканирования. Минимальное и максимальное времена берутся с момента последнего перехода из Программного режима в Рабочий. Вы можете также выполнять эту операцию в DirectSOFT32 через подменю «PLC/Diagnostics».

AUX 54 — Инициализировать электронный блокнот

Процессор DL06 поддерживает системные параметры в области памяти, которую часто называют «электронный блокнот»-ScratchPad. В некоторых случаях Вы можете вносить изменения в настройку системы, которая хранится в системной памяти. Например, если определить область Управляющих Реле (CR) как сохраняемую, то эти изменения запоминаются.

ПРИМЕЧАНИЕ. У вас нет необходимости применять эту функцию до тех пор, пока вы не делаете изменений, которые влияют на системную память. Обычно вы нуждаетесь в инициализации системной памяти 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.

AUX 54 возвращает системную память к значениям по умолчанию. Вы можете также выполнять эту операцию в DirectSOFT32 через подменю «PLC/Setup».

AUX 55 — Установить сторожевой таймер

Процессор DL06 имеет «сторожевой» таймер –Watchdog Timer, который применяется для отслеживания времени сканирования. Его значению по умолчанию, установленное в заводских условиях, равно 200мс. Когда время сканирования превосходит установленный временной предел, Процессор автоматически выходит из Рабочего режима и переходит в режим PGM (Программный). При этом на Ручной Программатор выдается следующее сообщение: E003 S/W TIMEOUT.

Используйте AUX 55 для уменьшения или увеличения значения «сторожевого» таймера. Вы можете также выполнять эту операцию в DirectSOFT32 через подменю «PLC/Setup».

AUX 56 — Сетевой адрес процессора

Поскольку Процессор Dl06 имеет дополнительный коммуникационный порт, для установки сетевого адреса и параметров связи порта необходимо использовать Ручной Программатор. Параметрами настройки по умолчанию являются:

- Адрес станции «1»
- Шестнадцатеричный режим (HEX mode)
- Контроль по нечетности (Odd parity)

Можно использовать этот порт для подключения Ручного Программатора, DirectSOFT32 или как коммуникационный порт DirectNET и MODBUS. В Руководстве по DirectNET 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 AUX 56 для установки сетевого адреса и параметров связи порта. Вы можете также выполнять эту операцию в DirectSOFT32 через подменю «PLC/Setup/Secondary Port».

AUX 57 — Установить области сохранения памяти

Процессор DL06 обеспечивает по умолчанию определенные области сохранения памяти (Retentive Range). Некоторые области памяти поддерживаются суперконденсатором. Неразрушаемая FLASH–память находится в V7400-V7577. Установленные по умолчанию области сохранения удобны для многих приложений, но вы можете изменить их, если ваше приложение требуют дополнительных областей сохранения или не требуют никаких областей сохранения. Областями по умолчанию являются:

Зоны памяти	DL06	
	Область по умолчанию	**Доступная область**
Управляющие реле	C1000 – C1777	C0 – C1777
V- Память	V400 – V37777	V0 – V37777
Таймеры	Нет, по умолчанию	T0 – T377
Счетчики	CT0 - CT177	CT0 - CT177
Стадии	Нет, по умолчанию	S0 – S1777

Используйте AUX 57 для изменения областей сохранения. Вы можете также выполнять эту операцию в DirectSOFT32 через подменю «PLC/Setup/Retentive Range».

ПРЕДУПРЕЖДЕНИЕ. Процессоры DL06 не имеют батарейки RAM в стандартной поставке. Суперконденсатор поддержит значения при потере питания, но не более 3 недель. При некоторых условиях продолжительность сохранения может быть еще меньше, например 4-5 дня при рабочей температуре 60°C. Покупайте литиевую батарейку D2-BAT-1 отдельно.

AUX 58 — операции тестирования

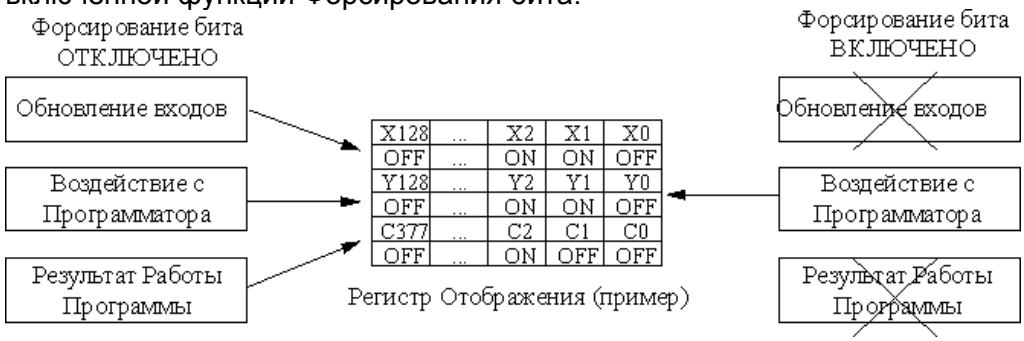
AUX 58 используется для форсирования функции блокировки выходов в команде Пауза (Pause).Используйте AUX 58 для программирования каждого отдельного выхода или диапазона выходов, которые будут работать нормально, даже если эти точки находятся в диапазоне команды Пауза.

AUX 59 — Форсирование бита

Форсирование бита (Bit Override) может быть включено для каждой точки с помощью функции AUX 59 с Ручного Программатора или с помощью DirectSOFT32. По существу при форсировании бита Процессор не может делать какие-либо изменения этого дискретного сигнала. Например, если вы включили подавление бита для X1, а X1 был временно отключен, то Процессор 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». Конечно, если X1 был включен при установлении подавления бита, то его всегда следует рассматривать как «включенный».

Использование функции форсирования бита принесет вам некоторую пользу. Нормальное форсирование не блокируется при включении форсирования бита. Например, если вы включили Форсирование Бита для Y0, а он был в это время отключен, то Процессор не сможет изменить состояние Y0. Однако вы можете использовать еще программирующее устройство для изменения этого состояния. Теперь, если вы использовали программирующее устройство для принудительного включения Y0, он останется включенным, а Процессор не сможет изменить его состояние. 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 Процессор не обновляет Регистр Отображения при включенной функции Форсирования бита.



AUX 5B — Конфигурация интерфейса счетчика

AUX 5B применяется к функции Высокоскоростного ввода/вывода (HSIO) при выборе его конфигурации. Вы можете выбрать тип счетчика, установить параметры счетчика и т. д. (Полную информацию см. в Главе 3)

AUX 5D — Выбор режима сканирования (Scan mode)

Процессор DL06 имеет два режима исполнения (сканирования) программы: фиксированный и переменный. В фиксированном режиме время цикла увеличивается до заданного времени цикла. Если реальное время цикла больше фиксированного, появляется ошибка «E504 BAD REF/VAL». В режиме с переменным циклом процессор начинает следующий цикл сразу после завершения предыдущего цикла.

AUX 6* — Настройка ручного программатора

Вы можете использовать эти несколько AUX функций для настройки, просмотра или изменения конфигурации Ручного Программатора.

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 AUX 61 вы можете быстро просмотреть номера версий Фирменного ПО Процессора и Ручного Программатора. Эта информация (по Процессору) доступна также в DirectSOFT32 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 AUX 65, чтобы запустить программу самодиагностики. Вы можете проверить:

- Клавиатуру
- Дисплей
- Светодиоды и лампы подсветки
- Электронно-перепрогаммируемое ПЗУ (ЭППЗУ) ручного программатора.

AUX 7* — Операции с ЭППЗУ

Вы можете использовать зти AUX функции для перемещения программ и выполнения других действий с программой.

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	Область памяти по умолчанию
1: PGM - Программная	$00000 - $02047
2: V - V-память	$00000 - $07777
3: SYS - Системная	Невыбираемые копии системных параметров
4: etc(ALL) - Программная, Системная и неразрушаемая V-память только.	Не выбирается

AUX 71 — Копировать из памяти ЦПУ в ЭППЗУ HPP

AUX 71 копирует информацию из памяти Процессора в ЭППЗУ, установленное в Ручном Программаторе. Вы можете копировать различные части памяти ЭППЗУ (HPP) в память Процессора в соответствии с приведенной таблицей.

AUX 72 — Копировать ЭППЗУ HPP в процессор

AUX 72 копирует информацию из ЭППЗУ, установленном в Ручном Программаторе, в Процессор. Вы можете копировать различные типы информации из памяти Процессора в соответствии с приведенной ранее таблицей.

AUX 73 — Сравнить ЭППЗУ процессора и HPP

AUX 73 сравнивает программы на Ручном Программаторе (в ЭППЗУ) с программами процессора. Вы можете сравнивать различные типы информации, как показано ранее.

AUX 74 — Проверить память (EEPROM Blanc Check)

AUX 74 дает вам возможность проверить ЭППЗУ, установленном 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

AUX 75 позволяет вам удалить все все данные в ЭППЗУ Ручного Программатора. Вы должны применять эту AUX функцию перед копированием программ из Процессора.

AUX 76 — Показать тип ЭППЗУ (процессора и HPP)

AUX 76 можно использовать для определения объема ЭППЗУ, установленного в Процессоре и в Ручном Программаторе. В DL230 и DL240 использованы ЭППЗУ различного объема. Для дополнительной информации обратитесь к Главе 3.

AUX 8* — Операции с паролем

Вы можете использовать несколько AUX функций для изменения пароля Процессора или для его включения. Вы можете применять эти функции при диалоговом общении с Процессором 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 Процессор
- AUX 83 — Заблокировать Процессор

AUX 81 — Изменить пароль

AUX 81 позволяет вам ввести особую меру защиты посредством введения пароля, с помощью которого предупреждаются несанкционированные машинные операции. Паролем должен быть 8-ми значный цифровой (0 -9) код. Введенный 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 Процессору. Блокировать Процессор с Ручного Программатора можно двумя способами:

- Процессор блокируется каждый раз после цикла включения питания (если пароль введен).
- Вы используете AUX 83 и AUX 82 для блокировки и разблокировки Процессора.

Вы можете также вводить и изменять пароль в DirectSOFT32 через подменю «Password» ПЛК. В DirectSOFT32 эта функция работает несколько иначе. Если пароль введен, Процессор автоматически блокируется, когда вы выходите из программного пакета. Он блокируется также в циклах включения питания.

ПРЕДУПРЕЖДЕНИЕ. Перед тем, как вы заблокируете Процессор, убедитесь, что вы запомнили пароль. При блокировке Процессора вы не сможете ни посмотреть, ни изменить, ни стереть этот пароль. Если вы забыли пароль, то Процессор должен быть возвращен на завод для удаления пароля.

Примечание: Процессор DL06 поддерживают многоуровневую защиту паролем релейной программы. Что позволяет организовать защиту программы паролем не блокируя порт для связи с интерфейсом оператора. Многоуровневый пароль может быть создан при помощи заглавной буквы “A” и семью числовыми символами (например. A1234567).

AUX 82 — Разблокировать процессор

AUX 82 может использоваться для разблокировки ЦПУ, защищенного паролем. DirectSOFT32 автоматически попросит Вас ввести пароль, если Вы попытаетесь взаимодействовать с ЦПУ, содержащим пароль.

AUX 83 — Заблокировать процессор

AUX 83 может использоваться для блокировки ЦПУ, который содержит пароль. После блокировке ЦПУ Вам необходимо ввести пароль, чтобы получить доступ. Напоминаем, что в DirectSOFT32 эта функция не является необходимой, так как в этом случае ЦПУ автоматически блокируется, как только Вы выходите из программного пакета.

Приложение B

12. Приложение B. Коды ошибок DL06

В данной главе…

Коды ошибок DL06

Код ошибки DL06	Описание
E001 ЦПУ FATAL ERROR	Эта ошибка может быть сброшена при отключении и включении питания микроконтроллера. Если ошибка восстановилась, то необходимо заменить DL06.
E003 SOFTWARE TIMEOUT	Эта ошибка возникает, когда время сканирования программы превышает время, установленное на сторожевом таймере. SP51 включается, а код ошибки записывается в V7755. Для исправления этой ошибки используйте AUX 55 для увеличения времени на сторожевом таймере.
E041 ЦПУ BATTERY LOW	Батарея DL06 разряжена и требует замены. SP43 включается, а код ошибки записывается в V7757
E104 WRITE FAILED	Запись в DL06 оказалась неудачной. Отключите и включите питание контроллера. Если ошибка не пропала, то замените DL06.
E151 BAD COMMAND	В программе возникла ошибка четности. SP44 включается, а код ошибки записывается в V7755. Возможно эта неисправность возникла из-за электрических помех. Очистите память и снова загрузите программу. Устраните неисправности в заземлении. Если ошибка повторяется, замените DL06..
E155 RAM FAILURE	Ошибка контрольной суммы произошла в системной RAM. SP44 будет включен, а код ошибки будет записан в V7755. Причиной может быть разряженная батарея или электрические помехи. Очистите память, и разгрузите программу снова. Проверьте заземление. Если ошибка повторяется, замените DL06.
E2** I/O MODULE FAILURE	Ошибка модуля ввода/вывода. Выполните команду AUX42, чтобы определить фактическую ошибку.
E202 MISSING I/O MODULE	Модуль ввода/вывода не сумел связаться с DL06 или отсутствует в разъме. SP45 будет включен, а код ошибки будет записан в V7756. Выполните команду AUX42, чтобы определить слот модуля, сообщавшего об ошибке.
E210 POWER FAULT	Короткое пропадание напряжения произошло в линии питания DL06.
E252 NEW I/O CFG	Эта ошибка происходит, когда автопроверка конфигурации отключена, а фактическая конфигурация ввода/вывода изменилась. Вы можете вернуть модули в первоначальное положение или выполнить AUX45, чтобы принять новую конфигурацию. SP47 будет включен, а код ошибки записывается в V7755.
E262 I/O OUT OF RANGE	Обращение к адресу вне диапазона с адресов ввода/вывода встретилось в программе. Исправьте неправильный адрес в программе. SP45 будет включен, а код ошибки записывается в V7755.
E263 CONFIGURED I/O ADDRESS OUT OF RANGE	Фактическая конфигурация ввода/вывода не совпадает с ручной конфигурацией. Откорректировать адреса можно с помощью AUX46.
E311 HP COMM ERROR 1	Запрос ручного программатора не обрабатывается DL06. Очистите ошибку и повторите запрос. Если ошибка появляется снова, замените DL06. SP46 включается, а код ошибки записывается в V7756.
E312 HP COMM ERROR 2	Обнаружена ошибка в данных при обмене с DL06. Очистите ошибку и повторите запрос. Если ошибка появляется снова, проверьте кабели между устройствами, замените Ручной Программатор, а затем и при необходимости — DL06. SP46 включается, а код ошибки записывается в V7756.

Код ошибки DL06	Описание
E313 HP COMM ERROR 3	Обнаружена ошибка в адресах при обмене с DL06. Очистите ошибочный адрес и повторите запрос. Если ошибка появляется снова, проверьте кабели между устройствами, замените Ручной Программатор, а затем и при необходимости — DL06. Код ошибки записывается в V7756.
E316 HP COMM ERROR 6	Обнаружена ошибка в режиме обменf с DL06. Удалите ошибку и повторите запрос. Если ошибка появляется снова, проверьте кабели между устройствами, замените Ручной Программатор, а затем и при необходимости — DL06. Код ошибки записывается в V7756.
E320 HP COMM TIME-OUT	Процессор не отвечает 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	Обнаружена ошибка в данных при обмене с DL06. Проверьте кабель, убедитесь, что он исправлен и правильно подсоединен. Повторите цикл включения питания и, если ошибка продолжает повторяться, сначала замените DL06, затем, если это будет необходимо, Ручной Программатор
E4** NO PROGRAM	Программа содержит синтаксическую ошибку. Наиболее частый случай — отсутствие оператора END. Запустите AUX 21, чтобы определить, какая из ошибок семейства E4** отмечена. SP52 включается, а код ошибки записывается в V7755.
E401 MISSING END STATEMENT	Программа должна заканчиваться оператором END. Введите оператор END в соответствующее место вашей программы. SP52 включается, а код ошибки записывается в V7755.
E402 MISSING LBL	Команды MOVMC или LDLBL использованы без соответствующей метки. В главе 5 есть информация по этим командам. SP52 включается, а код ошибки записывается в V7755.
E403 MISSING RET	Подпрограмма в программе не заканчивается командой RET. SP52 включается, а код ошибки записывается в V7755.
E404 MISSING FOR	Команда NEXT не имеет соответствующей команды FOR. SP52 включается, а код ошибки записывается в V7755.
E405 MISSING NEXT	Команда FOR не имеет соответствующей команды NEXT. SP52 включается, а код ошибки записывается в V7755.
E406 MISSING IRT	Подпрограмма прерывания в программе не заканчивается командой IRT. SP52 включается, а код ошибки записывается в V7755.
E412 SBR/LBL>256	В программе содержится более 256 команд SBR, или DLBL. Эта ошибка возникает также, если в программе более 4 команд INT. SP52 включается, а код ошибки записывается в V7755.
E421 DUPLICATE STAGE REFERENCE	В программе содержится две и большее число меток SG или ISG с одним и тем же номером. Допустим только уникальный номер для каждой Стадии, в т. ч. и для Начальной Стадии. SP52 включается, а код ошибки записывается в V7755.
E422 DUPLICATE LBL REFERENCE	В программе содержится две или больше команд LBL с одним и тем же номером. Допустим только уникальный номер для каждой Метки. SP52 включается, а код ошибки записывается в V7755.
E423 NESTED LOOPS	Вложенные циклы (когда один цикл FOR / NEXT находится внутри другого) не допустимы. SP52 включается, а код ошибки записывается в V7755.
E431 INVALID ISG/SG ADDRESS	ISG или SG не должны при программировании ставиться после оператора END, например, в подпрограмме. SP52 включается, а код ошибки записывается в V7755.

Код ошибки	Описание
E432 INVALID JUMP (GOTO) ADDRESS	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	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.
E434 INVALID RTC ADDRESS	RTC 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	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.
E437 INVALID IRTC ADDRESS	IRTC 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 должен 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 применена в главной программе (но после оператора END), или команда DLBL находится в цепи, содержащей входной контакт(ы).
E441 ACON/NCON	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.

Код ошибки	Описание
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.
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 INSTRUCTION	Существует команда между сходящимися стадиями.
E482 MAX. CV INSTRUCTIONS EXCEEDED	Число\ команд CV превышает 17.
E483 INVALID CV JUMP 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 MISSING REQUIRED INSTRUCTION	Команда CVJMP не размещена между командами CV и [SG, ISG, ST BLK, END BLK, END].
E486 INVALID CALL BLK ADDRESS	CALL BLK используется в подпрограмме или программе обработке прерываний. Команда CALL BLK может использоваться в только области главной программы (перед командой , чем END)
E487 MISSING ST BLK INSTRUCTION	Команда CALL BLK не имеет соотвествующей команды ST BLK.
E488 INVALID ST BLK ADDRESS	Команда ST BLK используется в подпрограмме или программе обработки прерываний. Другая команда ST BLK используется между командами CALL BLK и END BLK.
E489 DUPLICATE CR REFERENCE	Управляющее реле, используемое для команды BLK, используется как выход в другом месте.
E490 MISSING SG INSTRUCTION	Команда BLK не имеет следующей за ней команды SG.

Код ошибки	Описание
E491 INVALID ISG INSTRUCTION ADDRESS	Имеется команда ISG между командами ST BLK и END BLK.
E492 INVALID END BLK ADDRESS	Команда END BLK используется в подпрограмме или программе обработки прерываний. Команда END BLK инструкция не имеет предшествующей команды ST BLK.
E493 MISSING END REQUIRED INSTRUCTION	Команды [CV, SG, ISG, ST BLK, END] должна следовать сразу за командой END BLK.
E494 MISSING END BLK INSTRUCTION	Команда ST BLK не имеет соответсвующей команды END BLK.
E499 PRINT INSTRUCTION	Неправильное применение команды PRINT. Кавычки или пробелы не введены или введены неправильно.
E501 BAD ENTRY	На Ручном Программаторе неправильно используется клавиша или группа клавиш.
E502 BAD ADDRESS	На Ручном Программаторе введен неправильный адрес или адрес вне допустимого диапазона адресов.
E503 BAD COMMAND	На Ручном Программаторе введена неправильная команда.
E504 BAD REF/VAL	В команде неправильно введена ссылка или номер ссылки.
E505 INVALID INSTRUCTION	На Ручном Программаторе введена неправильная команда.
E506 INVALID OPERATION	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	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 PASSWORD	Неверный пароль, который используется для разблокирования Процессора с помощью AUX 82.
E542 PASSWORD RESET	Процессор включен с неправильным паролем и устанавил 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.

Код ошибки	Описание
E603 DATA MISSING	Функция поиска не нашла данные
E604 REFERENCE MISSING	Функция поиска не нашла указанную ссылку.
E610 BAD I/O TYPE	Прикладная программа сослалась на неправильный тип модуля ввода/вывода.
E620 OUT OF MEMORY	Выдается при попытке передавать между DL-06 и Ручным Программатором больше данных, чем может принять принимающее устройств0.
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.
E642 CHECKSUM ERROR	Ошибка была обнаружена при переносе данных в Картридж памяти программатора. Проверьте соединения и повторите операцию.
E650 HPP SYSTEM ERROR	В Ручном Программаторе имеет место системная ошибка. Повторно включите Ручной Программатор. Если ошибка появится снова, замените Ручной Программатор
E651 HPP ROM ERROR	В Ручном Программаторе найдена ошибка ПЗУ. Повторно включите Ручной Программатор. Если ошибка появится снова, замените Ручной Программатор.
E652 HPP RAM ERROR	В Ручном Программаторе найдена ошибка ОЗУ. Повторно включите Ручной Программатор. Если ошибка появится снова, замените Ручной Программатор.

Приложение C

13. Приложение C. Время выполнения команд

В данной главе…

Введение

В данном приложении приводятся таблицы времени выполнения команд для ПЛК DL06. 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 надо думать, что вы не можете загрузить битовую комбинацию в регистры этого типа, можете. Но основное их назначение — регистры данных. Следующие ячейки рассматриваются как регистры данных.

Регистры Данных	DL06
Текущие Значения Таймера	V0 – V377
Текущие Значения Счетчика	V1000 - V1177
Слова пользовательских данных	V1200 - V7377 V10000 – V17777

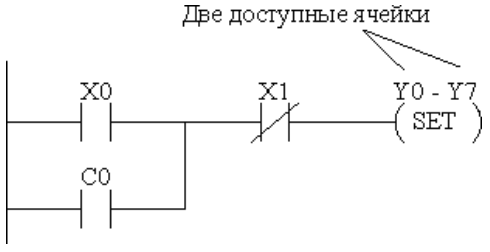
Битовые регистры V-памяти

Напоминаем вам, что некоторые из дискретных точек, например, X, Y, и др., автоматически отображаются в V-памяти. Следующие ячейки, содержащие такие данные, рассматриваются как регистры битов.

Регистры Битов	DL06
Входные точки (X)	V40400 - V40437
Выходные точки (Y)	V40500 - V40537
Управляющие Реле (C)	V40600 - V40677
Стадии(S)	V41000 - V41077
Биты Состояния Таймера	V41100 - V41177
Биты Состояния Счетчика	V41140 - V41147
Системные (специальные) реле(SP)	V41200 - V41237

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 2nd #: X, Y, C, S (N pt)	9.2 мкс 9.6мкс+0.9мкс * N
RST	1st #: X, Y, C, S 2nd #: X, Y, C, S (N pt)	9.2 мкс 9.6мкс +0.9 мкс * N

Время выполнения зависит от числа ячеек и используемого типа данных

Время выполнения команд

Булевы команды

Булевы команды		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
STR	X, Y, C, T, CT, S,SP, GX, GY	0.67 мкс	0.00 мкс
STRN	X, Y, C, T, CT, S,SP, GX, GY	0.67 мкс	0.0 мкс
OR	X, Y, C, T, CT, S,SP, GX, GY	0.51 мкс	0.51 мкс
ORN	X, Y, C, T, CT, S,SP, GX, GY	0.55 мкс	0.55 мкс
AND	X, Y, C, T, CT, S,SP, GX, GY	0.42 мкс	0.42 мкс
ANDN	X, Y, C, T, CT, S,SP, GX, GY	0.51 мкс	0.51 мкс
ANDSTR	Нет	0.37 мкс	0.37 мкс
ORSTR	Нет	0.37 мкс	0.37 мкс
OUT	X, Y, C, GX, GY	1.82 мкс	1.82 мкс
OROUT	X, Y, C, GX, GY	2.09 мкс	2.09 мкс
NOT	Нет	1.04 мкс	1.04 мкс
SET	1st #: X, Y, C, S, 2nd #: X, Y, C, S (N точек)	9.2 мкс 9.6 мкс+0.9 мкс x N	1.0 мкс 1.1 мкс
RST	1st #: X, Y, C,S, GX, GY 2nd #: X, Y, C,S (N точек), GX, GY	9.2 мкс 9.6 мкс+0.9 мкс x N	1.0 мкс 1.1 мкс
	1st #: T, CT, GX, GY 2nd #: T, CT (N точек), GX, GY	25.7 мкс 16.8 мкс + 2.7 мкс x N	1.1 мкс 1.4 мкс
PAUSE	1wd: Y 2wd: Y (N точек)	5.6 мкс 9.2 мкс + 0.3 мкс x N	5.4 мкс 4.8 мкс

Булевы команды сравнения

Булевы команды сравнения			DL06	
Команда	**Разрешенные типы данных**		**Исполняемые**	**Неисполняемые**
STRE	***1st***	***2nd***		
	V Data Reg.	V:Data Reg.	7.6 мкс	7.6 мкс
		V:Bit Reg	7.6 мкс	7.6 мкс
		K:Constant	4.8 мкс	4.8 мкс
		P:Indir. (Data)	30.2 мкс	30.2 мкс
		P:Indir. (Bit)	30.2 мкс	30.2 мкс
	V: Bit Reg.	V:Data Reg.	7.6 мкс	7.6 мкс
		V:Bit Reg	7.6 мкс	7.6 мкс
		K:Constant	4.8 мкс	4.8 мкс
		P:Indir. (Data)	30.2 мкс	30.2 мкс
		P:Indir. (Bit)	30.2 мкс	30.2 мкс
	P:Indir. (Data)	V:Data Reg	29.9 мкс	29.9 мкс
		V:Bit Reg	29.9 мкс	29.9 мкс
		K:Constant	27.7 мкс	27.7 мкс
		P:Indir. (Data)	51.0 мкс	51.0 мкс
		P:Indir. (Bit)	51.0 мкс	51.0 мкс
	P:Indir. (Bit)	V:Data Reg	29.9 мкс	29.9 мкс
		V:Bit Reg	29.9 мкс	29.9 мкс
		K:Constant	27.7 мкс	27.7 мкс
		P:Indir. (Data)	51.0 мкс	51.0 мкс
		P:Indir. (Bit)	51.0 мкс	51.0 мкс
STRNE	***1st***	***2nd***		
	V: Data Reg.	V:Data Reg.	7.6 мкс	7.6 мкс
		V:Bit Reg.	7.6 мкс	7.6 мкс
		K:Constant	4.8 мкс	4.8 мкс
		P:Indir. (Data)	30.2 мкс	30.2 мкс
		P:Indir. (Bit)	30.2 мкс	30.2 мкс
	V: Bit Reg.	V:Data Reg.	7.6 мкс	7.6 мкс
		V:Bit Reg.	7.6 мкс	7.6 мкс
		K:Constant	4.8 мкс	4.8 мкс
		P:Indir. (Data)	30.2 мкс	30.2 мкс
		P:Indir. (Bit)	30.2 мкс	30.2 мкс
	P:Indir. (Data)	V:Data Reg.	30.3 мкс	30.3 мкс
		V:Bit Reg.	30.3 мкс	30.3 мкс
		K:Constant	27.4 мкс	27.4 мкс
		P:Indir. (Data)	51.0 мкс	51.0 мкс
		P:Indir. (Bit)	51.0 мкс	51.0 мкс
	P:Indir. (Bit)	V:Data Reg.	30.3 мкс	30.3 мкс
		V:Bit Reg.	30.3 мкс	30.3 мкс
		K:Constant	27.4 мкс	27.4 мкс
		P:Indir. (Data)	51.0 мкс	51.0 мкс
		P:Indir. (Bit)	51.0 мкс	51.0 мкс

Булевы команды сравнения (продолжение)			DL06	
Команда	**Разрешенные типы данных**		**Исполняемые**	**Неисполняемые**
ORE	***1st***	***2nd***		
	V Data Reg	V:Data Reg	7.6 мкс	7.6 мкс
		V:Bit Reg	7.6 мкс	7.6 мкс
		K:Constant	4.8 мкс	4.8 мкс
		P:Indir. (Data)	30.2 мкс	30.2 мкс
		P:Indir. (Bit)	30.2 мкс	30.2 мкс
	V: Bit Reg.	V:Data Reg.	7.6 мкс	7.6 мкс
		V:Bit Reg	7.6 мкс	7.6 мкс
		K:Constant	4.8 мкс	4.8 мкс
		P:Indir. (Data)	30.2 мкс	30.2 мкс
		P:Indir. (Bit)	30.2 мкс	30.2 мкс
	P:Indir. (Data)	V:Data Reg	30.3 мкс	30.3 мкс
		V:Bit Reg	30.3 мкс	30.3 мкс
		K:Constant	27.4 мкс	27.4 мкс
		P:Indir. (Data)	50.4 мкс	50.4 мкс
		P:Indir. (Bit)	50.4 мкс	50.4 мкс
	P:Indir. (Bit)	V:Data Reg	30.3 мкс	30.3 мкс
		V:Bit Reg	30.3 мкс	30.3 мкс
		K:Constant	27.4 мкс	27.4 мкс
		P:Indir. (Data)	50.4 мкс	50.4 мкс
		P:Indir. (Bit)	50.4 мкс	50.4 мкс
ORNE	***1st***	***2nd***		
	Data Reg.	V:Data Reg.	7.6 мкс	7.6 мкс
		V:Bit Reg.	7.6 мкс	7.6 мкс
		K:Constant	4.8 мкс	4.8 мкс
		P:Indir. (Data)	30.2 мкс	30.2 мкс
		P:Indir. (Bit)	30.2 мкс	30.2 мкс
	V: Bit Reg.	V:Data Reg	7.6 мкс	7.6 мкс
		V:Bit Reg.	7.6 мкс	7.6 мкс
		K:Constant	4.8 мкс	4.8 мкс
		P:Indir. (Data)	30.2 мкс	30.2 мкс
		P:Indir. (Bit)	30.2 мкс	30.2 мкс
	P:Indir. (Data)	V:Data Reg.	29.9 мкс	29.9 мкс
		V:Bit Reg.	29.9 мкс	29.9 мкс
		K:Constant	27.4 мкс	27.4 мкс
		P:Indir. (Data)	51.0 мкс	51.0 мкс
		P:Indir. (Bit)	51.0 мкс	51.0 мкс
	P:Indir. (Bit)	V:Data Reg.	29.9 мкс	29.9 мкс
		V:Bit Reg.	29.9 мкс	29.9 мкс
		K:Constant	27.4 мкс	27.4 мкс
		P:Indir. (Data)	51.0 мкс	51.0 мкс
		P:Indir. (Bit)	51.0 мкс	51.0 мкс

Булевы команды сравнения (продолжение)			**DL06**	
Команда	**Разрешенные типы данных**		**Исполняемые**	**Неисполняемые**
ANDE	***1st***	***2nd***		
	V Data Reg.	V:Data Reg	7.6 мкс	7.6 мкс
		V:Bit Reg	7.6 мкс	7.6 мкс
		K:Constant	4.8 мкс	4.8 мкс
		P:Indir. (Data)	30.2 мкс	30.2 мкс
		P:Indir. (Bit)	30.2 мкс	30.2 мкс
	V: Bit Reg.	V:Data Reg	7.6 мкс	7.6 мкс
		V:Bit Reg	7.6 мкс	7.6 мкс
		K:Constant	4.8 мкс	4.8 мкс
		P:Indir. (Data)	30.2 мкс	30.2 мкс
		P:Indir. (Bit)	30.2 мкс	30.2 мкс
	P:Indir. (Data)	V:Data Reg	29.9 мкс	29.9 мкс
		V:Bit Reg	29.9 мкс	29.9 мкс
		K:Constant	27.4 мкс	27.4 мкс
		P:Indir. (Data)	51.0 мкс	51.0 мкс
		P:Indir. (Bit)	51.0 мкс	51.0 мкс
	P:Indir. (Bit)	V:Data Reg	29.9 мкс	29.9 мкс
		V:Bit Reg	29.9 мкс	29.9 мкс
		K:Constant	27.4 мкс	27.4 мкс
		P:Indir. (Data)	51.0 мкс	51.0 мкс
		P:Indir. (Bit)	51.0 мкс	51.0 мкс
ANDNE	***1st***	***2nd***		
	V: Data Reg.	V:Data Reg.	7.6 мкс	7.6 мкс
		V:Bit Reg.	7.6 мкс	7.6 мкс
		K:Constant	4.8 мкс	4.8 мкс
		P:Indir. (Data)	30.2 мкс	30.2 мкс
		P:Indir. (Bit)	30.2 мкс	30.2 мкс
	V: Bit Reg.	V:Data Reg.	7.6 мкс	7.6 мкс
		V:Bit Reg	7.6 мкс	7.6 мкс
		K:Constant	4.8 мкс	4.8 мкс
		P:Indir. (Data)	30.2 мкс	30.2 мкс
		P:Indir. (Bit)	30.2 мкс	30.2 мкс
	P:Indir. (Data)	V:Data Reg.	29.9 мкс	29.9 мкс
		V:Bit Reg.	29.9 мкс	29.9 мкс
		K:Constant	27.4 мкс	27.4 мкс
		P:Indir. (Data)	51.0 мкс	51.0 мкс
		P:Indir. (Bit)	51.0 мкс	51.0 мкс
	P:Indir. (Bit)	V:Data Reg.	29.9 мкс	29.9 мкс
		V:Bit Reg.	29.9 мкс	29.9 мкс
		K:Constant	27.4 мкс	27.4 мкс
		P:Indir. (Data)	51.0 мкс	51.0 мкс
		P:Indir. (Bit)	51.0 мкс	51.0 мкс

Булевы команды сравнения (продолжение)			DL06	
Команда	**Разрешенные типы данных**		**Исполняемые**	**Неисполняемые**
STR	***1st***	***2nd***		
	T, CT	V:Data Reg.	7.6 µs	7.6 µs
		V:Bit Reg	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	V Data Reg	V:Data Reg.	7.6 µs	7.6 µs
		V:Bit Reg	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	7.6 µs	7.6 µs
		V:Bit Reg	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg.	29.9 µs	29.9 µs
		V:Bit Reg	29.9 µs	29.9 µs
		K:Constant	27.4 µs	27.4 µs
		P:Indir. (Data)	51.0 µs	51.0 µs
		P:Indir. (Bit)	51.0 µs	51.0 µs
	P:Indir. (Bit)			
		V:Data Reg	29.9 µs	29.9 µs
		V:Bit Reg	29.9 µs	29.9 µs
		K:Constant	27.4 µs	27.4 µs
		P:Indir. (Data)	51.0 µs	51.0 µs
		P:Indir. (Bit)	51.0 µs	51.0 µs
STRN	***1st***	***2nd***		
	T, CT	V:Data Reg.	7.6 µs	7.6 µs
		V:Bit Reg.	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	V: Data Reg.	V:Data Reg.	7.6 µs	7.6 µs
		V:Bit Reg	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit	30.2 µs	30.2 µs

Булевы команды сравнения (продолжение)			DL06	
Команда	**Разрешенные типы данных**		**Исполняемые**	**Неисполняемые**
STRN (продолжение)	***1st***	***2nd***		
	V: Bit Reg	V:Data Reg.	7.6 µs	7.6 µs
		V:Bit Reg.	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg.	29.9 µs	29.9 µs
		V:Bit Reg.	29.9 µs	29.9 µs
		K:Constant	27.4 µs	27.4 µs
		P:Indir. (Data)	51.0 µs	51.0 µs
		P:Indir. (Bit)	51.0 µs	51.0 µs
	P:Indir. (Bit)	V:Data Reg.	29.9 µs	29.9 µs
		V:Bit Reg.	29.9 µs	29.9 µs
		K:Constant	27.4 µs	27.4 µs
		P:Indir. (Data)	51.0 µs	51.0 µs
		P:Indir. (Bit)	51.0 µs	51.0 µs

Булевы команды сравнения (продолжение)			DL06	
Команда	**Разрешенные типы данных**		**Исполняемые**	**Неисполняемые**
OR	***1st***	***2nd***		
	T, CT	V Data Reg	7.6 µs	7.6 µs
		V:Bit Reg	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	V Data Reg.	V:Data Reg.	7.6 µs	7.6 µs
		V:Bit Reg	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	7.6 µs	7.6 µs
		V:Bit Reg	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg	29.9 µs	29.9 µs
		V:Bit Reg	29.9 µs	29.9 µs
		K:Constant	27.4 µs	27.4 µs
		P:Indir. (Data)	51.0 µs	51.0 µs
		P:Indir. (Bit)	51.0 µs	51.0 µs
	P:Indir. (Bit)	V:Data Reg	29.9 µs	29.9 µs
		V:Bit Reg	29.9 µs	29.9 µs
		K:Constant	27.4 µs	27.4 µs
		P:Indir. (Data	51.0 µs	51.0 µs
		P:Indir. (Bit)	51.0 µs	51.0 µs

Булевы команды сравнения (продолжение)			**DL06**	
Команда	**Разрешенные типы данных**		**Исполняемые**	**Неисполняемые**
ORN	***1st***	***2nd***		
	T, CT	V:Data Reg.	7.6 µs	7.6 µs
		V:Bit Reg	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	V: Data Reg	V:Data Reg	7.6 µs	7.6 µs
		V:Bit Reg	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	7.6 µs	7.6 µs
		V:Bit Reg.	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg.	29.9 µs	29.9 µs
		V:Bit Reg.	29.9 µs	29.9 µs
		K:Constant	27.4 µs	27.4 µs
		P:Indir. (Data)	51.0 µs	51.0 µs
		P:Indir. (Bit)	51.0 µs	51.0 µs
	P:Indir. (Bit)	V:Data Reg.	29.9 µs	29.9 µs
		V:Bit Reg.	29.9 µs	29.9 µs
		K:Constant	27.4 µs	27.4 µs
		P:Indir. (Data)	51.0 µs	51.0 µs
		P:Indir. (Bit)	51.0 µs	51.0 µs

Булевы команды сравнения (продолжение)			DL06	
Команда	**Разрешенные типы данных**		**Исполняемые**	**Неисполняемые**
AND	***1st***	***2nd***		
	T, CT	V Data Reg.	7.6 µs	7.6 µs
		V:Bit Reg	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	V Data Reg.	V:Data Reg	7.6 µs	7.6 µs
		V:Bit Reg	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	7.6 µs	7.6 µs
		V:Bit Reg	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg	29.9 µs	29.9 µs
		V:Bit Reg	29.9 µs	29.9 µs
		K:Constant	27.4 µs	27.4 µs
		P:Indir. (Data)	51.0 µs	51.0 µs
		P:Indir. (Bit)	51.0 µs	51.0 µs
	P:Indir. (Bit)	V:Data Reg	29.9 µs	29.9 µs
		V:Bit Reg	29.9 µs	29.9 µs
		K:Constant	27.4 µs	27.4 µs
		P:Indir. (Data)	51.0 µs	51.0 µs
		P:Indir. (Bit)	51.0 µs	51.0 µs

Булевы команды сравнения (продолжение)			**DL06**	
Команда	**Разрешенные типы данных**		**Исполняемые**	**Неисполняемые**
ANDN	***1st***	***2nd***		
	T, CT	V:Data Reg.	7.6 µs	7.6 µs
		V:Bit Reg.	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	V: Data Reg.	V:Data Reg	7.6 µs	7.6 µs
		V:Bit Reg.	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	V: Bit Reg.	V:Data Reg.	7.6 µs	7.6 µs
		V:Bit Reg.	7.6 µs	7.6 µs
		K:Constant	4.8 µs	4.8 µs
		P:Indir. (Data)	30.2 µs	30.2 µs
		P:Indir. (Bit)	30.2 µs	30.2 µs
	P:Indir. (Data)	V:Data Reg.	29.9 µs	29.9 µs
		V:Bit Reg.	29.9 µs	29.9 µs
		K:Constant	27.4 µs	27.4 µs
		P:Indir. (Data)	51.0 µs	51.0 µs
		P:Indir. (Bit)	51.0 µs	51.0 µs
	P:Indir. (Bit)	V:Data Reg.	29.9 µs	29.9 µs
		V:Bit Reg.	29.9 µs	29.9 µs
		K:Constant	27.4 µs	27.4 µs
		P:Indir. (Data)	51.0 µs	51.0 µs
		P:Indir. (Bit)	51.0 µs	51.0 µs

Булевы команды - Бит из слова

Булевы команды – Бит из слова		DL06	
Команда	**Разрешенные типы данных**	**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
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
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
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
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
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
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
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
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

Команды немедленного действия

Команды немедленного действия		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
LDI	V	20.6 µs	1.1 µs
LDIF	1st #: Y 2nd #: K Constant	26.6 µs+0.9µs x N	1.4 µs
STRI	X	19.3 µs	19.3 µs
STRNI	X	19.4 µs	19.4 µs
ORI	X	19.1 µs	18.7 µs
ORNI	X	19.2 µs	18.9 µs
ANDI	X	18.7 µs	18.7 µs
ANDNI	X	18.8 µs	18.8 µs
OUTI	Y	25.5 µs	25.5 µs
OROUTI	Y	25.7 µs	25.7 µs
OUTIF	1st #: Y 2nd #: Y (N pt)	66.1 µs+0.9µs x N	1.4 µs
SETI	1st #: Y 2nd #: K Constant	23.1 µs, 22.8 µs+1.4µsxN	0.9 µs, 0.9 µs
RSTI	1st #: Y 2nd #: Y (N pt)	23.2 µs, 22.8 µs+1.4µsxN	0.9 µs, 0.9 µs

Таймеры, счетчики и регистры сдвига

Таймеры, счетчики и регистры сдвига			DL06	
Команда	**Разрешенные типы данных**		**Исполняемые**	**Неисполняемые**
	1st	***2nd***		
TMR	T	V Data Reg. V:Bit Reg K:Constant P:Indir. (Data) P:Indir. (Bit)	26.8 µs 26.8 µs 20.0 µs 45.6 µs 45.6 µs	7.3 µs 7.3 µs 4.8 µs 30.2 µs 30.2 µs
TMRF	T	V:Data Reg. V:Bit Reg K:Constant P:Indir. (Data) P:Indir. (Bit)	51.4 µs 51.4 µs 48.4 µs 75.9 µs 75.9 µs	7.3 µs 7.3 µs 4.6 µs 30.2 µs 30.2 µs
TMRA	T	V:Data Reg. V:Bit Reg K:Constant P:Indir. (Data) P:Indir. (Bit)	48.9 µs 48.9 µs 45.0 µs 75.9 µs 75.9 µs	7.3 µs 7.3 µs 4.6 µs 30.2 µs 30.2 µs

Таймеры, счетчики и сдвиг регистров			**DL06**	
Команда	**Разрешенные типы данных**		**Исполняемые**	**Неисполняемые**
	1st	***2nd***		
TMRAF	T	V Data Reg. V:Bit Reg K:Constant P:Indir. (Data) P:Indir. (Bit)	54.2 µs 54.2 µs 50.3 µs 81.2 µs 81.2 µs	7.3 µs 7.3 µs 4.6 µs 30.2 µs 30.2 µs
CNT	CT	V:Data Reg. V:Bit Reg K:Constant P:Indir. (Data) P:Indir. (Bit)	25.8 µs 25.8 µs 22.2 µs 53.5 µs 53.5 µs	7.3 µs 7.3 µs 4.6 µs 30.2 µs 30.2 µs
SGCNT	CT	V:Data Reg. V:Bit Reg K:Constant P:Indir. (Data) P:Indir. (Bit)	27.3 µs 27.3 µs 23.5 µs 54.9 µs 54.9 µs	7.3 µs 7.3 µs 4.6 µs 30.2 µs 30.2 µs
UDC	CT	V:Data Reg V:Bit Reg K:Constant P:Indir. (Data) P:Indir. (Bit)	39.8 µs 39.8 µs 35.4 µs 67.8 µs 67.8 µs	7.3 µs 7.3 µs 4.6 µs 30.2 µs 30.2 µs
SR	C (N points to shift)		17.8 µs + 0.9 µs x N	9.8 µs

Команды загрузки аккумулятора/стека и вывода данных

Команды загрузки аккумулятора/стека и вывода данных			DL06	
Команда	**Разрешенные типы данных**		**Исполняемые**	**Неисполняемые**
LD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit) V:Data Reg. V:Bit Reg.		11.8 µs 11.8µs 9.0 µs 33.9 µs 33.9 µs 12.2 µs 12.2 µs	1.0 µs 1.0 µs 1.0 µs 0.9 µs 0.9 µs 1.0 µs 1.0 µs
LDD	K:Constant P:Indir. (Data) P:Indir. (Bit)		9.0 µs 37.8 µs 37.8 µs	1.0 µs 0.9 µs 0.9 µs
LDF	***1st*** X, Y, C,S T, CT, SP	***2nd*** K:Constant	20.5 µs+0.9 µsxN	0.9 µs
LDA	O: (Octal constant for address)		10.4 µs	1.0 µs
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
LDSX	K: Constant		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
OUT	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)		9.3 µs 9.3 µs 35.2 µs 35.2 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs

Команды загрузки аккумулятора/стека и вывода данных			DL06	
Команда	**Разрешенные типы данных**		**Исполняемые**	**Неисполняемые**
OUTD	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)		10.2 µs 10.2 µs 35.8 µs 35.8 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs
OUTF	**1st** X, Y, C	**2nd** K:Constant	54 µs+1.0 µs xN	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		8.4 µs	1.0 µs

Логические команды

Логические команды (Аккумулятор)		**DL06**	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
AND	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	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)	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	21.6 µs + 0.9 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)	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)	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

Логические команды		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
ORF	***1st:*** X, Y, C,S T, CT, SP, GX, GY ***2nd:*** K:Constant	20.9 µs + 0.9 µs 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)	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)	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
XORF	***1st:*** X, Y, C,S T, CT, SP, GX, GY ***2nd:*** K:Constant	20.9 µs + 0.9 µs 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)	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)	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:*** X, Y, C,S T, CT, SP, GX, GY ***2nd:*** K:Constant	20.9 µs +1.0 µs x N	1.0 µs
CMPR	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	42.8 µs 42.8 µs 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)		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
ADD	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	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. (Daa) P:Indir. (Bit)	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)	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)	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)	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
DIV	V:Data Reg. V:Bit Reg K:Constant P:Indir. (Data) P:Indir. (Bit)	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
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
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
INCB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	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)	13.2 µs 13.2 µs 38.0 µs 38.0 µs	1.0 µs 1.0 µs 0.9 µs 0.9 µs

Математические команды		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
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
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
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
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
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
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
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
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
ADDF	***1st:*** X, Y, C,S T, CT, SP, GX, GY ***2nd:*** K:Constant	109.3 µs +0.9 µs x N	1.0 µs

Математические команды		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
SUBF	***1st:*** X, Y, C,S,T, CT, SP, GX, GY ***2nd:*** K:Constant	107.3 µs +0.9 µs x N	1.0 µs
MULF	***1st:*** X, Y, C,S, T, CT, SP, GX, GY ***2nd:*** K:Constant	352.5 µs +0.9 µs x N	1.0 µs
DIVF	***1st:*** X, Y, C,S, T, CT, SP, GX, GY ***2nd:*** K:Constant	477.3 µs +0.8 µs 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
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		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
PD	X, Y, C	14.4 µs	14.4 µs
STRPD	X, Y, C, S, T, CT	5.4 µs	5.4 µs
STRND	X, Y, C, S, T, CT	7.3 µs	7.3 µs
ORPD	X, Y, C, S, T, CT	6.8 µs	5.2 µs
ORND	X, Y, C, S, T, CT	7.1 µs	4.9 µs
ANDPD	X, Y, C, S, T, CT	6.8 µs	5.2 µs
ANDND	X, Y, C, S, T, CT	7.1 µs	4.9 µs

Битовые команды

Битовые команды		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
SUM	None	6.7 µs	1.0 µs
SHFR	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	12.1 µs +0.1 x N 8.4 µs +0.1 x N	0.9 µs
SHFL	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	12.1 µs +0.1 x N 8.4 µs +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	33.9 µs	0.9 µs
DECO	None	5.7 µs	1.0 µs

Команды преобразования чисел

Команды преобразования чисел		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
BIN	None	100.2 µs	0.9 µs
BCD	None	95.2 µs	0.9 µs
INV	None	2.5 µs	1.0 µs
BCDPL	None	75.6 µs	1.0 µs
ATH	V	25.4 µs	1.0 µs
HTA	V	25.4 µs	1.0 µs
GRAY	None	110.8 µs	1.0 µs
SFLDGT	None	23.1 µs	1.0 µs
BTOR	None	18.6 µs	1.0 µs
RTOB	None	8.6 µs	1.0 µs
RADR	None	51.4 µs	1.0 µs
DEGR	None	81.5 µs	1.0 µs

Табличные команды

Табличные команды		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
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	60.2 µs+9.5 x N	0.9 µs
MOVMC	Move V:data Reg to E^2 Move V:Bit Reg to E^2 Move from $E^{2\text{ to V:Data Reg}}$ Move from $E^{2\text{ to V:Bit Reg}}$ N= #of words	35 µs + 10.4 µs x N	0.9 µs
LDLBL	K	6.4 µs	1.3 µs
FILL	V: Data Reg V:Bit Reg K:Constant P:Indir. (Data) P:Indir. (bit)	29.4 µs + 8.0 µs x N 26.2 µs + 8.0 µs x N 55.1 µs + 8.0 µs 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

Табличные команды (продолжение)		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
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
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
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

Команды управления процессором

Команды управления процессором		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
NOP	None	1.1 µs	1.1 µs
END	None	24.0 µs	24.0 µs
STOP	None	10.0 µs	1.1 µs
RSTWT	None	5.9 µs	2.2 µs

Команды управления программой

Команды управления программой		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
GOTO	K	5.1 µs	4.8 µs
LBL	K	5.7 µs	0.0 µs
FOR	V, K	125.9 µs	14.5 µs
NEXT	None	64.4 µs	64.4 µs
GTS	K	27.5 µs	14.8 µs
SBR	K	1.5 µs	1.5 µs
RTC	None	25.7 µs	12.1 µs
RT	None	21.2 µs	21.2 µs
MLS	K	(1–7) 35.2 µs	35.2 µs
MLR	K	(0–7) 30.9 µs	30.9 µs

Команды прерывания

Команды прерывания		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
ENI	None	24.2 µs	2.7 µs
DISI	None	9.4 µs	2.3 µs
INT	O(0,1)	7.5 µs	-
IRTC	None	0.9 µs	1.3 µs
IRT	None	6.6 µs	-

Сетевые команды

Сетевые команды		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
RX	X, Y, C, T, CT, SP, S, $	852.0 µs	4.4 µs
	V:Data Reg.	852.0 µs	4.4 µs
	V:Bit Reg.	852.0 µs	4.4 µs
	P:Indir. (Data)	868.2 µs	4.2 µs
	P:Indir. (Bit)	868.2 µs	4.2 µs
WX	X, Y, C, T, CT, SP, S, $	1614.0 µs	4.4 µs
	V:Data Reg.	1614.0 µs	4.4 µs
	V:Bit Reg.	1614.0 µs	4.4 µs
	P:Indir. (Data)	1630.0 µs	4.2 µs
	P:Indir. (Bit)	1630.0 µs	4.2 µs

Команды интеллектуального ввода/вывода

Команды интеллектуального ввода/вывода по сети		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
RD	V:Data Reg. V:Bit Reg	385.7 µs 385.7 µs	1.2 µs 1.2 µs
WT	V:Data Reg. V:Bit Reg	385.6 µs 385.6 µs	1.2 µs 1.2 µs

Команды вывода сообщений

Команды вывода сообщений		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
FAULT	V:Data Reg. V:Bit Reg. K:Constant	65.0 µs 65.0 µs 204.7 µs	4.4 µs 4.4 µs 4.4 µs
DLBL	K	–	–
NCON	K	–	–
ACON	A	–	–
PRINT	ASCII	631.0 µs	3.6 µs

Команды RLL *plus*

Команды RLL *plus*		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
ISG	S	44.0 µs	41.1 µs
SG	S	44.0 µs	41.1 µs
JMP	S	76.0 µs	9.3 µs
NJMP	S	77.4 µs	9.3 µs
CV	S	42.1 µs	27.5 µs
CVJMP	S	89.5 µs	17.6 µs
BCALL	C	22.1 µs	22.6 µs
BLK	C	17.1 µs	14.6 µs
BEND	None	8.7 µs	0.0 µs

Команды барабанного командоаппарата

Команды барабанного командоаппарата		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
DRUM	CT	840.0 µs	339.6 µs
EDRUM	CT	753.2 µs	357.0 µs
MDRMD	CT	411.3 µs	216.4 µs
MDRMW	CT	378.6 µs	147.0 µs

Команды даты / времени

Команды даты / времени		DL06	
Команда	**Разрешенные типы данных**	**Исполняемые**	**Неисполняемые**
DATE	V:Data Reg. V:Bit Reg.	24.0 µs	1.2 µs
TIME	V:Data Reg. V:Bit Reg.	50.8 µs	1.2 µs

Команды MODBUS

Команды MODBUS		DL06	
Команда	**Разрешенные типы данных**	**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		DL06	
Команда	**Разрешенные типы данных**	**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

14. Приложение D. Специальные реле

В данной главе…

Специальные реле PLC DL06

"Специальные реле" — это контакты, которые устанавлены операционной системой процессора (ЦПУ) для сообщения, что произошло определенное системное событие. Эти контакты можно использовать в своей релейной программе. Зная, какой специальный контакт нужно использовать в определенной ситуации, можно сэкономить время программирования. Как только операционная система установит или сбросит специальные контакты реле, релейная программа может использовать их как входы в релейную логику.

Реле запуска и реального времени

SP0	Первое сканирование	Включается при первом сканировании после включения питания или при переходе из программного режима в рабочий. Реле отключается на втором сканировании. Полезно для функций, выполняемых только при пуске программы.
SP1	Всегда ВКЛЮЧЕНО	Обеспечивает контакт (ON), гарантирующий выполнение команды при каждом сканировании.
SP2	Всегда ВЫКЛЮЧЕНО	Обеспечивает всегда выключенный контакт(OFF)
SP3	1 мин. таймер	Включено 30 сек и отключено в течение 30 сек.
SP4	1 сек. таймер	Включено 0.5 сек и отключено в течение 0.5 сек.
SP5	100 мс таймер	Включено 50 мс и отключено в течение 50 мс.
SP6	50 мс таймер	Включено 25 мс и отключено в течение 25 мс.
SP7	Чередующееся сканирование	Включается через одно сканирование.

Реле состояния процессора

SP11	Принудительный рабочий режим	Включено всегда, когда переключатель ЦПУ находится в положении RUN (Работа).
SP12	Терминальный рабочий режим	Включено, когда переключатель находится в положении TERM, а ЦПУ находится в режиме RUN.
SP13	Режим работы тестирования	Включено, когда Процессор находится в режиме тестирования (Test RUN).
SP15	Режим останова тестирования	Включено, когда Процессор находится в режиме останова тестирования (Test STOP).
SP16	Терминальный программный режим	Включено, когда переключатель находится в положении TERM, а ЦПУ находится в режиме PGM.
SP17	Реле вынужденного останова	Включено всегда, когда переключатель ЦПУ находится в положении STOP.
SP20	Реле вынужденного режима останова	Включено, когда выполнена команда STOP.
SP22	Разрешение прерывания	Включено, когда прерывание разрешено командой ENI.

Реле контроля работы системы

SP36	Реле установки подмены	Включено, когда используется функция форсирования переменных (Подмена реальных значений X/Y)
SP37	Ошибка времени цикла	Включено, когда реальное время цикла превышает заданное.
SP40	Критическая ошибка	Включено, когда возникает критическая ошибка, например, потеря связи с точками Ввода/Вывода
SP41	Предупреждение	Включено, когда возникает некритическая ошибка,
SP42	Диагностическая ошибка	Включено, когда возникает диагностическая или системная ошибка
SP43	Низкое напряжение батареи	Включено, когда напряжение батареи ЦПУ мало.
SP44	Ошибка программной памяти	Включено, когда возникает ошибка памяти, например, ошибка четности памяти.
SP45	Ошибка Ввода/Вывода	Включено, когда возникает ошибка Ввода/Вывода. Например, перегорел предохранитель.
SP46	Ошибка Связи	Включено, когда возникает ошибка связи на любом последовательном порту ЦПУ.
SP50	Неправильная команда	Включено при выполнении неправильной команды. (Fault Instruction)
SP51	Истечение времени сторожевого таймера	Включено, когда время сторожевого таймера Процессора истекло. (Watch Dog Timeout)
SP52	Грамматическая ошибка	Включено, когда обнаружена грамматическая ошибка при работе Процессора или при проверке синтаксиса. В V7755 хранится точный код ошибки
SP53	Логическая ошибка	Включено, когда Процессор не может разрешить логику.
SP54	Ошибка связи	Включено, когда команды RX,WX выполняются с неправильными параметрами.
SP56	Ошибка табличной команды	Включено, когда табличная команда использует указатель, а значение указателя вышло за пределы таблицы.

Состояние аккумулятора

SP60	Значение меньше, чем	Включено, когда значение в Аккумуляторе меньше, чем значение в команде.
SP61	Значение равно	Включено, когда значение в Аккумуляторе равно значению в команде
SP62	Больше, чем	Включеноо, когда значение в Аккумуляторе больше, чем значение 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	Включено всякий раз, когда значение в Аккумуляторе становится отрицательным.
SP71	Ошибка назначения указателя	Включено, когда указателем (P) определена неправильная ячейка V-памяти.
SP72	Число с плавающей запятой	Включено всякий раз, когда значение в Аккумуляторе действительное число с плавающей запятой.
SP73	Переполнение	Включено, когда происходит переполнение Аккумулятора, или сложение/вычитание со знаком приводит к неправильному биту знака.
SP74	Антипереполнение	Включено всякий раз, когда работа с плавающей точкой приводит к ошибке обнуления.
SP75	Ошибка в данных	Включено, если число должно быть в формате BCD, а оно не в формате BCD.
SP76	Загрузка нуля	Включено, когда команда загружает в Аккумулятор значение нуль.

Состояние входа высокоскоростного ввода/ вывода

SP100	состояние X0	Включено, когда X0 включен
SP101	состояние X1	Включено, когда X1 включен

Реле выходных импульсов высокоскоростного ввода/вывода

SP104	Профиль завершен	Включено, когда профиль выходных импульсов завершен (Режим 30).

Реле контроля связи

SP116	Порт 2 Процессора занят	Включено, когда порт 2 назначен мастером и посылает данные
SP117	Порт 2. Ошибка Связи	Включено, когда порт 2 назначен мастером и есть ошибка связи.

Реле контроля связи дополнительного слота

SP120	Слот 1 занят	Порт 2 H0-ECOM/D0-DCM
SP121	Слот 1 ошибка	Порт 2 H0-ECOM/D0-DCM
SP122	Слот 2 занят	Порт 2 H0-ECOM/D0-DCM
SP123	Слот 2 ошибка	Порт 2 H0-ECOM/D0-DCM
SP124	Слот 3 занят	Порт 2 H0-ECOM/D0-DCM
SP125	Слот 3 ошибка	Порт 2 H0-ECOM/D0-DCM
SP126	Слот 4 занят	Порт 2 H0-ECOM/D0-DCM
SP127	Слот 4 ошибка	Порт 2 H0-ECOM/D0-DCM

Специальные реле дополнительного слота

SP140-237	Слот 1	SP реле для дополнительного модуля
SP240-337	Слот 2	SP реле для дополнительного модуля
SP340-437	Слот 3	SP реле для дополнительного модуля
SP430-537	Слот 4	SP реле для дополнительного модуля

Реле равенства высокоскоростного ввода/вывода в режиме 10 счетчика 1

SP540	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3631/3630
SP541	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3633/3632
SP542	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3635/3634
SP543	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3637/3636
SP544	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3641/3640
SP545	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3643/3642
SP546	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3645/3644
SP547	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3647/3646
SP550	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3651/3650
SP551	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3653/3652
SP552	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3655/3654
SP553	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3657/3656
SP554	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3661/3660
SP555	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3663/3662
SP556	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3665/3664
SP557	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3667/3666.
SP560	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3671/3670.
SP561	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3673/3672.
SP562	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3675/3674.
SP563	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3677/3676.
SP564	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3771/3770.
SP565	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3703/3702.
SP566	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3705/3704.
SP567	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3707/3706.

Реле равенства высокоскоростного ввода/вывода в режиме 10 счетчика 2

SP570	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3711/3710
SP571	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3713/3712
SP572	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3715/3714
SP573	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3717/3716
SP574	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3721/3720
SP575	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3723/3722
SP576	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3725/3724
SP577	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3727/3726
SP600	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3731/3730
SP601	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3733/3732
SP602	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3735/3734
SP603	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3737/3736
SP604	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3741/3740
SP605	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3743/3742
SP606	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3745/3744
SP607	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3747/3746
SP610	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3751/3750
SP611	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3753/3752
SP612	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3755/3754
SP613	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3757/3756
SP614	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3761/3760
SP615	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3763/3762
SP616	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3765/3764
SP617	Текущее = заданному значению	Включено, когда текущее значение счетчика равно значению в V3767/3766

Приложение E

15. Приложение E. Вес изделий

В данной главе…

Таблица веса изделий

Микроконтроллер	Вес
D0–06AR	0,807 кг
D0–06DR	0,798 кг
D0–06DR-D	0,78 кг
D0–06AA	0,807 кг
D0–06DA	0,798 кг
D0–06DD1	0,762 кг
D0–06DD1-D	0,743 кг
D0–06DD2-D	0,743 кг
D0–06DD2	0,798 кг
D0–06LCD	0,0544 кг

Приложение F

16. Приложение F. Память контроллера

В данной главе…

Память контроллера DL06

Когда программируется контроллер, пользователю важно понимать различие между типами памяти контроллера. В контроллере DL06 используется два типа памяти: RAM и EEPROM. Эта память может быть сконфигурирована пользователем либо как сохраняемая, либо как несохраняемая.

Сохраняемая память – это память, значения параметров в которой сохраняются после выключения источника питания или при переходе из режима RUN в режим PROGRAM. Несохраняемая память – это память, в которой значения параметров не сохранятся при выключении питания или при переходе из RUN в PROGRAM. Диапазон сохраняемых параметров может быть задан с помощью ручного программатора, используя AUX57, или пакета программирования DirectSOFT32(PLCSetup).

Данные могут быть записаны в RAM память и читаться из нее бесконечное число раз. Для сохранения содержимого памяти используется внутренний источник питания(=5В), если у контроллера есть внешнее питание. Если питание контроллера выключено, содержимое RAM памяти сохраняется при помощи «Супер-Конденсатора». Если «Супер-Конденсатор» разрядится, то данные в RAM памяти будут потеряны. Время поддержания содержимого памяти «Супер-Конденсатором» - максимум 3 недели и минимум 4,5 дня (при 60^{o}C).

Содержимое EEPROM памяти может быть прочитано бесконечное число раз, но данные могут быть записаны ограниченное число раз (типично – 100000 раз). Для сохранения данных в EEPROM памяти не требуется источник питания. Содержимое памяти будет храниться бесконечно.

Пользовательская V-память хранится в энергозависимой RAM памяти и энергонезависимой EEPROM памяти. RAM память использует следующие адреса: V400-V677, V1200-V7377 и V10000-V17777. EEPROM память использует следующие адреса: V7400-V7577 и V700-V777, V7600-V7777 и V36000- V37777.

Значения параметров, которые должны храниться длительное время, когда контроллер выключен, необходимо сохранить в EEPROM памяти.

Значения параметров, которые непрерывно меняются или инициализированные программой, нужно хранить в RAM памяти.

Приложение G

17. Приложение G. Таблица ASCII

В данной главе…

СООТВЕТСТВИЕ ЦИФР В ДЕСЯТИЧНОЙ СИСТЕМЕ СЧИСЛЕНИЯ В ШЕСТНАДЦАТИРИЧНОЙ И ASCII											
DEC	HEX	ASCII	DEC	HEX	ASCII	DEC	HEX	ASCII	DEC	HEX	ASCII
0	00	NUL	32	20	space	64	40	@	96	60	`
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	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	l
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

18. Приложение H. Директивы Европейского сообщества (CE)

В данной главе…

Директивы Европейского союза (ЕС)

ПРИМЕЧАНИЕ. Информация, содержащаяся в данном 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 делать бизнес 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 Европейского союза. Основная цель единого стандарта состоит в том, чтобы облегчить торговлю и транспортировку товаров, обеспечить безопасную работу, сохранить среду обитания. Директивы, которые являются результатом объединения стандартов, в настоящее время представляют собой официальные требования для всех, кто занимается бизнесом в Европе. Изделия, удовлетворяющие этим Директивам, должны иметь марку CE, означающую соответствие

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 без чрезмерного электромагнитного воздействия на что-либо в этой среде.
- **Директива по безопасности машинного оборудования** — эта Директива включает аспекты безопасности аппаратуры, установок и др. Затрагивается несколько областей, включая стандарты на тестирование, относящиеся как к защите от электрических помех, так и к генерации этих помех.
- **Директива по низкому напряжению** — эта Директива также относится к безопасности и касается электрического 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, повторному использованию и утилизации батарей.

Соответствие

Определенные стандарты в рамках каждой Директивы уже требуют обязательного соответствия. Директива по электромагнитной совместимости, стала обязательной с 1 января 1996 г. Директива по низкому напряжению стала обязательной с 1 января 1997 г.

В конечном счете, мы все ответственны за различные аспекты проблемы. Как изготовители, мы должны проводить испытание наших изделий и документировать любые результаты испытания и/или методов сборки, которые необходимы для соблюдения Директив. Как механики, вы ответственны за такую установку изделий, которая сохранит соответствие. Вы также ответственны за испытание любых комбинаций изделий, которые при совместном использовании могут соответствовать Директивам (или не соответствовать).

Конечный пользователь изделий должен соблюдать любые Директивы, относящиеся к сопровождению, утилизации и т. д. оборудования или различных его компонентов. Хотя мы прилагаем большие усилия в этом направлении, 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 машинное оборудование (и все остальное) соответствовало данным Директивам и чтобы это соответствие поддерживалось в процессе эксплуатации.

С 1 января 1999 г. системы DL05, DL06, DL205, DL305 и DL405 производства фирм Koyo Electronic Industries или FACTS Engineering при надлежащей установке и использовании соответствуют требованиям Директив по электромагнитной совместимости, низкому напряжению и безопасности машинного оборудования в составе следующих стандартов.

- Стандарты (Директивы) по электромагнитной совместимости, относящиеся к ПЛК:
  EN50081-1 Общий стандарт по излучению для бытовой техники, торгового оборудования и для легкой промышленности
  EN50081-2 Общий стандарт по излучению для промышленности
  EN50082-1 Общий стандарт по защищенности для бытовой техники, торгового оборудования и для легкой промышленности
  EN50082-2 Общий стандарт по защищенности для бытовой техники, торгового оборудования и для легкой промышленности
- Стандарт (Директивы) по низкому напряжению, применимый к ПЛК:
  EN61010-1 Требования безопасности для электрического оборудования для измерения, контроля и лабораторного применения
- Специальный стандарт для ПЛК:
  EN61131-2 Программируемые контроллеры, требования к оборудованию и испытаниям. Данный стандарт заменяет указанные выше стандарты по защищенности и безопасности. Но общие стандарты по излучению должны пока использоваться вместе со следующими стандартами:
  EN61000-3-2 — Колебания (Harmonics)
  EN61000-3-2 — Флуктуации
- Защита от поражения электрическим током (ESD)
  Персоналу необходимо соблюдать меры безопасности для защиты от поражения электрическим током. К таким мерам относятся, например, использование шины заземления, отключение внешнего питания при проведениии работ по обслуживанию оборудования и т.п.
- Защита от электромагнитных помех (RFI)
  Контроллеры DL06 по излучаемым электромагнитным помехам относятся к классу A.

Общая безопасность

- Внешние переключатели, автоматический выключатель или внешний предохранитель требуются для этого оборудования.
- Переключатель или автоматический выключатель должен монтироваться около оборудования контроллера.

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 указываем специальное руководство, которое вы можете заказать:

- DA—EU-M: Руководство ЕС по установке, которое включает конкретные требования к установке, удовлетворяющие Директивам ЕС. Закажите это руководство, чтобы получить самую последнюю информацию.

Другие источники информации

Хотя Директива по электромагнитной совместимости стала объектом наибольшего внимания, другие базовые Директивы, например, Директивы по машинному оборудованию и низкому напряжению, они также накладывают ограничения на построение системы 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.
- EN60204-1, 1992 г: Общие электрические требования к машинному оборудованию, включая анализ требований 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 частным образом или обратиться к нам по электронной почте: support@plcsystem.ru или к нашему WEB-сайту: www.plcsystems.ru.

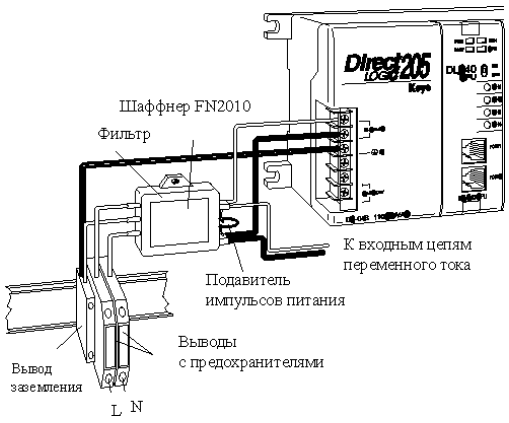
Основные руководящие указания по электромагнитной совместимости (ЭМС) оборудования

Шкафы

Самый простой способ выполнения требований безопасности Директив по машинному оборудованию и низкому напряжению состоит в том, чтобы поместить всю аппаратуру управления в стандартный запираемый стальной шкаф заводского производства. Этот способ обычно имеет дополнительные преимущества, поскольку помогает обеспечить выполнение требований Директивы по электромагнитной совместимости. Хотя излучение радиочастот аппаратурой ПЛК, измеренное на открытом воздухе, значительно ниже пределов, установленных Директивами по ЭМС, определенные конфигурации могут увеличивать уровень излучения. Стандартный стальной шкаф заводского производства без каких-либо специальных мер по экранированию может обеспечить поглощение до 10 дБ. Отверстия в шкафу для прокладки кабелей или для монтажа панелей оператора могут увеличить излучение.

Фильтры в сети переменного тока

Чтобы обеспечить соответствие Директивам по электромагнитной совместимости в части кондуктивного излучения радиоволн, блоки питания переменного тока каркасов DL05, DL06, DL205, DL305 должны иметь дополнительную фильтрацию сетевого питания. Вся аппаратура ПЛК испытывается с фильтрами производства Schaffner, которые уменьшают уровни излучения при надлежащем заземлении. Следует выбирать фильтр с токовым номиналом, подходящим для всех блоков питания ПЛК и входных модулей переменного тока. Для систем DL05/DL06/DL205 мы предлагаем фильтр FN2010, для систем DL305 — FN2080. Системы DL405 не требует дополнительной фильтрации.



ПРИМЕЧАНИЕ. Не все фильтры в питающей сети могут уменьшить излучение при неисправностях до малых уровней. В некоторых случаях фильтры могут даже увеличить кондуктивное излучение, если они подобраны не правильно.

Подавление импульсов питания и установка предохранителей

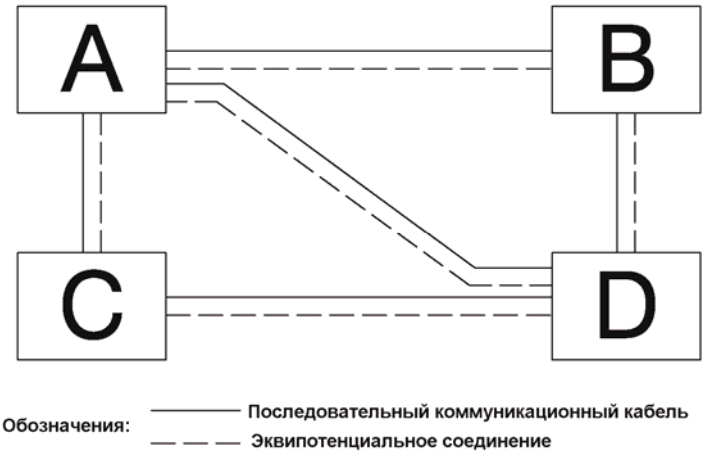
Чтобы обеспечить соответствие требованиям электротехнических стандартов EN61010-1 и EN60204-1 Директив по Машинному Оборудованию и Низкому Напряжению в части пожарной безопасности посредством ограничения мощности в схемах сетевого питания «без ограничений» с реверсивными силовыми проводами, необходимо устанавливать предохранители на входах питания как переменного, так и постоянного тока. Вы также должны установить подавитель бросков напряжения между точками подсоединения входного питания ПЛК. Выбирайте подавитель типа варистор на окислах металла 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

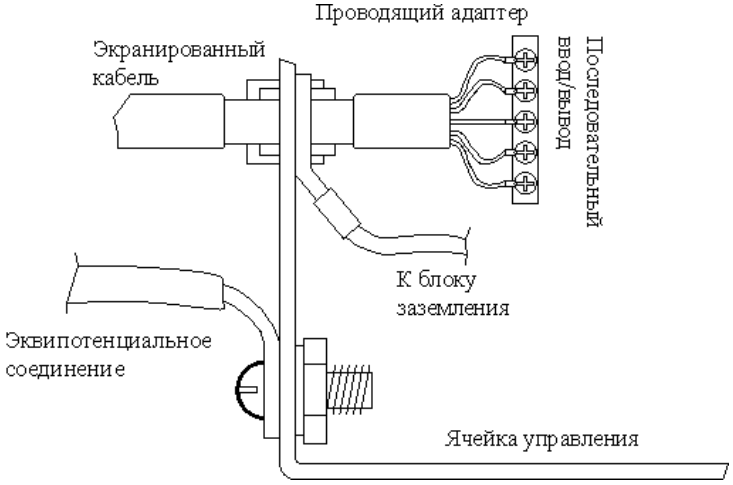
В каждом шкафу должны быть предусмотрены мощные клеммные блоки для звездообразного подсоединения всех шин заземления, схем защитного заземления, проводов заземления фильтров сетевого питания, а также для подключения заземления механических узлов. Это необходимо для соответствия требованиям по безопасности и электромагнитной совместимости, частных стандартов, а также стандарта МЭК 1000-5-2. Директива по машинному оборудованию требует, чтобы общие полюса входных модулей ПЛК и общий провод питания нагрузок, управляемых выходными модулями ПЛК, также подсоединялись к клемме защитного заземления.

Эквипотенциальное заземление



Для оборудования, содержащего современные электронные схемы, должен быть предусмотрен адекватный узел заземления. Применение для электронных систем отдельных заземляющих электродов запрещено в некоторых странах. Стандарт МЭК 1000-5-2 предусматривает эквипотенциальное соединение всех цепей заземления. Но особое внимание следует отдать аппаратуре управления, которая содержит устройства Ввода/Вывода, блоки удаленного Ввода/Вывода или блоки, имеющие межсистемные связи с основным шкафом системы ПЛК. Должен быть предусмотрен эквипотенциальный связывающий провод параллельно всем кабелям последовательных соединений и параллельно кабелям к любым отдельным объектам предприятия, которые имеют устройства Ввода/Вывода, подсоединенные к ПЛК. На схеме выше показан пример из четырех физических объектов, соединенных коммуникационным кабелем.

Связи и экранированные кабели



Для кабельной разводки аналоговых сигналов и кабельной разводки коммуникаций вне шкафа ПЛК рекомендуется экранированный кабель с витой парой сечением минимум 0,2 мм^2 с экраном из фольги и оплетки.

Общепринятая практика, действующая до самого последнего времени, состояла в том, что экран кабеля заземлялся только с одного конца, чтобы минимизировать риск помех, вызванных контурным током заземления между отдельными приборами. Метод заземления только с одного конца возник как попытка уменьшить радиопомехи в аудио системах, он не применяется более для сложного промышленного оборудования. Экранированные кабели являются также эффективным генератором радиопомех, излучаемых системами ПЛК, которые могут отрицательно влиять на работу сетей.

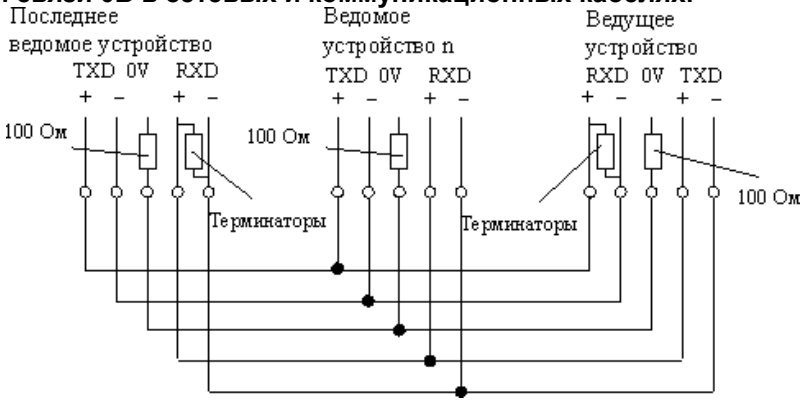
Рекомендуется использовать экранированные кабели как электростатические «трубы» между прибором и системами, а также проложить вдоль всех экранированных кабелей провода большого диаметра для эквипотенциального соединения. В случае, когда экранированный кабель проходит через металлическую стенку шкафа или машины, МЭК 1000-5-2 рекомендует подсоединять экран по всему его периметру к стенке, предпочтительно используя проводящий адаптер. Не рекомендуется соединять экран через гибкий провод к болту заземления. Экраны должны подсоединяться к каждой стенке шкафа или к каждому кожуху машины, через которые они проходят.

Кабели для аналоговых сигналов и для RS232

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, выполненному на клеммах модуля. Такой метод применим также к кабелям RS232.

Кабели для многоточечных соединений

Кабели с двойной витой парой для RS422 и с одной витой парой для RS485 также требуют провода 0 В, вместо который раньше часто использовался экран кабеля. В настоящее время рекомендуется использовать кабель с тройной витой парой для линий связи RS422 и кабель с двойной витой парой для линий связи RS485. Это связано с тем, что дополнительная пара может использоваться как 0В межсистемная линия связи. С источниками питания постоянного тока в контурах, заземленных в обоих системах, контуры заземления создаются таким же способом через 0В межсистемную линию связи. Инструкции по установке требуют создания таких контуров заземления, которые имеют малое сопротивление за счет применения проводов большого диаметра при эквипотенциальном соединении. **Для не европейских систем, использующих заземление только в одном конце и имеющих далеко не идеальные характеристики заземления, мы рекомендуем установить дополнительные резисторы по 100 Ом на каждой линии связи 0В в сетевых и коммуникационных кабелях.**



Экранированные кабели внутри шкафов

При прокладке кабелей между различными элементами внутри шкафа, в котором находится чувствительная электронная аппаратура разных изготовителей, следует помнить, что эти кабели могут быть источником излучения радиочастот. Существуют способы минимизации этого риска. Стандартные кабели для передачи данных, соединяющие ПЛК и интерфейсы оператора, должны прокладываться вдали от другого оборудования и связывающей его кабельной разводки. Можно использовать специальные кабели, в которых экран кабеля подсоединяется с обоих концов к заземлению шкафа таким же способом, как и внешние кабели.

Отключение сети связи

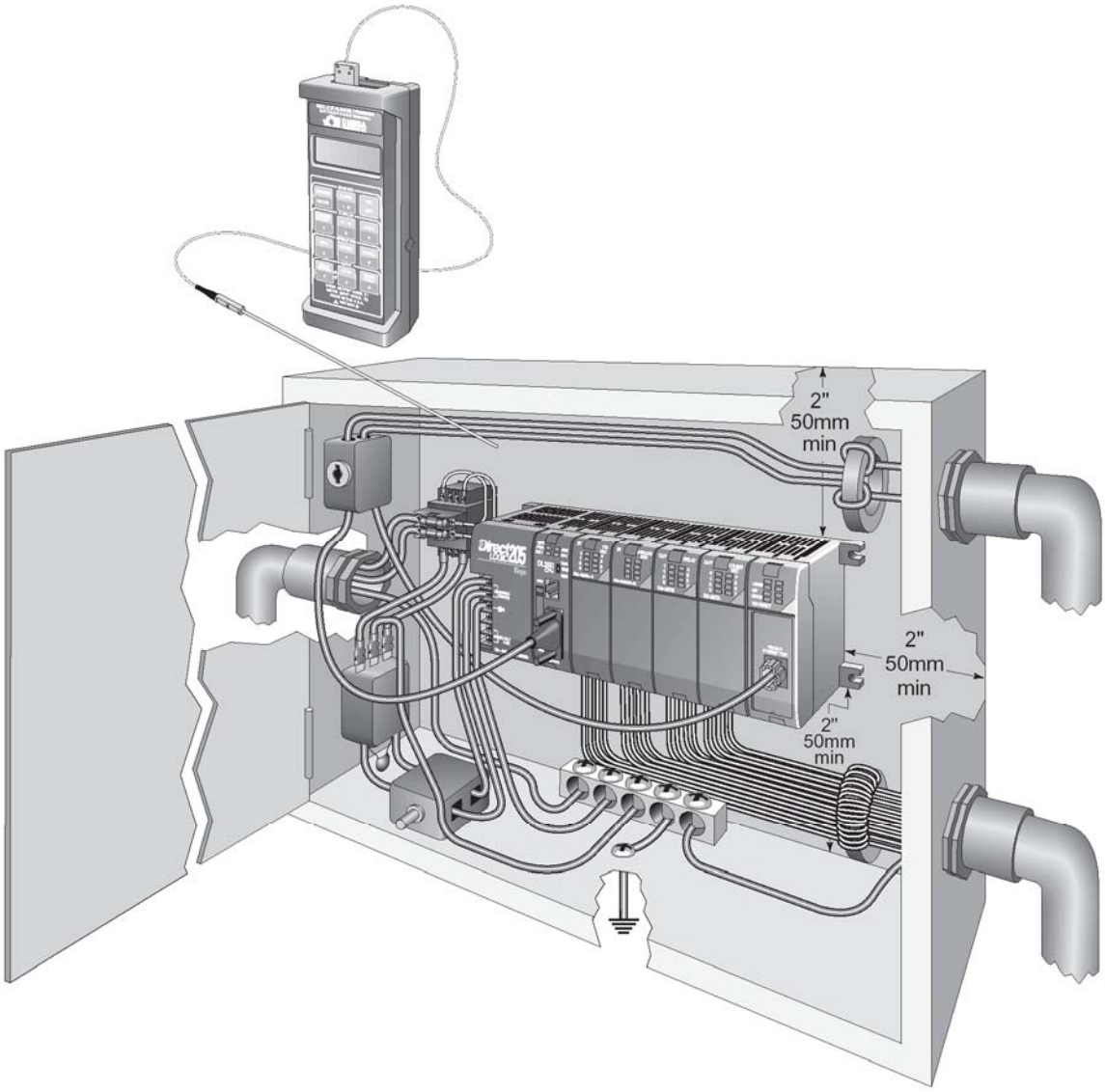
В целях обеспечения безопасности в Директивах по машинному оборудованию предусмотрено требование установки выключателя, который отключает на период обслуживания входные сигналы сети, при этом удаленные команды не могут повлиять на работу оборудования. FA-ISONET не имеет выключателя. Используйте блокировку 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://eur-op.eu.int/

Модели с питанием постоянным током

Из-за слегка более высокого излучения от моделей микроконтроллера DL06 с питанием от постоянного тока, и различных характеристик излучения для различных напряжений питания, необходимо соблюдать рекомендации:

- Контроллер должен быть размещен внутри металлического шкафа с минимальным количеством отверстий.
- Кабели Ввода/вывода и связи, приходящие в шкаф должены быть размещены внутри металлических труб или кабельных каналов.



Пункты специфические для DL06

- Номинальное значение изоляции между всеми цепями в этом изделии установлено только как основная изоляция, т.е. соответствующая условиям одиночного отказа.
- не имеется никакой изоляции между контроллером и аналоговыми входами F0-04-AD-1.
- Ответственностью проектировщика системы является подключение всех управляющих и питающих цепей к одному контуру заземления.
- Оборудование должно быть установлено с соблюдением местных требований руководящих документов, требований руководства по монтажу DA-EU-M и стандартов по установке IEC 1000-5-1, IEC 1000-5-2 и IEC 1131-4.
- Необходимым условием является размещение всего оборудования в стальном защитном корпусе шкафу), который ограничивает доступ операторам при помощи замков и прерывателей питания.
- Необходимо отметить, что директива по безопасности машин EN60204-1 требует, чтобы все цепи питания оборудования должны быть подключены через изолирующие трансформаторы или изолированные блоки питания, и чтобы все цепи управления переменного и постоянного тока были бы заземлены с одной стороны.
- Обе линии питания контроллера должны быть защищены отдельными плавкими предохранителями на 3А типа Т и подавителем импульсов питания.
- Если оборудование используется не указанным изготовителем способом, то защита ородования может быть повреждена.