DL405

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

ОГЛАВЛЕНИЕ

Глава 1. Начальные сведения

Глава 2. Установка, монтаж и спецификации

Глава 3. Спецификации и работа процессора

Глава 4. Проектирование и конфигурирование системы

Глава 5. Стандартные команды RLL

Глава 6. Программирование с помощью барабанного командоаппарата (только для DL450)

Глава 7. Стадийное программирование RLL^{PLUS}

Глава 8. Работа контуров ПИД-регулирования (только DL450)

Глапа 9. Обслуживание и поиск неисправностей

Приложение A. Вспомогательные функции

Приложение D. Специальные реле

Приложение E. Веса изделий DL405

Начальные сведения

1

В этой главе...

— Введение

— Методы программирования

— Система нумерации частей DirectLOGIC™

— Как быстро начать проверку и программирование ПЛК

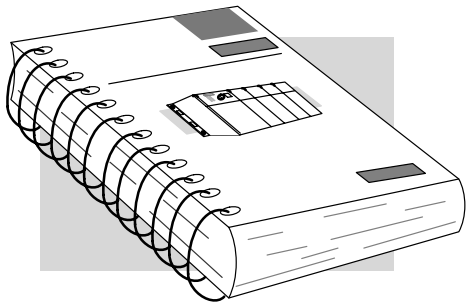
— Шаги по созданию хорошей системы

— Часто задаваемые вопросы

Введение

Цели данного руководства

Благодарим вас за покупку нашей серии продуктов DL405. Данное руководство покажет вам, как устанавливать, программировать и обслуживать оборудование. Оно также поможет вам понять, как оно взаимодействует с другими устройствами в системе управления.



Данное Руководство содержит важную информацию для лиц, которые устанавливают программируемый логический контроллер (ПЛК) DL405 и его компоненты, а также для программистов ПЛК. Если вы знакомы с ПЛК, то наши руководства обеспечат вас всей необходимой информацией, чтобы запустить и поддерживать вашу систему, а также эксплуатировать ее.

Где начинать чтение

Если вы уже знакомы с ПЛК, пожалуйста, прочитайте главу 2 «Установка, монтаж и спецификации», при необходимости читайте другие главы. Держите данное Руководство под рукой для справок при появлении вопросов. Если вы — новый покупатель DL405, то мы предлагаем вам прочитать данное Руководство полностью, чтобы понять широкое многообразие возможностей серии продуктов DL405. Мы надеемся, что вы будете приятно удивлены, как много вы можете выполнить с продуктами DirectTM Logic

Дополнительные руководства

Если вы приобрели интерфейсы оператора или DirectSOFTTM, то в дополнение к данному Руководству вам необходимы руководства с описанием этих продуктов.

Техническая поддержка

Мы сознаем, что, несмотря на наши усилия, информация может быть расположена таким образом, что вы не можете найти то, что вам необходимо. Первым делом проверьте следующие возможности помощи в определении местонахождения информации:

- Оглавление — перечисление содержания по главам и разделам, в начале данного Руководства
- Быстрые указания по содержанию — краткое перечисление содержания главы на следующей странице
- Приложения — справочный материал по ключевым темам, ближе к концу данного Руководства

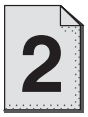
Вы также можете использовать наши оперативные возможности для получения информации по поддержке новых продуктов:

- Интернет — адрес нашего WEB-сайта: www.plcsystems.ru
- Электронная почта — support@plcsystems.ru

Главы Основное содержание данного Руководства расположено в следующих девяти главах

Начальные сведения

Вводит различные компоненты системы программируемого логического контроллера DL405. Включает также подсказки к начальным сведениям, а также подсказки, как создать хорошую систему.



Установка, монтаж и спецификации

Объясняет, как подготовиться к монтажу системы и дает указания по безопасности, чтобы помочь в защите вашего персонала и техники. Включает схемы системы и схемы соединений ввода/вывода, а также спецификации модулей ввода/вывода.

Спецификации и работа процессора

Перечисляет спецификации операционной системы и коммуникационных портов процессоров для DL430, DL440 и DL450. Для каждого типа процессоров приводятся операции программируемого контроллера по сканированию, временам выполнения задач, типам данных и картам распределения памяти.

Проектирование и конфигурирование системы

Объясняет различные способы расширения ввода/вывода и построения коммуникационных сетей для вашей системы DL405 ПЛК. Описаны расчет потребляемой мощности для модулей ввода/вывода, удаленный ввод/вывод и сети MODBUS®.

Стандартные команды RLL

Описывает, как работает каждая из 200+ стандартных команд. Включены примеры программ как для **Direct**SOFT, так и для Ручного программатора. Отмечается, что команды барабанного коммандоаппарата и стадийные команды рассматриваются в последующих главах.

Программирование с помощью барабанного командоаппарата

Описывает четыре различных барабанных коммандоаппарата для задания последовательности по таймеру/событиям, доступных процессору DL405. Глава начинается с описания функционирования, включает методы управления и показывает, как барабанные коммандоаппараты решают проблемы последовательных процессов.

Стадийное программирование на RLLplus

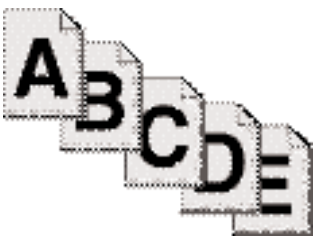
Идеальная для начинающего программиста, в ней показывается, как разработать простые диаграммы переходов состояния для вашего процесса и затем преобразовать их в стадийные программы. Этот метод программирования реально экономит время на программирование и отладку.

Организация контуров ПИД-Регулятора

Объясняет, как ПИД-регулятор DL405 может управлять 4 контурами. Обсуждается несколько проблем, включая переменные данные регулирования ПИД, настройку контура, аварийные сигналы, профили программного задатчика и другие.

Обслуживание и поиск неисправностей

Представляет собой указания по диагностике, восстановлению и предупреждению неисправностей системы. Глава включает подсказки, как обнаружить неисправности в цепях ввода/вывода, ошибки в программах лестничного типа и др.



Приложения Дополнительная справочная информация по DL405 приводится в следующих пяти приложениях:

Справочные приложения

- **A** — Вспомогательные функции
- **B** — Коды ошибок DL405
- **C** — Времена выполнения команд
- **D** — Специальные реле
- **E** — Веса модулей

Компоненты системы DL405

Семейство изделий DL405 включает максимально гибкие и широко распространенные программируемые микроконтроллеры (ПЛК), используемые в сфере управления. Процессоры, небольшие по размеру, но достаточно мощные. Их модульная структура и возможность расширения хорошо вписываются в быстро развивающуюся индустрию систем управления. Ниже приводится краткая сводка основных компонентов системы DL405.

Процессоры

Существует три варианта усовершенствованных блоков Процессора в данной серии продуктов: DL430, DL440 и новый DL450. Все Процессоры имеют встроенные источники питания и коммуникационные порты. В каждом Процессоре предлагается большой объем программной памяти, существенный набор команд и улучшенная диагностика. DL450 отличается возможностью программирования с помощью барабанного командоаппарата, математикой с плавающей точкой, встроенными контурами ПИД-регулятора и дополнительными коммуникационными портами. Детали этих и других особенностей процессора рассматриваются в главе 3 "Спецификации и работа процессора".

Каркасы

Для систем доступны три размера каркасов: на 4 слота, на 6 слотов и на 8 слотов.

Конфигурация ввода/вывода

Процессоры DL430 и DL450 могут поддерживать до 640 вводов/выводов в основном каркасе и каркасе расширения DL450 поддерживает 2048 точек. В систему можно добавить в виде каркасов удаленного ввода/вывода или секционированных блоков ввода/вывода еще до 512 дополнительных точек ввода/вывода для DL430, до 1024 точек для DL440 и до 1536 точек для DL450. Каждая из этих конфигураций рассматривается в Главе 4 "Проектирование и конфигурирование системы".

Модули ввода/вывода

DL405 имеет набор наиболее эффективных в промышленности модулей ввода/вывода. Предлагается полный набор дискретных модулей, которые поддерживают 24, 125 В постоянного тока, 110/220 В переменного тока и до 10 А на выходных реле. Аналоговые модули обеспечивают 12 — битовое разрешение и несколько вариантов диапазонов входных и выходных сигналов (включая биполярные). Специальные модули включают высоко скоростной (до 100 КГц) ввод импульсов, термопары, коммуникации общего назначения, магнитный импульсный ввод, функцию 16 контуров ПИД-регулятора и др.

Методы программирования

Имеется два метода программирования, доступных для процессоров DL405: RLL (язык релейной логики) и RLLplus (Стадийное программирование). Как пакет программирования DirectSOFTTM, так и Ручной Программатор поддерживают RLL и Стадийное программирование.

Программирование в DirectSOFT для Windows™

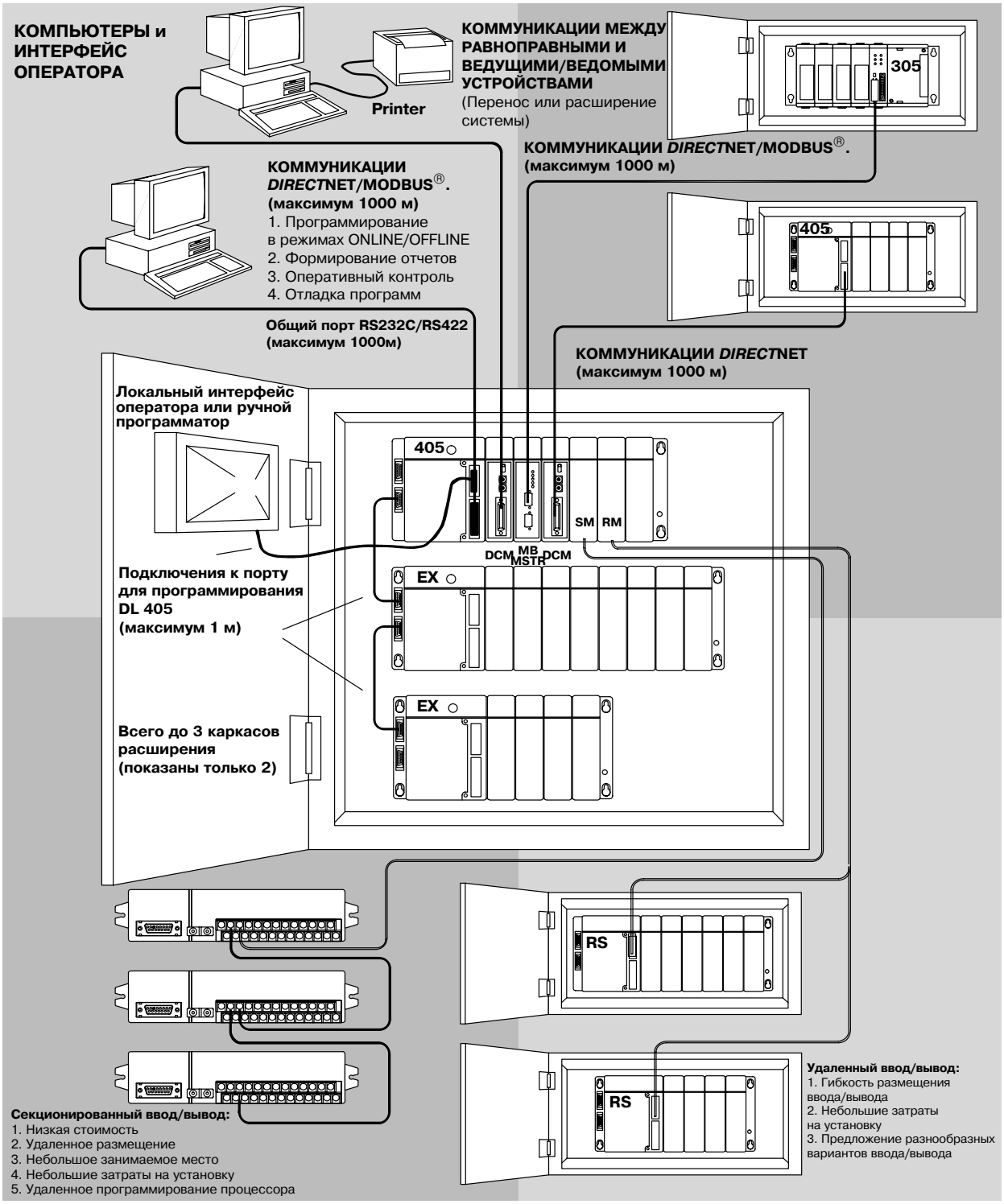
DL405 можно программировать с помощью одного из самых современных в промышленности пакетов — DirectSOFT. DirectSOFT является программным пакетом на базе Windows, он поддерживает многие хорошо известные вам функции Windows, такие как "вырезание и вставка" между приложениями (cut and paste), "указания и щелчка мышью" (point and click), просмотр и редактирование одновременно многих прикладных программ и др. DirectSOFT поддерживает все семейство процессоров DirectLOGICTM. Таким образом, вы можете использовать один и тот же пакет DirectSOFT для программирования DL05, DL105, DL205, DL305, DL405 или любого нового процессора, который мы можем добавить в нашу серию продуктов. Программное обеспечение DirectSOFT рассматривается в отдельном руководстве.

Ручной программатор

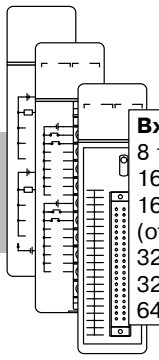
Все Процессоры DL405 имеют встроенный порт для программирования с использованием Ручного программатора (D4-HPP). Ручной программатор можно применять для создания, изменения или отладки вашей прикладной программы. Ручной программатор рассматривается в отдельном руководстве.

Примеры структурных схем системы DL405

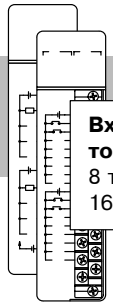
На приведенной ниже схеме показаны главные компоненты и конфигурации системы DL405. На следующих двух страницах показаны конкретные компоненты, необходимые для построения вашей системы.



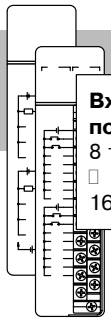
Серия DirectLOGIC DL405



Вход постоянного тока
8 точек 24-48 В
16 точек 12-24 В
16 точек 12-24 В
(отклик 1 мс)
32 точки 24 В
32 точки 5-12 В
64 точки 24 В

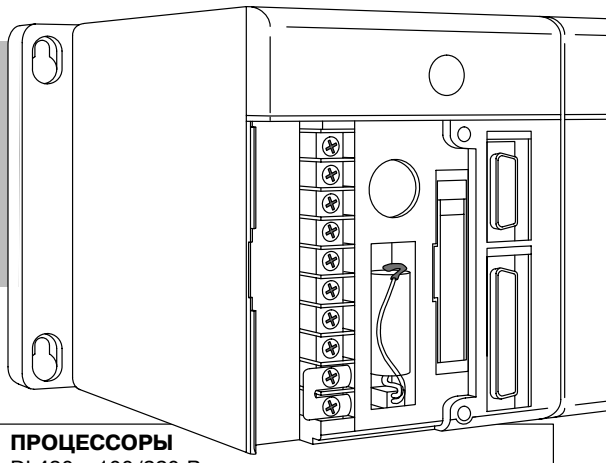


Вход переменного тока
8 точек 110 - 200 В
16 точек 110 В



Вход переменного/ постоянного тока
8 точек 90 - 150 В, изолированные
16 точек 12 -24 В

ПРОГРАММИРОВАНИЕ
Ручной программатор с встроенным RLL^{plus}
Программирование на DirectSOFT под Windows

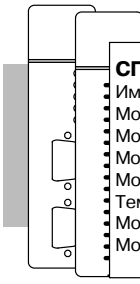


МОДУЛИ СОПРОЦЕССОРА^{TM}
RS232 / RS422 / RS485
Телефонный модем
Радиомодем
Области программной памяти
От 128K до 512K

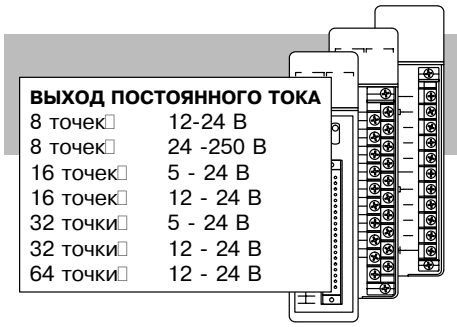
ПРОЦЕССОРЫ
DL430 - 100/220 В переменного тока
6.5 K встроенной памяти ЭППЗУ
DL440 -100/220 В переменного тока
22.5 K памяти (требуется картридж памяти)
DL440 -24 В постоянного тока
22.5 K памяти (требуется картридж памяти)
DL440 -125 В постоянного тока
22.5 K памяти (требуется картридж памяти)
DL450 - 110/220 В переменного тока
7.5 K встроенной Флэш - памяти

КАРКАСЫ
Каркасы на 4 слота (расширяемые и нерасширяемые)
Каркасы на 6 слотов (расширяемые и нерасширяемые)
Каркасы на 8 слотов (расширяемые и нерасширяемые)

КАРТРИДЖИ ПАМЯТИ
КМОП ОЗУ с батареей
ЭППЗУ с ультрафиолетовым стиранием информации
ЭППЗУ
КМОП ОЗУ без батареи



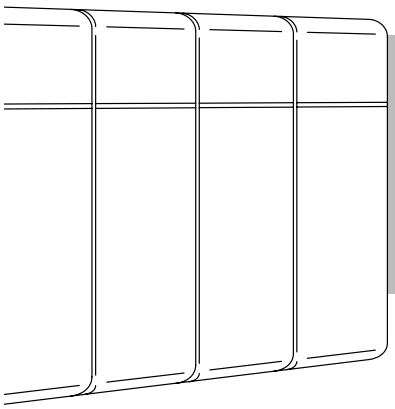
СПЕЦИАЛЬНЫЕ МОДУЛИ
Имитатор входа на 8/16 точек
Модуль ввода с прерыванием на 8 точек
Модуль высоко скоростного счетчика
Модуль магнитно-импульсного ввода на 8 точек
Модуль на 16 контуров ПИД-регулирования
Температурный контроллер на 4 контура
Модуль интерфейса SDSTM
Модуль Заглушки



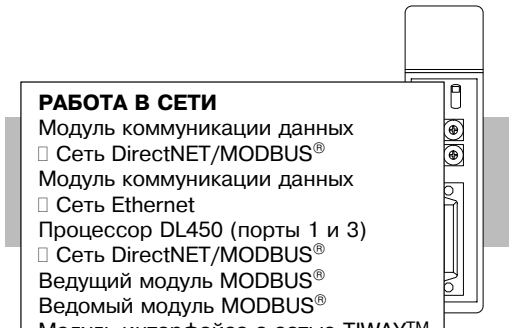
ВЫХОД ПОСТОЯННОГО ТОКА
8 точек 12-24 В
8 точек 24 -250 В
16 точек 5 - 24 В
16 точек 12 - 24 В
32 точки 5 - 24 В
32 точки 12 - 24 В
64 точки 12 - 24 В

ВЫХОД ПЕРЕМЕННОГО ТОКА
8 точек 18-110 В
16 точек 18-220 В

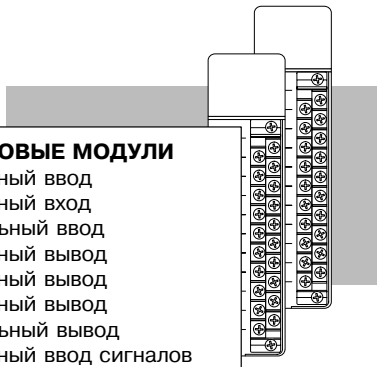
РЕЛЕЙНЫЙ ВЫХОД
8 точек 10 А на точку
8 точек 5 А на точку
8 точек 2 А на точку
16 точек 1 А на точку



Системы ввода/вывода
Каркасы расширения
Блок расширения локального каркаса
110 - 220 В переменного тока
Блок расширения локального каркаса
24 В постоянного тока
Блок расширения локального каркаса
125 В постоянного тока
Удаленный ввод/вывод
Ведущий модуль удаленного ввода/вывода
Ведомый блок удаленного ввода/вывода
10 - 220 В переменного тока
Ведомый блок удаленного ввода/вывода
24 В постоянного тока
Процессор DL450
Прямая связь с удаленным вводом/выводом с порта 3
Ведущий модуль секционированного ввода/вывода (Slice I/O)
Ведомое устройство секционированного ввода/вывода (8 ВХОДОВ
12 - 24 В постоянного тока и 8 ВЫХОДОВ 5 - 24 В постоянного тока)
Ведомое устройство секционированного ввода/вывода (10 ВХОДОВ
12 - 24 В постоянного тока и 6 Выходов 5 - 24 В постоянного тока)
Ведомое устройство секционированного ввода/вывода (16 ВХОДОВ
12 - 24 В постоянного тока)
Ведомое устройство секционированного ввода/вывода (16 ВЫХОДОВ
5 - 24 В постоянного тока)



РАБОТА В СЕТИ
Модуль коммуникации данных
Сеть DirectNET/MODBUS®
Модуль коммуникации данных
Сеть Ethernet
Процессор DL450 (порты 1 и 3)
Сеть DirectNET/MODBUS®
Ведущий модуль MODBUS®
Ведомый модуль MODBUS®
Модуль интерфейса с сетью TIWAY™
Модуль сети с совместно
используемым данным SDN

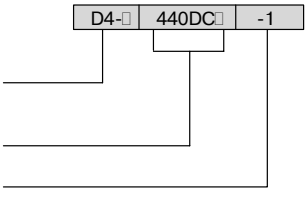


АНАЛОГОВЫЕ МОДУЛИ
4 канальный ввод
8 канальный вход
16 канальный ввод
2 канальный вывод
4 канальный вывод
8 канальный вывод
16 канальный вывод
8 канальный ввод сигналов
термосопротивлений

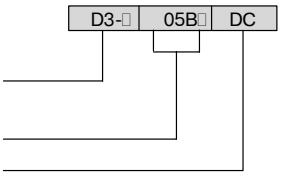
Система нумерации частей DirectLOGIC™

Системы управления на базе ПЛК DL405 включают много изделий семейства DL405. В следующих таблицах показывается, какая принята система нумерации частей для каждой категории устройств. Обозначениями для вспомогательных элементов, таких как кабели, батареи, картриджи памяти и др. обычно служат сокращенные их описания.

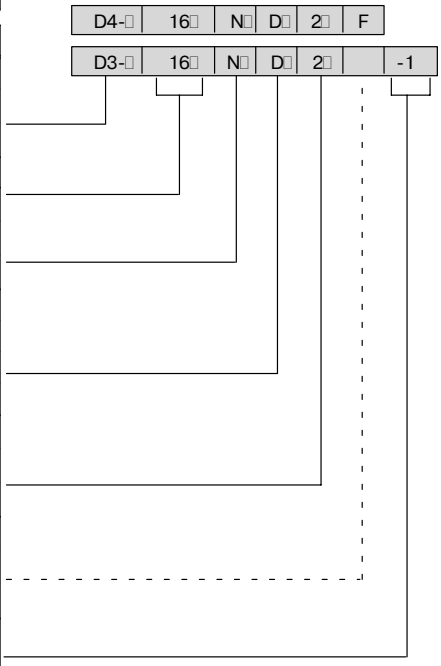
Процессоры и программируемые микроконтроллеры Специальные процессоры	
Семейство изделий	D1/F1 D2/F2 D3/F3 D4/F4
Класс процессора/сокращения	230...,330...,430...
Обозначение различия между подобными модулями	-1, -2, -3, -4



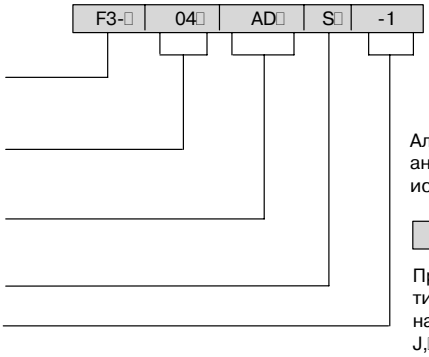
Каркасы	
Семейство изделий	D2/F2 D3/F3 D4/F4
Число слотов	##B
Тип блока питания □ - постоянного тока □ - переменного тока	DC пусто



Дискретный ввод/вывод	
Семейство изделий DL405	D2/F2
Семейство изделий DL305	D3/F3
Семейство изделий DL405	D4/F4
Число точек	04/08/12/16/32/64
Вход	N
Выход	T
Комбинация	C
Переменный ток (AC)	A
Постоянный ток (DC)	D
Любой	E
Реле	R
Приемник тока	1
Источник тока	2
Приемник/источник тока	3
Повышенный ток	H
Изоляция	S
Быстрый ввод/вывод	F
Обозначение различия между подобными модулями	-1, -2, -3, -4



Аналоговый ввод/вывод	
Семейство изделий DL405	D2/F2
Семейство изделий DL305	D3/F3
Семейство изделий DL405	D4/F4
Число каналов	02/04/08/16
Входные (аналоговые в цифровые)	AD
Выходные (цифровые в аналоговые)	DA
Комбинация	xADxDA
Изолированные	S
Обозначение различия между подобными модулями	-1, -2, -3, -4

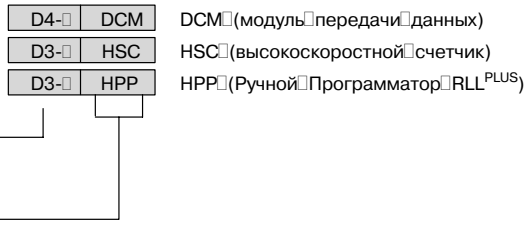


Альтернативный пример аналоговоговвода/вывода использующий сокращения

F3-□	08□	THM□	-n

Примечание: -n указывает тип термопары, как на пример, J, K, T, R, S или E

Коммуникационный и сетевой специальный ввод/вывод и устройства программирования	
Семейство изделий DL405	D2/F2
Семейство изделий DL305	D3/F3
Семейство изделий DL405	D4/F4
Сокращение имени	См. пример

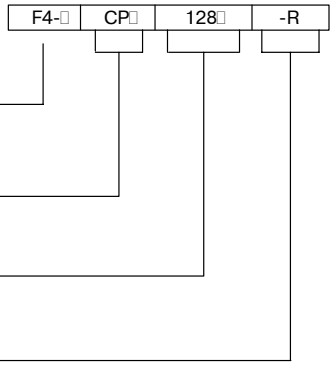


DCM (модуль передачи данных)

HSC (высокоскоростной счетчик)

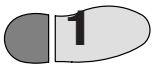
HPP (Ручной Программатор RLL^{PLUS})

Сопроцессоры и модули ASCII BASIC	
Семейство изделий DL405	D2/F2
Семейство изделий DL305	D3/F3
Семейство изделий DL405	D4/F4
Сопроцессор	CP
ASCII BASIC	AB
Память 64K	64
Память 128K	128
Память 512K	512
Модем радиосвязи	R
Модем телефонной связи	T



Как быстро начать проверку и программирование ПЛК

Если вы имеете опыт работы с ПЛК или хотите быстро собрать образец, то данный раздел — это то, что вам нужно использовать. Этот пример не предназначен для объяснения всего, что необходимо для запуска вашей системы. Он только содержит общее описание того, что необходимо для подключения питания к вашей системе.



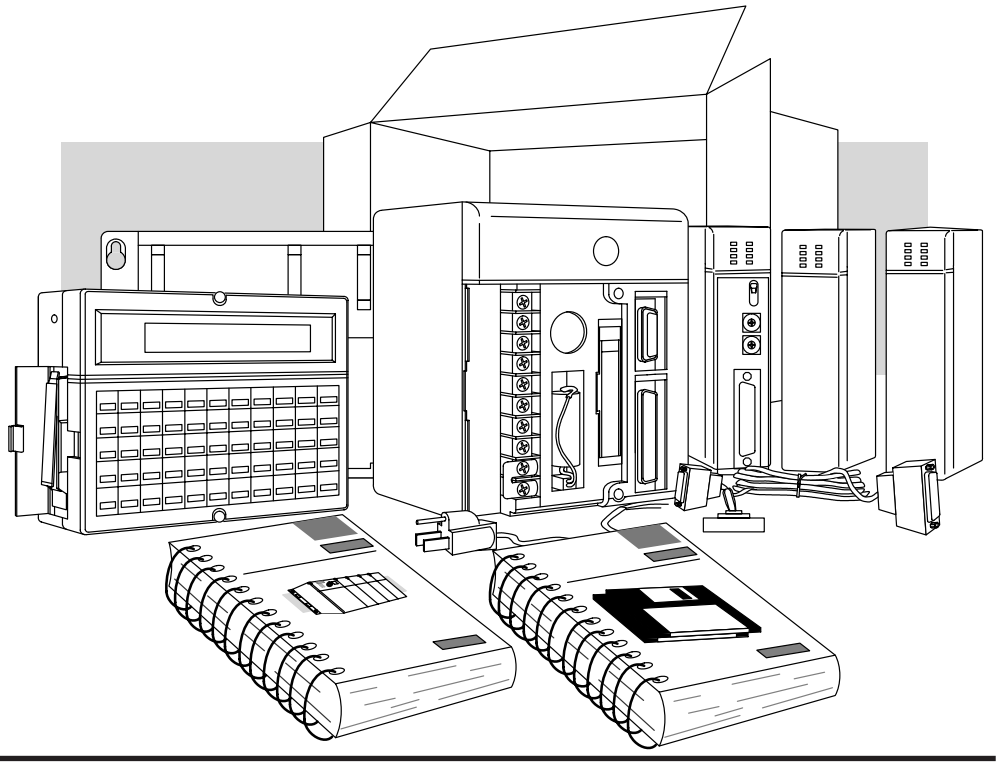
Шаг 1: Распаковать оборудование DL405

Распакуйте оборудование DL405 и проверьте, имеете ли вы все необходимые детали для построения вашей демонстрационной системы. Минимально необходимыми деталями являются:

- Каркас
- Процессор (с картриджем памяти, если Вы используете DL440 или DL450)
- Входной модуль постоянного тока D4-16ND2 или модуль входного имитатора D4-16SIM
- Выходной релейный модуль постоянного тока D4-16TR
- Шнур питания
- Провода для подключения
- Тумблер на 24В постоянного тока (если не используется модуль входного имитатора)
- Отвертка, обычная или типа ”Филлипс”

Вам необходимо иметь, по крайней мере, один из следующих вариантов для программирования:

- Программное обеспечение DirectSOFT, Руководство по DirectSOFT и кабель для программирования (подсоединяющего процессор к персональному компьютеру), или
- Ручной Программатор D4-HPP (кабель для подсоединения не обязателен) и Руководство по Ручному Программатору.

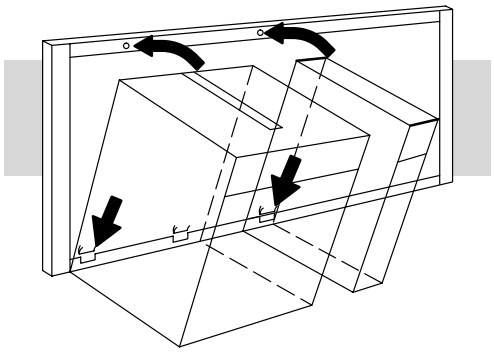


Шаг 2: Установить процессор и модули ввода/вывода

Вставьте процессор и модули ввода/вывода в каркас. Процессор должен быть установлен в крайнем левом слоте каркаса, имеющем надпись ”CPU/Power Supply” (”Процессор/Источник питания”). Когда Вы вставляете компонент в каркас, слегка наклоните его вперед и вдвиньте выступы в нижней части компонента в слот каркаса. Толкайте верхнюю часть компонента в каркас, пока он не будет твердо в нем установлен, затем затяните зажимной винт в верхней части модуля/блока.

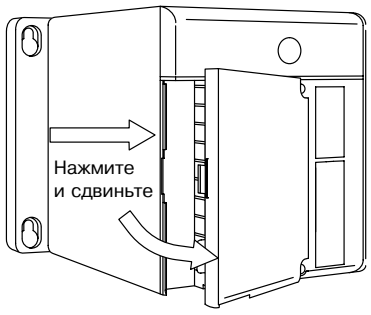
Местоположение дискретных, аналоговых и релейных модулей не критично, они могут быть установлены в любой слот любого каркаса, однако в данном примере установите выходной модуль в слот 1 рядом с процессором (слотом 0). Ограничения для других типов модулей рассматриваются в Главе 4 в разделах ”Каркасы, Блоки расширения и Конфигурация ввода/вывода”. Вы также не должны превышать потребляемую мощность каждым каркасом в вашей системной конфигурации. Расчет потребляемой мощности также рассматривается в Главе 4 ”Проектирование и конфигурирование системы”.

- Каждый блок имеет пластиковые выступы в нижней части и винты — верхней.
- При слегка наклоненном вперед блоке вставьте пластиковые выступы модуля в каркас.
- Мягко толкайте верхнюю часть блока, пока он не будет прочно установлен в каркасе.
- Закрепите блок в каркасе, затягивая верхний винт.



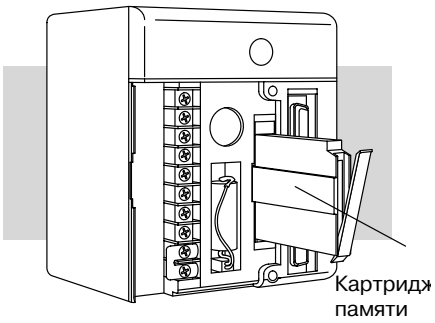
Шаг 3: Снять крышку доступа к клеммнику

Снимите крышку клеммника. С левого края она имеет небольшую предохранительную защелку. Нажмите на нее внутрь и вверх, затем снимите крышку.



Шаг 4: Установить картридж памяти

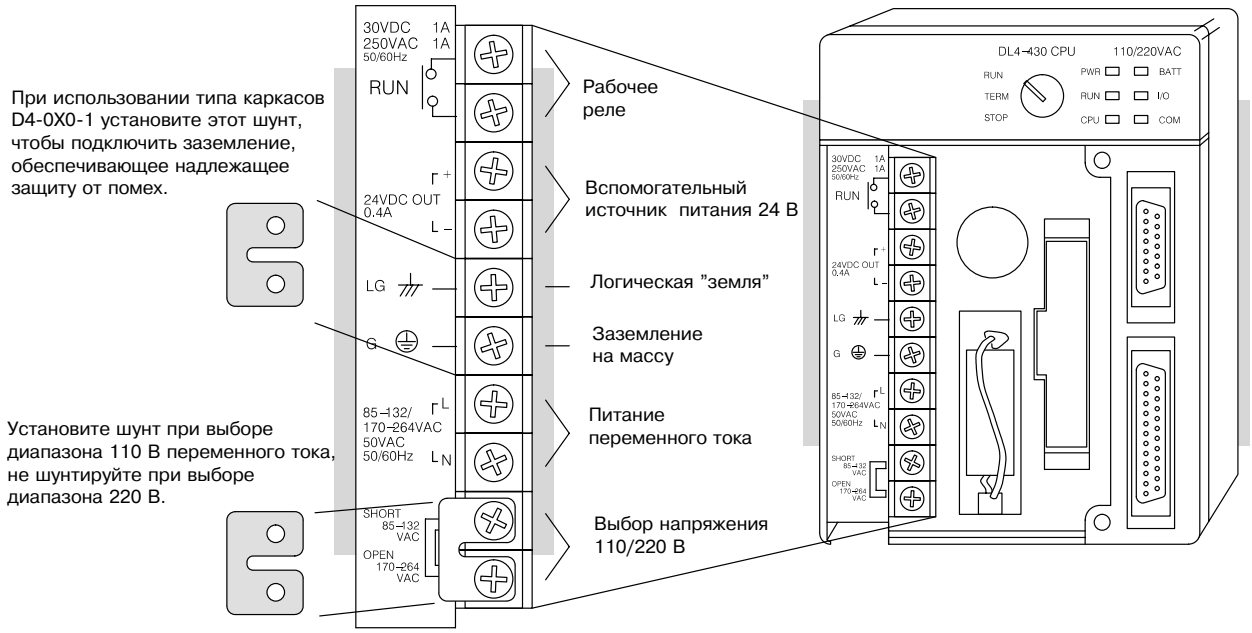
Если Вы используете процессор DL440 (это не обязательно для DL450), то Вы должны установить картридж памяти в слот справа от батареи. Убедитесь, что он надежно установлен. Более подробная информация по картриджам памяти приведена в главе 3.



Шаг 5: Выбрать рабочий диапазон напряжения

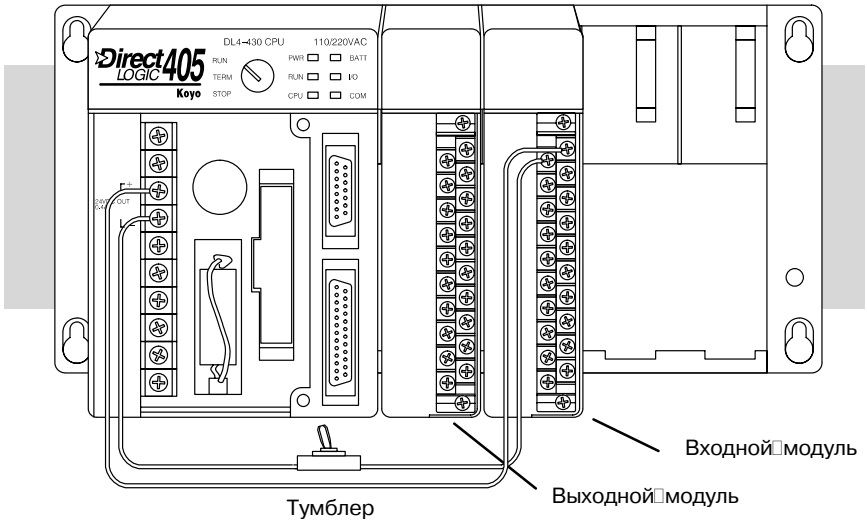
Если Вы используете 110 В переменного тока, установите перемычку выбора напряжения в нижней части ПЛК на два контакта. Если Вы используете питание 220 В переменного тока, перемычку не надо устанавливать. Более подробное описание клеммного блока как для процессора, так и для блоков расширения можно найти в главе 2 ”Установка, монтаж и спецификации”.

ПРЕДУПРЕЖДЕНИЕ. Можно повредить источник питания, если 220 В переменного тока подключить к контактам, зашунтированных на 110 В.



Шаг 6: Добавить имитатор ввода/вывода

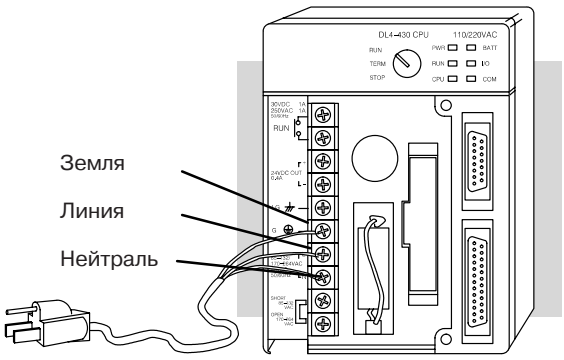
Для того чтобы завершить данный пример быстрого запуска или приступить к изучению других примеров данного руководства, вам необходимо установить модуль имитатора входа (или установить переключатель входа, как показано ниже) и добавить выходной модуль. Использование имитатора входа — самый быстрый способ получения физических входов для проверки системы или новых программ. Для отслеживания состояния выхода можно использовать любой дискретный выходной модуль.

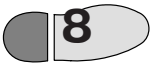


Перед подачей напряжения в систему установите выключатели или другие полевые устройства для гарантии того, что какая-либо точка не будет случайно включена при монтаже соединений. Подсоедините выходной модуль (X0) к тумблеру, а вспомогательный источник питания 24 В к клеммнику процессора, как показано на рис. В Главе 2 "Установка, монтаж и спецификации" приводится список руководящих указаний по монтажу ввода/вывода.

Шаг 7: Подсоединить разводку питания

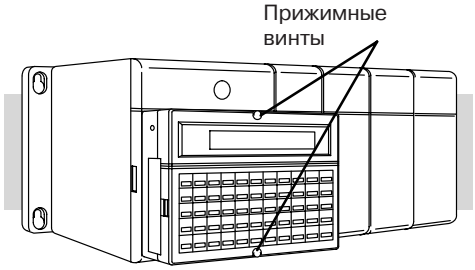
Подсоедините провода как показано на рисунке. Соблюдайте все предосторожности, указанные ранее в данном руководстве. Более детальную информацию по монтажу см. в главе 2 ”Установка, монтаж и спецификации”. Когда монтаж закончен, верните на место крышки процессора и модулей. Пока не подключайте электроэнергию.





Шаг 8: Подсоединить ручной программатор

Снимите крышку разъема в верхней правой части процессора. Подсоедините 15-контактный разъем типа D ручного программатора (расположенный на его задней стороне) к разъему процессора, с которого Вы сняли крышку. Затяните с усилием от руки прижимные винты в верхней и нижней части ручного программатора.



Шаг 9: Включить питание системы

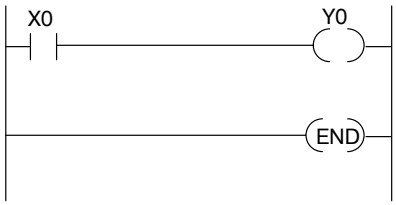
Подайте питание в систему и проверьте, работает ли индикатор PWR процессора. Если нет, то отключите питание системы и проверьте все соединения, обратитесь для помощи к разделу поиска неисправностей в главе 9.

Шаг 10: Ввести программу

Переведите переключатель процессора в положение STOP, а затем вернитесь в положение TERM. Это установит процессор в программный режим и даст доступ к программам процессора. Индикатор PWR на HPP (ручном программаторе) должен подсвечиваться. Введите на HPP следующие наборы клавиш:

ПРИМЕЧАНИЕ. Вам нет необходимости конфигурировать Ввод/Вывод для вашей системы, поскольку процессоры DL430 автоматически сконфигурирует ввод/вывод, а процессоры DL440 и DL450 по умолчанию выбирают автоматическую конфигурацию ввода/вывода.

CLR	CLR				
AUX	2	4	↵	↵	↵
CLR	$(AD)	0	NXT		
STR	X(IN)	0	↵		
OUT	Y(OUT)	0	↵		
END	↵				



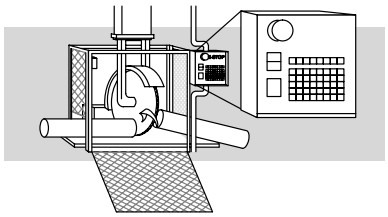
После ввода программы для простого примера переставьте переключатель из положения TERM в положение RUN, а потом верните в положение TERM. Индикатор RUN Процессора включится, указывая, что он установлен в рабочий режим. Если он не включится, то следует повторить шаг 10, чтобы ввести программу надлежащим образом, или обратиться к разделу поиска неисправностей в главе 9.

При операциях в рабочем режиме индикатор состояния выхода 0 в выходном модуле должен отражать положение переключателя. Когда переключатель находится в положении ”включен”, выход также должен быть включен.

Шаги по созданию хорошей системы

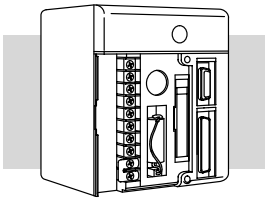
Шаг 1: Просмотреть руководства по установке

Всегда ставьте безопасность в качестве первого приоритета при любом приложении системы. В главе 2 дается несколько руководящих указаний, которые помогут создать безопасную, более надежную систему. Данная глава также включает руководящие указания по монтажу для различных компонентов системы.



Шаг 2: Понять методы настройки процессора

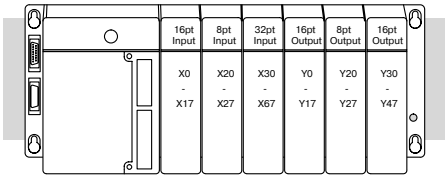
Процессор является сердцем вашей автоматической системы. Выделите время для того, чтобы разобраться с разнообразными функциями и требованиями к настройке.



Шаг 3: Понять конфигурации системы ввода/вывода

Важно понять, как сконфигурировать вашу систему ввода/вывода. Вы имеете несколько различных типов этих систем:

- Локальная система
- Система расширения
- Система удаленного ввода/вывода
- Сетевые подключения

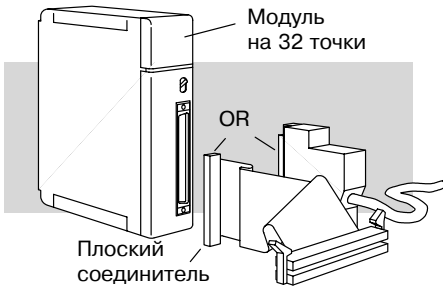


Также важно понять, как рассчитывается потребляемая мощность системы. Более подробную информацию см. в главе 4.

Шаг 4: Определить спецификации модулей и монтажные характеристики

Это повлияет на ваше размещение точек Ввода/Вывода и/или на варианты конфигурации.

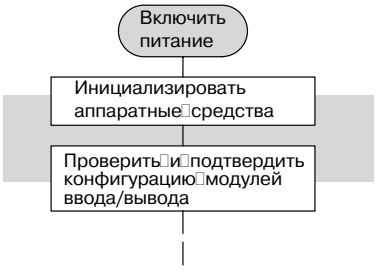
Существует много различных модулей ввода/вывода, пригодных для системы DL405. В главе 2 приводятся спецификации и схемы подключения для дискретных модулей ввода/вывода.

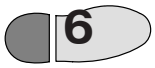


ПРИМЕЧАНИЕ. Для аналоговых и других специальных модулей имеются собственные руководства, они не включены в данное руководство.

Шаг 5: Понять работу процессора

Перед началом ввода программы весьма полезно понять, как процессор DL405 обрабатывает информацию. Сюда относятся не только шаги выполнения программы, но также различные режимы работы и характеристики распределения памяти. См. главу 3 с более детальной информацией.



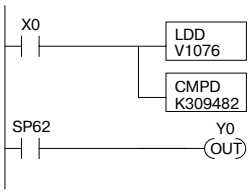


Шаг 6: Просмотреть концепции программирования

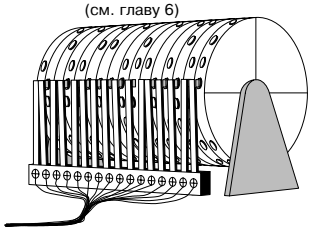
DL405 предусматривает четыре главных подхода к программированию прикладных задач, включая задачи ПИД-регулирования, изображенные на следующем рисунке.

- Программирование релейного типа является наилучшим инструментом для решения задач булевой логики и для манипуляции с регистром/аккумулятором процессора. Этот язык включает дюжины команд, которые могут быть дополнены барабанными командоаппаратами, стадиями и контурами.
- DL405 имеет четыре типа барабанных командоаппаратов с управлением по таймеру/событиям, каждый включает до 16 шагов. В каждом из них предлагаются пошаговые переходы на базе времени и/или по событиям. Барабанные командоаппараты являются наилучшим инструментом для повторяющегося процесса с одной последовательностью шагов.
- Стадийное программирование (называемое также RLLplus) основано на диаграммах перехода состояния. Стадии в этом языке разделяют программу на секции, соответствующие состояниям в блок-схеме вашего процесса.
- При организации контуров ПИД-регулятора DL450 используются таблицы настройки, позволяющие сконфигурировать до 16 контуров. Имеются также такие функции, как настройка контура, задание аварийных сигналов, программный задатчик и другие.

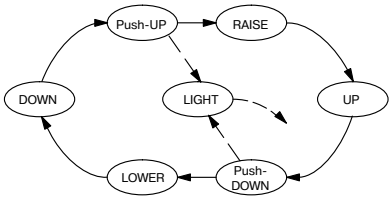
Стандартное программирование на RLL
(см. главу 5)



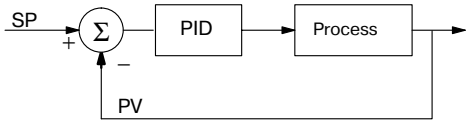
Барабанный коммандоаппарат последовательности по таймеру/событиям
(см. главу 6)



Стадийное программирование
(см. Главу 7)

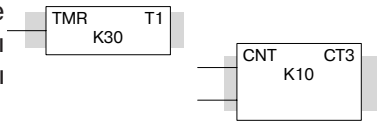


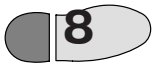
Организация контуров PID-регулятора
(см. Главу 8)



Шаг 7: Выбрать команды

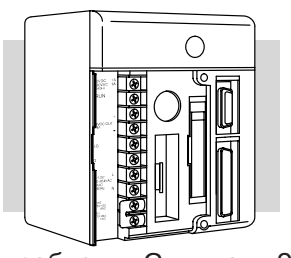
После того, как вы установили систему и изучили концепции программирования, Вы можете приступить к написанию прикладной программы или к конфигурированию работы контуров. Вы можете открыть эффективный набор команд!





Шаг 8: Понять методы обслуживания и поиска неисправности

Оборудование может отказать, когда мы это меньше всего ожидаем. Переключатели могут выйти из строя, в нагрузке может возникнуть короткое замыкание и потребоваться замена и т. д. В большинстве случаев при поиске неисправности и при техническом обслуживании основное время тратится на локализацию проблемы. Система DL405 имеет много встроенных средств, например, коды ошибок, которые помогут вам быстро идентифицировать проблему. См. главу 9 по диагностике и с подсказками при поиске неисправности.



Часто задаваемые вопросы

Вопрос. Что нового в этой третьей редакции руководства пользователя?

Ответ. Процессор DL450! Он имеет математику с плавающей точкой, управление 16 контурами ПИД- регулирования, четыре типа команд барабанного командоаппарата, способность к взаимодействию встроенного пакета MODBUS и удаленного ввода/вывода, больший объем памяти, более быстрый внутренний процессор (фактически используются 2 процессора) и многоуровневый пароль. Само руководство имеет новую компоновку и новую главу по Стадийному программированию.

Вопрос. Чем отличается новый тип каркаса?

Ответ. Каркас с "расширенной шиной" позволяет процессору DL450 использовать специальные модули ввода/вывода в локальных каркасах расширения, если каркасы расширения, как и основной каркас, имеют тип "с расширенной шиной". Номера этих частей имеют суффикс "-1". Эти каркасы являются "совместимыми назад" со стандартными каркасами, они имеют ту же стоимость.

Вопрос. Могу ли я иметь в системе более 16 контуров ПИД- регулирования при использовании DL450?

Ответ. Да. Вы можете еще использовать модули на 16 контуров ПИД плюс контуры DL450.

Вопрос. По спецификациям кажется, что DL450 более медленный, чем DL440. Верно ли это?

Ответ. Фактически нет. Стандартная скорость "1 K булевых операций" действительно показывает, что DL440 быстрее, но в большинстве приложений DL450 используются более сложные функции, чем булевы. Двоичная математика, математика с плавающей точкой, контуры ПИД- регулирования, барабанные командоаппараты и обработка данных в DL450 в 5 - 10 раз быстрее, чем аналогичные операции в DL440, если они существуют.

Вопрос. Можно ли на DL450 обрабатывать отрицательные действительные числа?

Ответ. Да. Действительные числа в DL450 представляются в 32-битовом формате IEEE. Поэтому Вы можете отображать очень большие и очень маленькие числа, как положительные, так и отрицательные.

Вопрос. Какие новые сетевые возможности у DL450 ?

Ответ. Напоминаем, что процессоры DL430/DL440 могут быть ведомыми устройствами DirectNET. Но DL450 имеет два сетевых порта, и он может быть ведущим или ведомым устройством DirectNET, а также ведущим или ведомым устройством MODBUS.

Вопрос. Какую версию DirectSOFT должен я иметь, чтобы программировать DL450?

Ответ. Для программирования DL450 нужна версия DirectSOFT 2.0 или более поздняя.

Вопрос. Можно ли на ручном программаторе D4-HPP программировать DL450?

Ответ. Вам необходима новая версия встроенного программного обеспечения 5.5 или более поздняя. Новый HPP с номером части D4-HPP-1 поставляется с новым встроенным программным обеспечением. Чтобы модернизировать существующий D4-HPP, Вы можете заказать усовершенствованный комплект D4-HPP-U. Пожалуйста обратите внимание на то, что DL450 имеет новые сложные функции, такие как барабанные командоаппараты, контуры ПИД- регулирования, которые гораздо легче программировать при использовании DirectSOFT, в настоящее время не существует способов вводить непосредственно с ручного программатора действительные числа (отсутствует десятичная точка).

Вопрос. Я понял, что для работы с DL450 специальные модули D4 требуют изменения встроенного программного обеспечения...

Ответ. Изделия, изготавливаемые с октября 1996 г., уже имеют новое встроенное программное обеспечение и совместимы с DL450. Ранее изготовленные изделия требуют новое программное обеспечение.

Установка, монтаж и спецификации

2

В этой главе...

— Руководство по безопасности

— Руководство по монтажу

— Установка каркасов DL405

— Установка компонентов в каркас

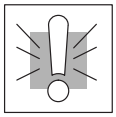
— Указания по монтажу процессора и блока расширения

— Стратегии монтажа подсистемы ввода/вывода

— Монтаж и спецификации модулей ввода/вывода

— Словарь терминов в спецификациях

Руководство по безопасности



ПРЕДУПРЕЖДЕНИЕ. Обеспечение безопасной рабочей среды для персонала и оборудования является вашей обязанностью и должно стать вашей первостепенной целью при планировании и установке системы. Автоматические системы могут иметь отказы и могут приводить к ситуациям, в которых могут иметь место серьезные травмы персонала и повреждение оборудования. Не полагайтесь только на автоматическую систему для обеспечения безопасной рабочей среды. Вы должны использовать внешние электромеханические устройства, независимые от приложений ПЛК, такие как реле или концевые выключатели, чтобы обеспечить защиту для любой части системы, которая может травмировать персонал или повредить оборудование.

План по технике безопасности

Каждая область применения автоматики имеет отличия, поэтому должны быть предусмотрены специальные требования для вашей конкретной системы. Убедитесь, что вы соблюдаете все национальные, государственные и местные требования к правильной установке и применению вашего оборудования.

Наилучший способ обеспечения безопасной рабочей среды состоит в том, чтобы сделать безопасность персонала и оборудования частью процесса планирования. Вы должны проверить каждый аспект системы, чтобы определить какие области являются критичными для безопасности оператора и техники.

Если у Вас не было практики установки ПЛК или в вашей организации нет общепринятых руководств по установке, то вы должны получить дополнительную информацию из следующих источников.

- NEMA — Национальная Ассоциация Производителей электрооборудования, находится в Вашингтоне, публикует много разнообразных документов, в которых рассматриваются стандарты по промышленным системам управления, Вы можете заказать эти публикации непосредственно в NEMA. Некоторые из них:
  ICS 1, Общие Стандарты для Промышленного Контроля и Систем
  ICS 3, Промышленные Системы
  ICS 6, Шкафы для Промышленных Систем Управления
- NEC — Национальные Электрические Правила предусматривают требования, касающиеся установки и применения различных типов электрооборудования. Копии Справочного Руководства NEC часто можно получить у вашего местного дистрибьютора электрооборудования или в вашей местной библиотеке.
- Местные и Государственные Агентства — многие местные и государственные органы выдвигают дополнительные требования сверх и помимо тех, которые описаны в Справочном Руководстве NEC. Обратитесь в ваши местные органы энергонадзора.
- ПУЭ — Правила устройства электрооборудования.

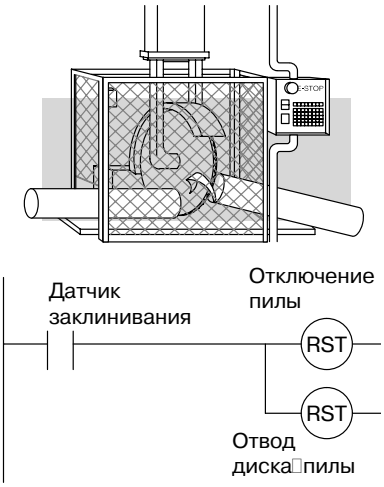
Техника безопасности

В указанных публикациях содержится много соображений и требований к безопасности систем. Как минимум, вы должны следовать этим правилам. Использование перечисленных ниже методов дополнительно поможет вам уменьшить риск при решении проблем безопасности.

- Порядок отключения системы в программе управления ПЛК
- Отключение питания системы (с помощью выключателей защитных ограждений, аварийного останова и др.).

Порядок отключения системы

Управляющая программа ПЛК может обеспечить первый уровень защиты при идентификации неисправностей механизмов. Проанализируйте ваше приложение и определите порядок отключения, который должен соблюдаться. Типичными неисправностями являются заклиненные или отсутствующие детали, пустые бункеры и др., которые не представляют собой риск травмирования персонала или повреждения оборудования.



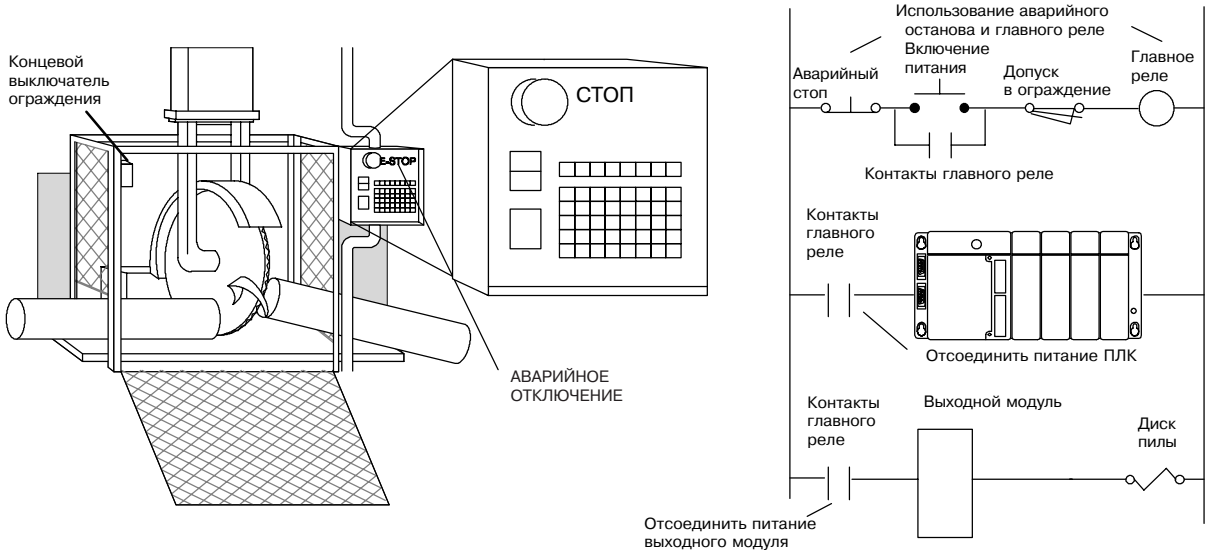
ПРЕДУПРЕЖДЕНИЕ. Управляющие программы не должны быть единственным способом защиты при всех неисправностях, которые связаны с риском травмирования персонала или повреждения оборудования.

Отключение питания системы

Используя электромеханические устройства, такие как главные управляющие реле и/или концевые выключатели, вы можете предотвратить случайный запуск оборудования. Установленные надлежащим образом эти устройства предотвратят выполнение любых машинных операций.

Например, если в механизме заклинило деталь, то управляющая программа ПЛК может отключить пилу и отвести диск пилы. Однако, поскольку оператор должен входить за ограждение для удаления детали, вы должны предусмотреть запасной выключатель, чтобы отключать питания всей системы всякий раз, когда открывается ограждение.

Оператор машины должен также иметь быстрый способ отключения питания всей системы. Это должно выполняться с помощью механического устройства, помеченного как выключатель Аварийный Останов.



После Аварийного отключения или после любого другого прерывания питания необходимо выполнить некоторые условия перед тем, как управляющая программа ПЛК может быть запущена. Например, могут существовать конкретные значения регистра, которые должны быть установлены (или сохранены по состоянию до отключения) перед тем, как операции будут возобновлены. В этом случае вы можете использовать ячейки памяти для сохранения или включить константы в управляющую программу для обеспечения известной начальной точки.

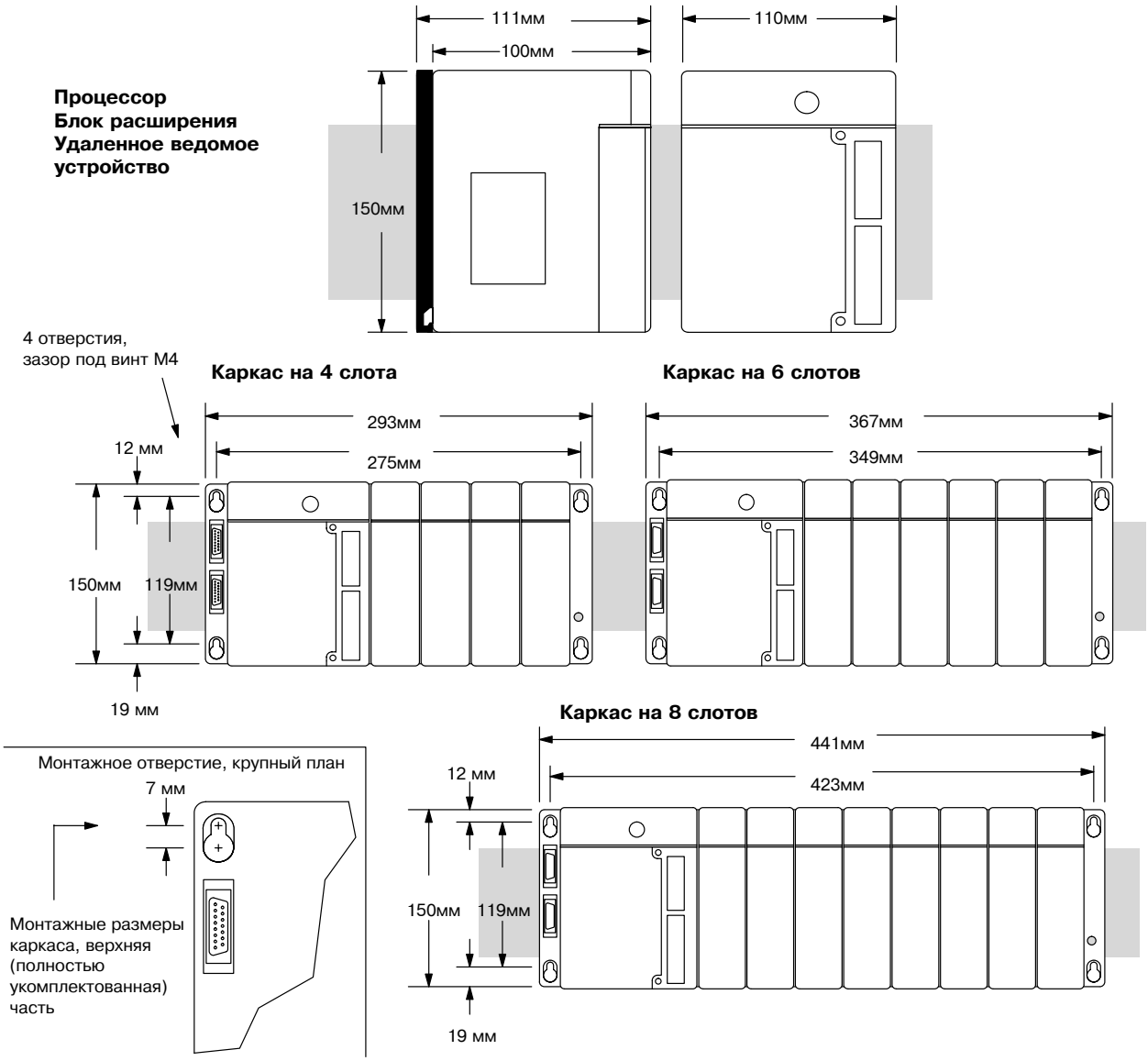
Руководство по монтажу

Кроме руководства по компоновке шкафа вам необходимо рассматривать другие спецификации, которые могут повлиять на установку системы ПЛК. Всегда рассматривайте следующее:

- Технические условия окружающей среды,
- Спецификации источников питания,
- Согласование с организациями по надзору,
- Размеры компонентов и выбор шкафа.

Размеры каркаса

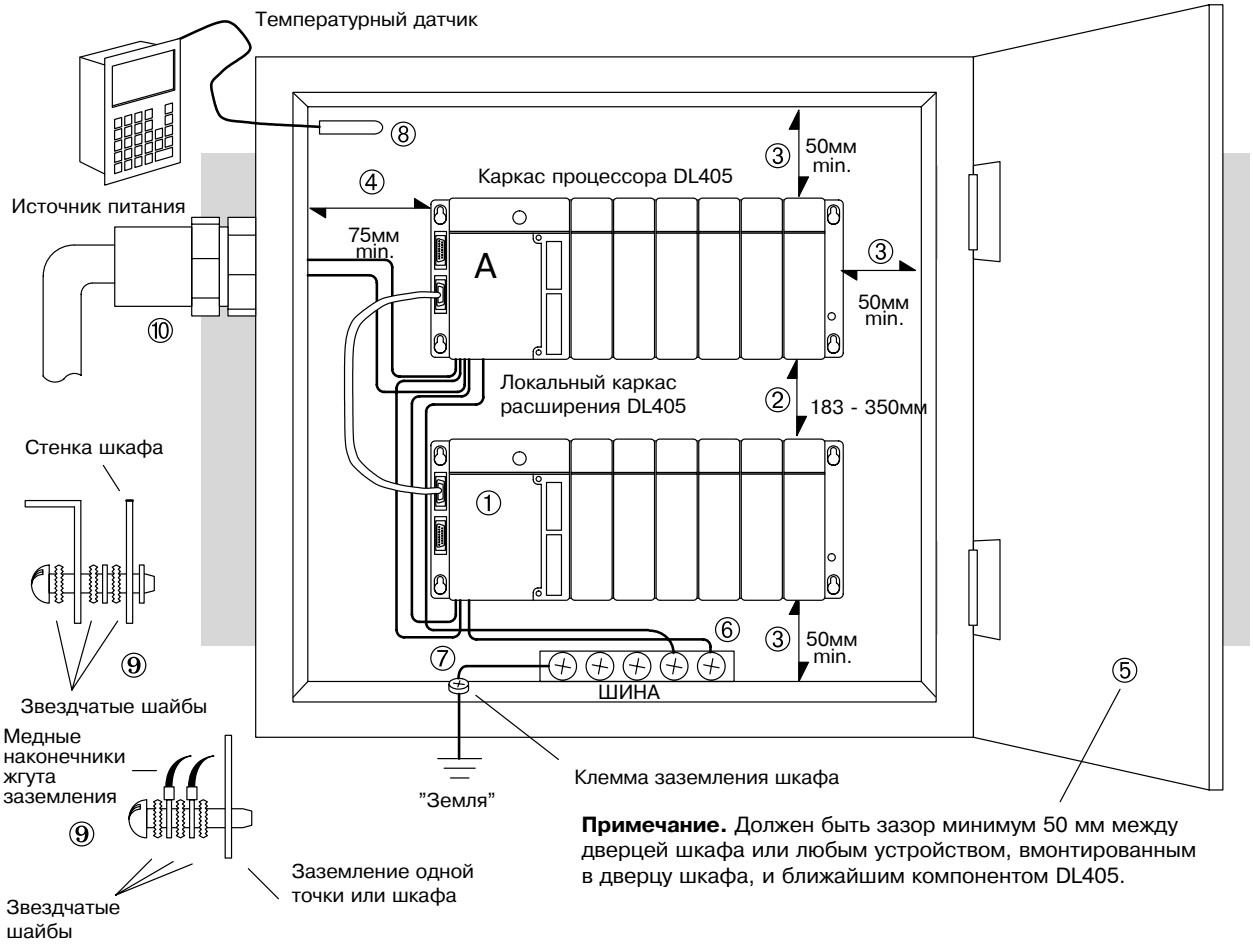
На следующей схеме показаны внешние размеры и размещение монтажных отверстий для каркасов на 4 слота, на 6 лотов и на 8 слотов. Убедитесь, что вы выбираете расстояния между компонентами в соответствии с руководством по установке.



Компоновка шкафа и зазоры

При компоновке шкафа необходимо рассматривать много аспектов. Следующие ниже пункты соответствуют приведенной схеме. Примечание: могут существовать дополнительные требования, зависящие от вашего приложения и размещения в шкафу других компонентов.

1. Каркасы должны устанавливаться горизонтально, чтобы обеспечить хорошую вентиляцию.
2. Между каркасами должно быть расстояние минимум 183мм и максимум 350 мм.
3. Между каркасом и верхней, нижней и правой стенками шкафа должен быть зазор минимум 50 мм.
4. Между каркасом и левой стенкой шкафа должен быть зазор минимум 75 мм.
5. Между дверцей шкафа и ближайшим компонентом DL405 должен быть зазор минимум 50 мм.



Примечание. Должен быть зазор минимум 50 мм между дверцей шкафа или любым устройством, вмонтированным в дверцу шкафа, и ближайшим компонентом DL405.

6. Клемма заземления каркаса DL405 должна быть соединена с единой точкой заземления. Используйте медный витой провод для снижения сопротивления. Медные наконечники должны быть обжаты и припаяны к концам витого провода, чтобы гарантировать хороший поверхностный контакт. Удалите анодированное покрытие и используйте медные наконечники и звездчатые шайбы в точках подключения. Общее правило состоит в том, чтобы получить сопротивление постоянного тока 0.1 ом между каркасом DL405 и единой точкой заземления.

7. Должна быть единая точка заземления (то есть общая шина) для всех устройств в шкафу, требующих заземления. Единая точка заземления должна соединяться с клеммой заземления шкафа.

   Клемма заземления шкафа должна соединяться с "Землей". Для этого соединения вы должны использовать, как минимум, витой медный провод 3.1 мм. Минимальные размеры проводов, цветовая маркировка и общие способы обеспечения безопасности должны подчиняться соответствующим электротехническим правилам и стандартам вашего региона.

   Хороший общий контакт заземления существенен для надлежащей работы DL405. Существует несколько методов обеспечения адекватного общего контакта заземления, в том числе:

   a) Установка заземляющего стержня как можно ближе к шкафу.

   b) Соединение с заземлением входного питания системы.

8. Тщательно проверьте любые места установки аппаратуры, где окружающая температура приближается к определенной в спецификации нижней или верхней границе. Поместите температурный датчик в шкаф, закройте дверцу и дайте поработать системе до тех пор, пока температура окружающей среды не стабилизируется. Если температура окружающей среды выходит за пределы действующей спецификации DL405, то примите меры, такие как установка источника охлаждения/нагрева, чтобы получить температуру в пределах действующих спецификаций DL405

9. Монтажными болтами устройств и болтами заделки жгутов заземления должны быть медные болты M25 или эквивалентные им. Везде, где это возможно, должны применяться резьбовые отверстия вместо болтов с гайками. Для того чтобы обеспечить хороший контакт в местах подключения, необходимо удалить с поверхности контакта краску, покрытие или ржавчину.

10. Система DL405 рассчитана на питание 110, 220 В переменного тока или 24 В постоянного тока, которые обычно доступны в промышленных условиях. Развязывающие трансформаторы, устройства подавления помех обычно не являются необходимыми, но могут быть полезными при исключении/сокращении возможных проблем с питанием.

Шкафы

Выбор подходящего шкафа важен для обеспечения безопасной и правильной работы вашей системы DL405. Приложения систем DL405 разнообразны, для них могут потребоваться дополнительные условия. Минимальными условиями выбора шкафа являются:

- Соответствие электротехническим стандартам
- Защита от предметов промышленного окружения
- Общее заземление
- Поддержание специфицированной температуры окружающей среды
- Доступ к оборудованию
- Обеспечение безопасности или ограничение доступа
- Достаточное пространство для надлежащей установки и обслуживания оборудования

Технические условия окружающей среды

В приводимой ниже таблице приведены технические условия окружающей среды, которые приемлемы для системы DL405 (процессора, блока расширения, каркасов, модулей ввода/вывода). Пределы значений для ручного программатора указаны под данной таблицей. Функционирование модуля ввода/вывода может изменяться в зависимости от температуры окружающей среды и вашего приложения. Пожалуйста, обратитесь к соответствующей спецификации конкретного модуля ввода/вывода для определения кривой снижения параметров в зависимости от температуры.

Техническое условие	Номинальное значение
Температура хранения	От - 20˚ C до 70˚ C*
Рабочая температура окружающей среды*	От 0˚ до 60˚ C
Влажность окружающей среды	5 - 95 % относительная влажность (без конденсата) **
Сопротивление вибрации	MIL STD 810C, метод 514.2
Сопротивление ударной нагрузке	MIL STD 810C, метод 516.2
Помехозащищенность	NEMA (ICS3 - 304)
Атмосфера	Отсутствие агрессивных газов

* Температура хранения для ручного программатора составляет от - 10˚ до 65˚ C
** Влажность окружающей среды для ручного программатора - от 20 до 90%,
 без конденсата.

Питание

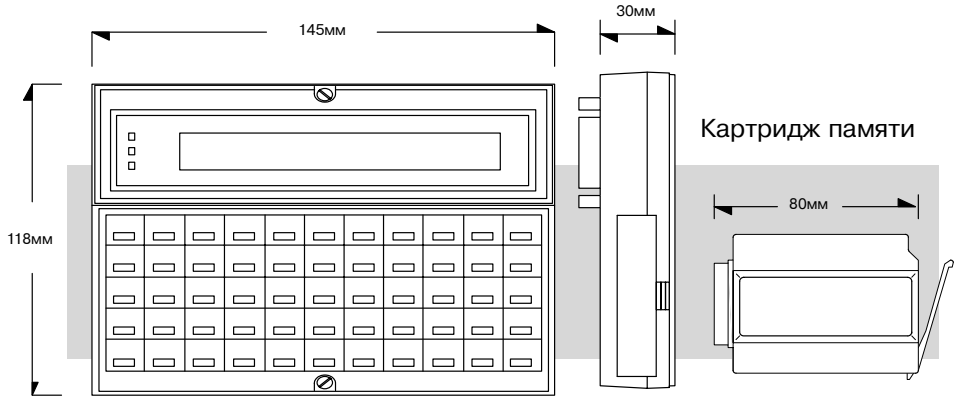
Внешний источник питания должен поддерживать напряжение и ток в соответствии с техническими условиями для блоков питания каркаса ПЛК.

Технические условия	Каркасы с питанием от переменного тока Каркасы с питанием 24 В постоянного тока
Стойкость к напряжению (электрическая прочность диэлектрика)	1 мин при 1500 В переменного тока между основным, дополнительным, эксплуатационным заземлением и пусковым реле
Сопротивление изоляции	> 10 МОм при 500 В постоянного тока
Диапазон входного напряжения D4-430/ D4-440/ D4-450/D4-EX	85 - 132В / 170-264В
Диапазон входного напряжения D4-440DC-1/ D4-EXDC	20 - 29 В постоянного тока с пульсацией менее 10%
Диапазон входного напряжения D4-440DC-2/ D4-EXDC-2	90 - 146 В постоянного тока с пульсацией менее 10%
Максимальный неразрушающий ток D4-430/ D4-440/ D4-EX	20 А
Максимальный неразрушающий ток D4-440DC-1/ D4-EXDC	10 А
Максимальный неразрушающий ток D4-440DC-2/ D4-EXDC-2	20 А
Максимальная мощность D4-430/ D4-440/ D4-450/D4-EX	50 ВА
Максимальная мощность D4-440DC-1/ D4-EXDC	38 Вт
Максимальная мощность D4-440DC-2/ D4-EXDC-2	30 Вт
Вспомогательный источник питания 24 В постоянного тока (только D4-EX)	20 -28 В постоянного тока при 0.4 А максимум, пульсация > 1 В для р - р

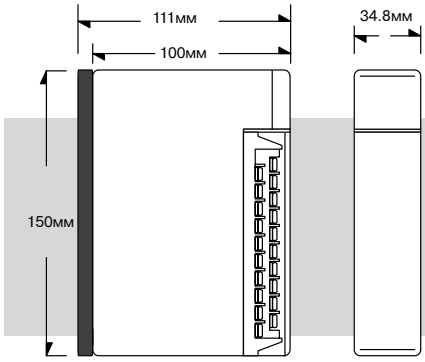
Размеры компонентов

Для установки ПЛК вам необходимо знать размеры компонентов в вашей системе. На рисунках на этой странице приводятся размеры компонентов, они должны использоваться для установления технических характеристик ваших шкафов. Не забудьте оставить место для возможного расширения. В приложении E приводится вес каждого компонента.

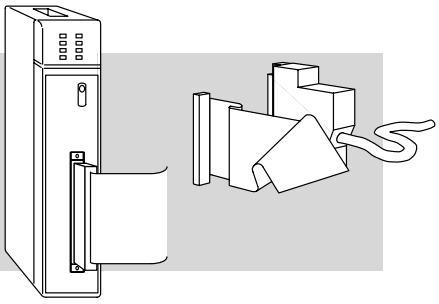
Ручной программатор



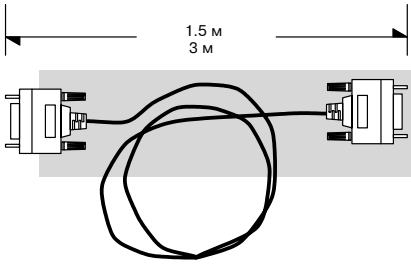
Модули ввода/вывода



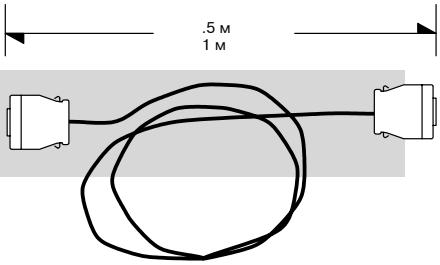
I/O module w/Ribbon connector



Кабель для ручного программатора



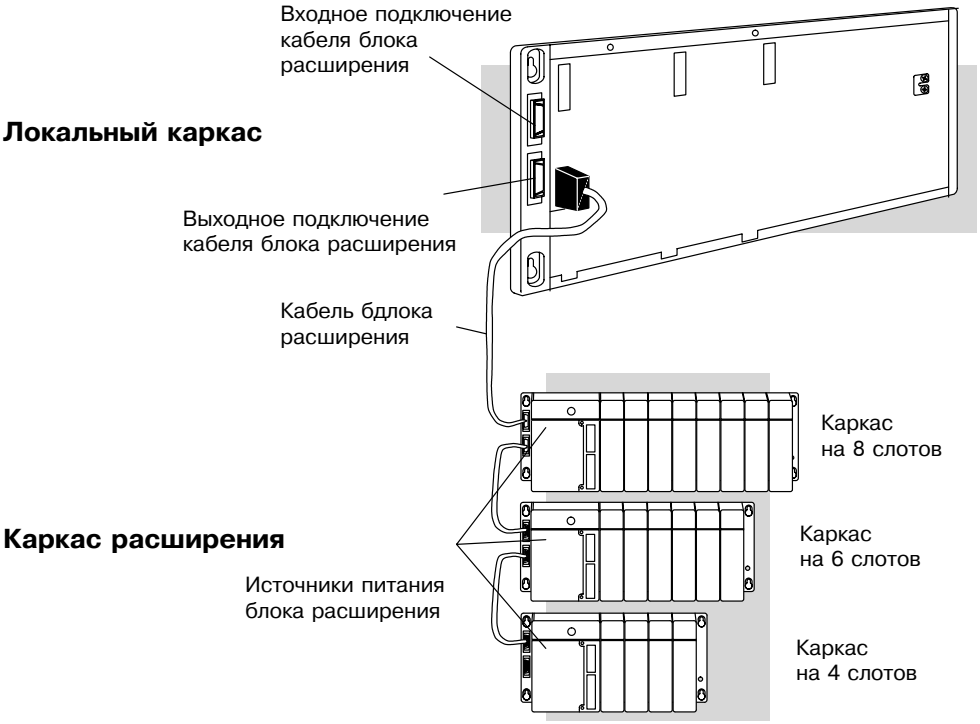
Кабель для каркаса расширения

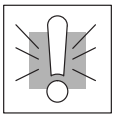


Установка каркасов DL405

Три размера каркасов

Во всех конфигурациях ввода/вывода выбираются каркасы либо на 4, либо на 6, либо на 8 слотов. Локальные каркасы и каркасы расширения могут иметь размеры на 4, 6 и 8 слотов. Эти каркасы различаются только схемой подключения к системе.



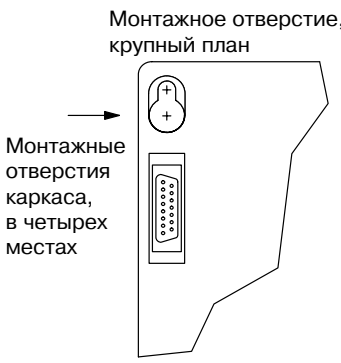


ПРЕДУПРЕЖДЕНИЕ. Чтобы минимизировать риск электрического удара, поражения персонала и повреждения аппаратуры, всегда отключайте питание системы перед установкой или снятием любого компонента системы.

Монтаж каркаса

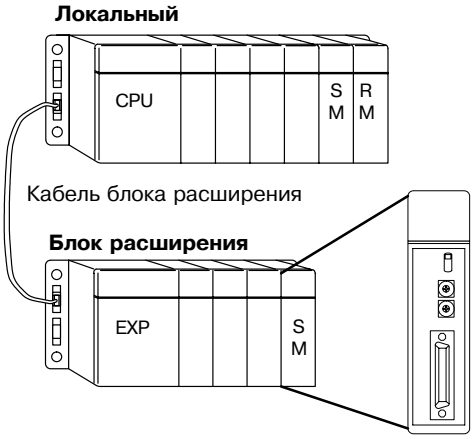
Процессор/Блок расширения/ Удаленное ведомое устройство всегда должны устанавливаться в самый левый слот каркаса. Этот слот маркирован P/S, CPU. Модули ввода/вывода могут устанавливаться в любые оставшиеся слоты. Для обеспечения правильной работы системы необходимо, чтобы все слоты были заполнены. Чтобы заполнить пустые слоты в каркасе, вы можете использовать модули — заглушки.

Каркас крепится к панели с оборудованием или к машине с помощью четырех винтов M4 по углам каркаса, как показано на рисунке справа. Монтажные вырезы позволяют снимать каркас после его установки без полного удаления монтажных винтов. Необходимые монтажные размеры приведены в предыдущем разделе.а



Выбор типа каркаса

Каркасы выбираются из двух типов. Стандартный тип каркаса имеет ограничения на размещение специальных (или интеллектуальных) модулей в локальном каркасе вместе с процессором. Если Вы используете процессор DL450 и новый каркас с "расширенной шиной", то Вы можете применять специальные модули в каркасах с блоком расширения, как показано на рисунке справа. Если все каркасы в системе являются каркасами нового типа, то DL450 может взаимодействовать со специальными модулями в любом каркасе. Во всех остальных отношениях новый каркас является точной копией стандартного каркаса.



Номера стандартных и новых каркасов приведены ниже.

Стандартные каркасы

D4-04B

D4-06B

D4-08B

Каркасы с "расширенной шиной"

D4-04B-1

D4-06B-1

D4-08B-1

Позволяют использовать специальные модули в каркасах с блоком расширения

Разъемы блоков расширения в новых каркасах имеют новые сигналы данных, используемые для взаимодействия со специальными модулями ввода/вывода, установленными в этих каркасах. Поэтому Вы должны соблюдать следующие ограничения и указания при использовании новых каркасов:

- Только процессор типа DL450 (в локальном каркасе) может взаимодействовать со специальным модулем, установленным в каркасе расширения.
- В данном случае локальный каркас, как и каркас расширения должны быть нового типа(-1).
- Конечно, Вы можете иметь еще специальные модули в локальном каркасе.
- Новые каркасы могут использоваться также и с процессорами DL430 и DL440 (однако эти процессоры не могут взаимодействовать со специальными модулями в этих каркасах).
- Вы можете сочетать стандартные и новые каркасы в вашей системе, но в этом случае специальные модули ввода/вывода не могут применяться в каркасах расширения (стандартные каркасы не могут передавать сигналы для специальных модулей ввода/вывода через разъемы каркасов расширения).

ПРИМЕЧАНИЕ. Если Вы разрабатываете новое приложение на базе процессора DL450, то мы рекомендуем использовать новые каркасы (типа -1) с тем, чтобы Вы позже могли добавлять специальные модули в любой каркас.

Установка модулей в каркас

Установка DIP -переключателей процессора (Только в DL430/440)

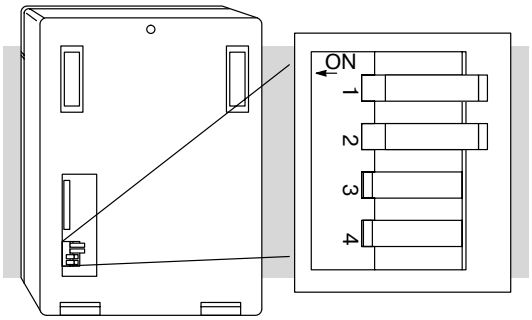
На задней панели процессоров DL430 и DL440 расположен блок из четырех переключателей конфигурации. С помощью этих переключателей можно устанавливать определение низкого напряжения на батарее, заменять адрес станции, устанавливать скорость передачи в бодах вторичного порта (с 25-контактным разъемом типа D). На рисунке ниже показано размещение этих DIP -переключателей. Эквивалентные функции в процессоре DL450 требуют применения вспомогательных AUX-функций с устройства программирования.

Переключатель 1

- ON = Индикатор низкого напряжения на батарее заблокирован
- OFF = Индикатор низкого напряжения на батарее включен

Переключатель 2

- ON = Ручная корректировка адреса станции разрешена (адрес 1)
- OFF = Адрес станции устанавливается с помощью AUX-функции с устройства программирования.

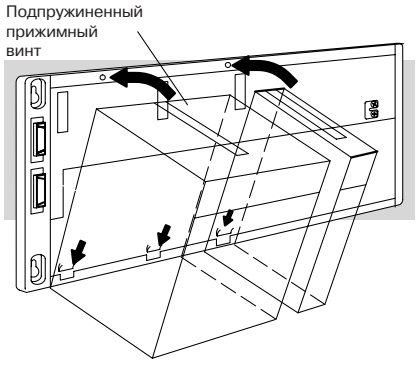


ПРИМЕЧАНИЕ. Установка переключателя 2 в положение ON переводит адрес станции в значение ”1”. При этом адрес, установленный с помощью устройства программирования, не изменяется. Если снова перевести переключатель 2 в положение OFF, то адрес, установленный с помощью AUX — функции и хранимый в памяти, будет возвращен.

Скорость передачи в бодах порта 1	Переключатель 3	Переключатель 4
300	Off	Off
1200	Off	On
9600	On	Off
19200	On	On

ПРИМЕЧАНИЕ. Четность, режим и адрес станции для порта 2 устанавливаются с помощью AUX - функции с устройства программирования.

1. Заметьте, что компоненты имеют пластиковые выступы сзади внизу и прижимный винт наверху.
2. При слегка наклоненном вперед устройстве вставьте выступы в пазы каркаса.
3. Затем мягко толкайте верхнюю часть компонента, пока он не будет прочно установлен в каркасе.
4. Далее затяните винт в верхней части компонента, чтобы закрепить его в каркасе.

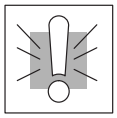


ПРЕДУПРЕЖДЕНИЕ. Чтобы минимизировать риск электрического удара, поражения персонала и повреждения аппаратуры, всегда отключайте питание системы перед установкой или снятием любого компонента системы.

Указания по монтажу процессора и блока расширения

Клеммы подключения сетевого питания расположены на передней панели процессоров DL405 и блоков расширения (под крышкой). Ниже дается описание каждого клеммного зажима. Большинство клеммных зажимов одинаковы и в процессоре, и в блоке расширения. Если клеммные зажимы относятся только к одному устройству, то это отмечается при описании.

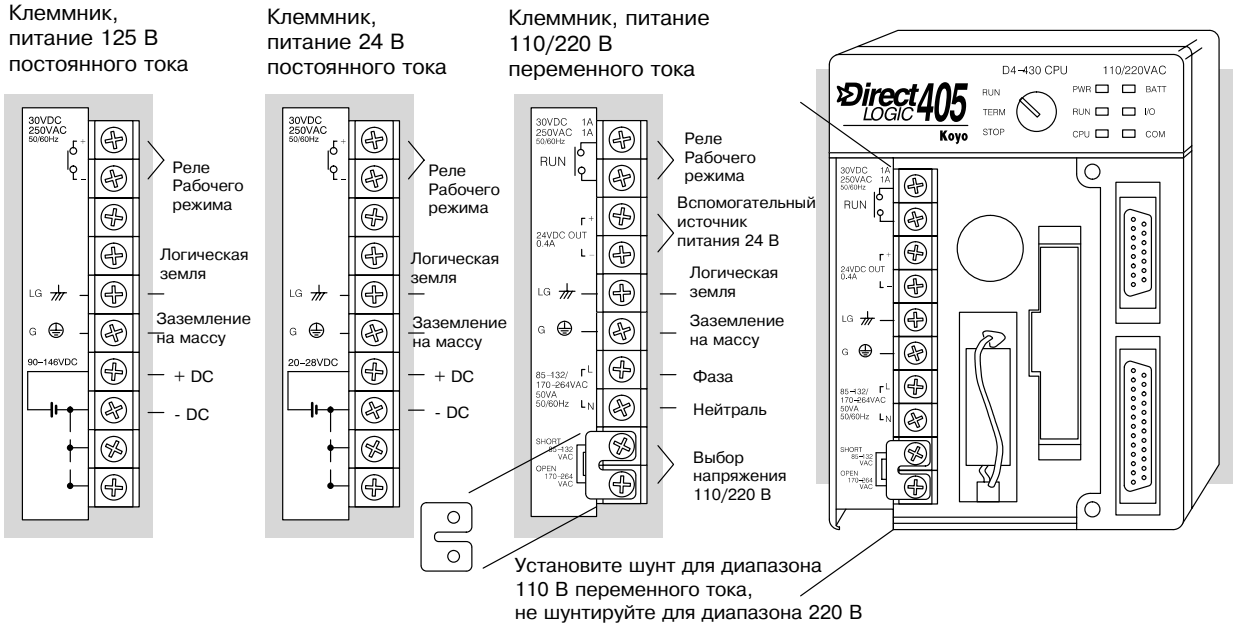
- Реле рабочего режима — (только в процессоре) указывает внешнему устройству, что процессор находится в рабочем режиме (при замкнутых контактах реле). Эти нормально открытые контакты могут также отключать питание критических точек ввода/вывода, когда процессор выходит из рабочего режима.
- Вспомогательное питание 24 В постоянного тока — может использоваться для питания полевых устройств или модулей ввода/вывода, для которых необходим внешний источник питания. Он обеспечивает ток до 400 мА при 20 -28 В постоянного тока с пульсацией менее 1 В P-P. (Не доступно для процессоров с питанием от постоянного тока).
- Логическая земля — внутренняя ”земля” в системе, которая может быть привязана к полевым устройствам / коммуникационным портам, чтобы объединить все сигналы заземления.
- Заземление на массу — здесь ”земля” подключается к устройству.
- Питание переменного тока — здесь фаза (с напряжением) и нейтраль (общий провод) подключаются к процессору/блоку расширения. (Сюда также подсоединяется источник питания постоянного тока для процессора с питанием 24/125 В постоянного тока. Положительный полюс подсоединяется к фазе, отрицательный — к заземлению).
- Выбор напряжения 110/220 В — шунтирование этих двух клемм определяет выбор напряжения. Установите шунт при выборе входного напряжения 110 В, удалите шунт при выборе входного напряжения 220 В (шунт не требуется для процессоров или блоков расширения с питанием от постоянного тока).



ПРЕДУПРЕЖДЕНИЕ. Может произойти повреждение источника питания, если 220 В переменного тока будет поведено к клеммам, когда установлен шунт на 110 В. Сразу после подключения питания установите предохранительную крышку, чтобы избежать риска случайного электрического удара.

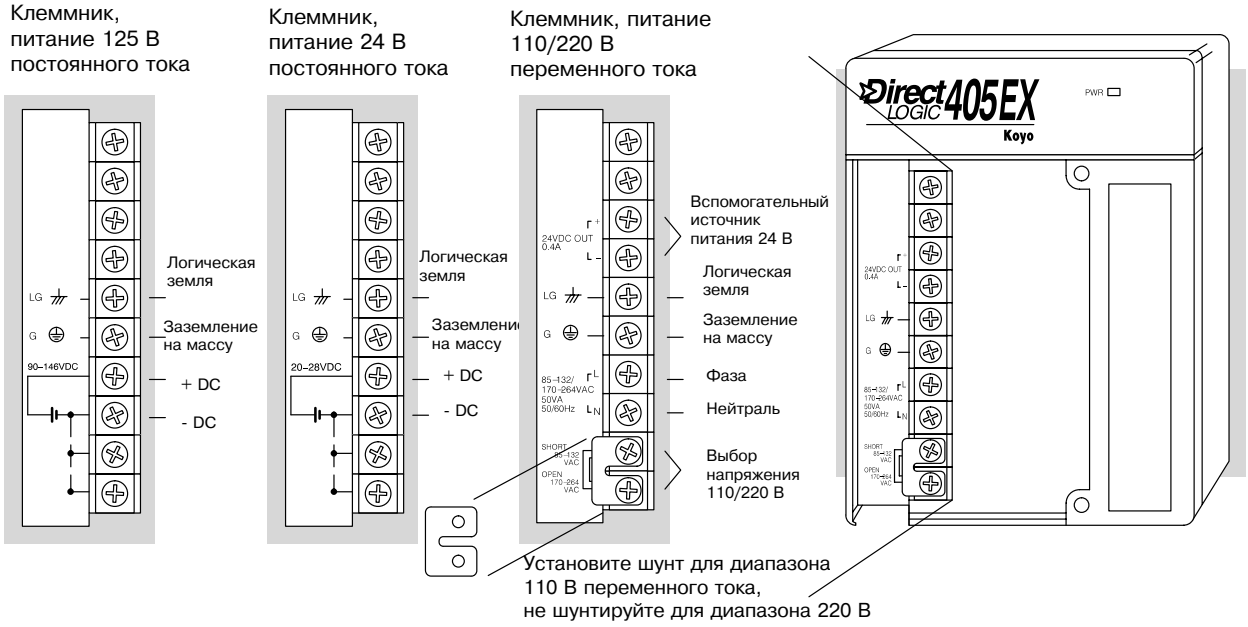
Монтаж процессора

На следующих рисунках детализируются соответствующие подключения к каждой клемме.



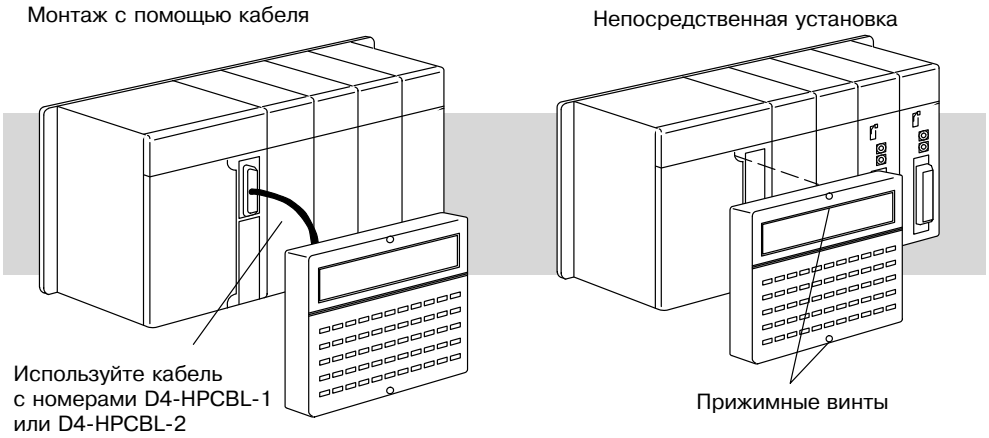
Монтаж блока расширения

На следующих рисунках детализируются соответствующие подключения к каждой клемме.

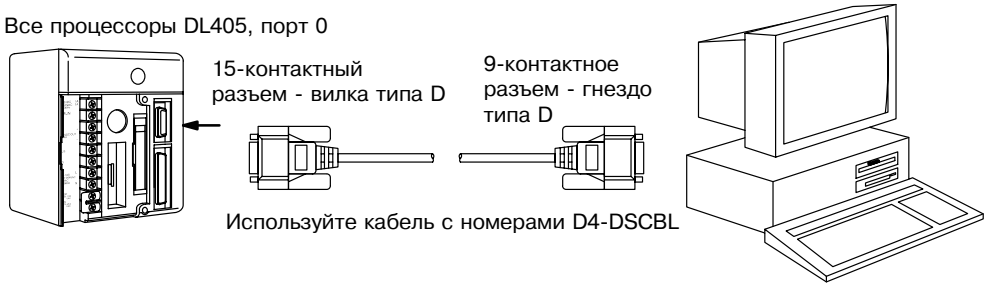


Подсоединение устройств для программирования

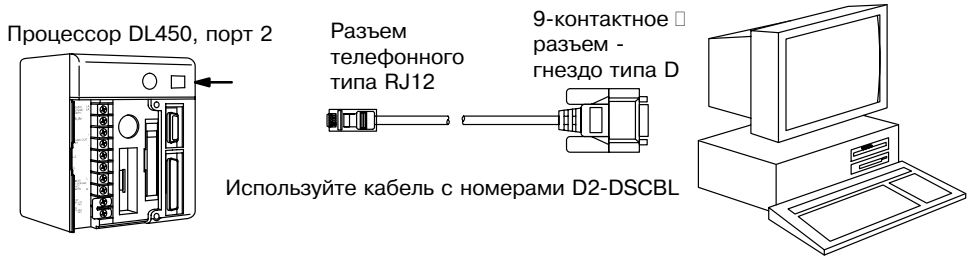
Вы можете установить ручной программатор непосредственно в Порту 0 какого-либо процессора DL405 (с помощью 15-контактного разъема типа D) или Вы можете использовать для подсоединения кабель 3 м или кабель 1.5 м, как показано ниже.



Для подсоединения устройств программирования DirectSOFT стандартным портом для всех процессоров DL405 служит 15-контактный порт 0. Показанный на рисунке ниже кабель имеет длину примерно 3.66 м.

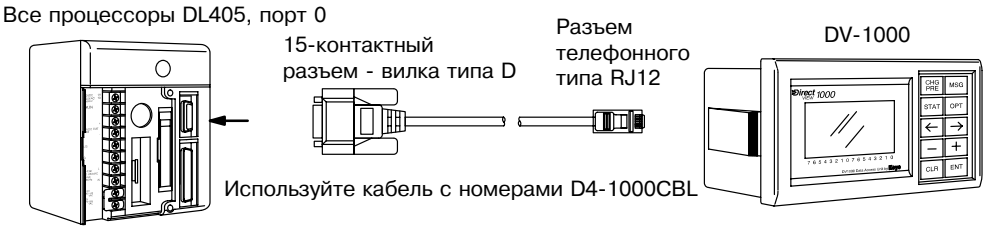


В процессоре DL450 Вы, кроме того, можете использовать для подсоединения устройств программирования DirectSOFT порт 2. Показанный на рисунке ниже кабель имеет длину примерно 3.66 м.

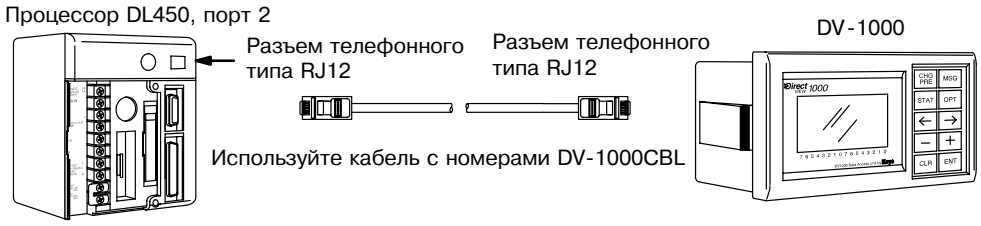


Подсоединение устройств с интерфейсом оператора

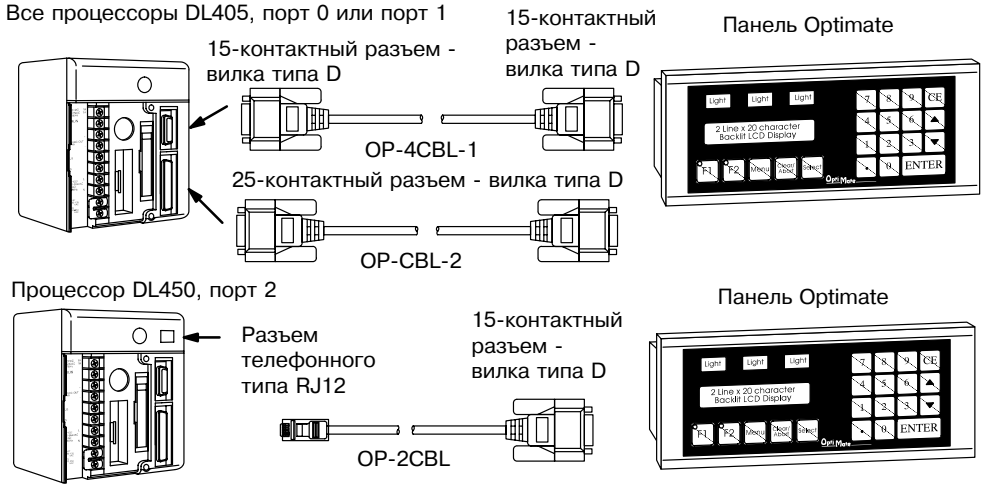
Интерфейсы оператора обычно требуют соединений для передачи данных и для подключения питания. Однако Блок Доступа к Данным DV-1000 может получать данные и питание непосредственно с какого-либо процессора DL405 при использовании кабеля длиной 2 м, как показано ниже.



Процессор DL450 можно подсоединить к DV-1000 через порт 2, используя показанный ниже кабель длиной примерно 2 м.



Панели с интерфейсом оператора Optimate требует отдельных соединений для данных и для питания. Подсоедините порт 0, порт 1 или порт 2 (DL450) к панели Optimate, выбрав соответствующий кабель длиной 2 м из показанных ниже трех кабелей.

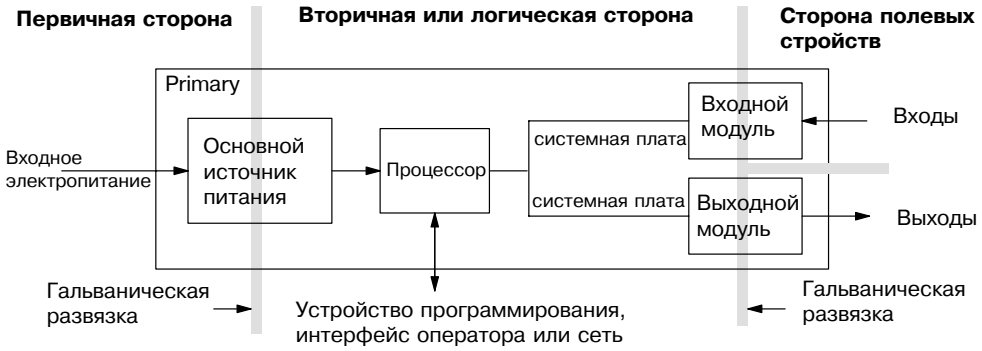


Стратегии монтажа цепей ввода/вывода

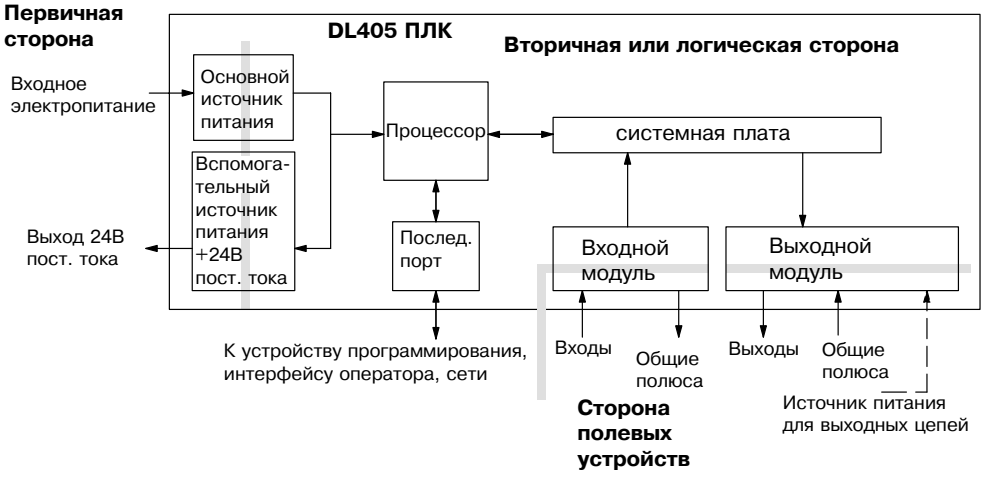
Гальваническая развязка ПЛК

Система DL405 ПЛК весьма гибкая система, она может работать при различных конфигурациях соединений. Изучив данный раздел перед установкой, вы, возможно, найдете наилучшую стратегию коммутации для вашего приложения. Это позволит уменьшить стоимость системы, сократить ошибки при монтаже и избежать проблем безопасности.

Цепи ПЛК разделены на три основные гальванически изолированных зоны, показанные на приведенных ниже рисунках. Электрическая изоляция обеспечивает безопасность, когда отказ в одной зоне не приводит к нарушениям в другой. Трансформатор в системе электропитания обеспечивает магнитную изоляцию между первичной и вторичной сторонами. Оптопары обеспечивают оптическую изоляцию во Входной и Выходной цепях. Это отделяет логические цепи от стороны полевых устройств, к которой подсоединены производственные механизмы. Следует отметить, что дискретные входы отделены от дискретных выходов, поскольку каждый из них отделен от логической стороны. Гальваническая изоляция защищает интерфейс оператора (и самого оператора) от сбоев во входном электропитании или от сбоев в производственных схемах. При монтаже ПЛК чрезвычайно важно исключить создание внешних связей, которые соединяют цепи логической стороны с любой другой.



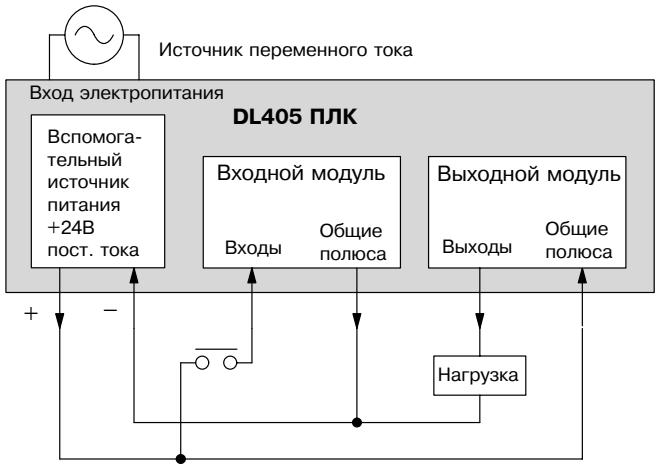
На следующем рисунке показана физическая компоновка системы ПЛК DL405, вид спереди. Дополнительно к базовым цепям, рассмотренным выше, каркасы с питанием от переменного тока включают вспомогательный источник питания 24 В постоянного тока с собственной гальванической развязкой. Поскольку выход этого источника питания изолирован от других трех цепей, он может использоваться для питания входных и/или выходных цепей!



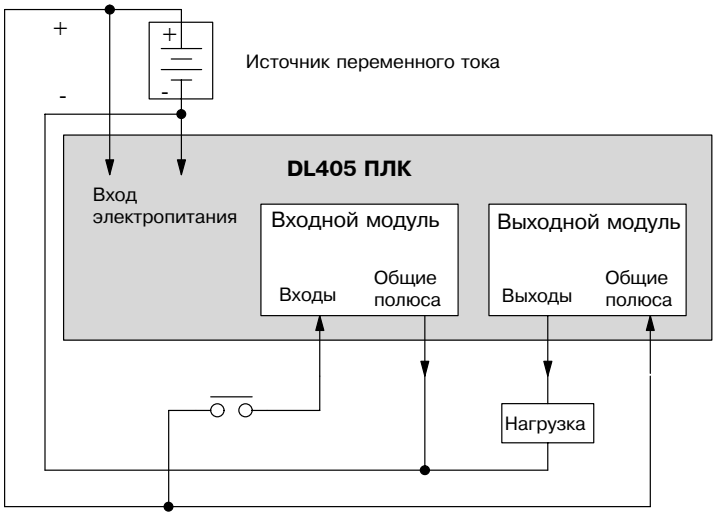
В некоторых случаях использование встроенного вспомогательного источника 24 В постоянного тока может привести к снижению стоимости вашей системы управления. Он может питать комбинированные нагрузки током до 400 мА. Будьте внимательны, не превышайте токовый номинал источника питания. Если вы — системный разработчик вашего приложения, то вы можете выбрать и запроектировать полевые устройства, которые могут использовать вспомогательный источник питания 24В постоянного тока.

Электропитание цепей ввода/вывода от вспомогательного источника

Характерной чертой всех каркасов DL405 с питанием от переменного тока является наличие вспомогательного внутреннего источника питания. Если входные устройства и выходная нагрузка требуют питания 24 В постоянного тока, то вспомогательный источник может обеспечить обе эти цепи, как показано на приведенном ниже рисунке (максимум 400 мА).



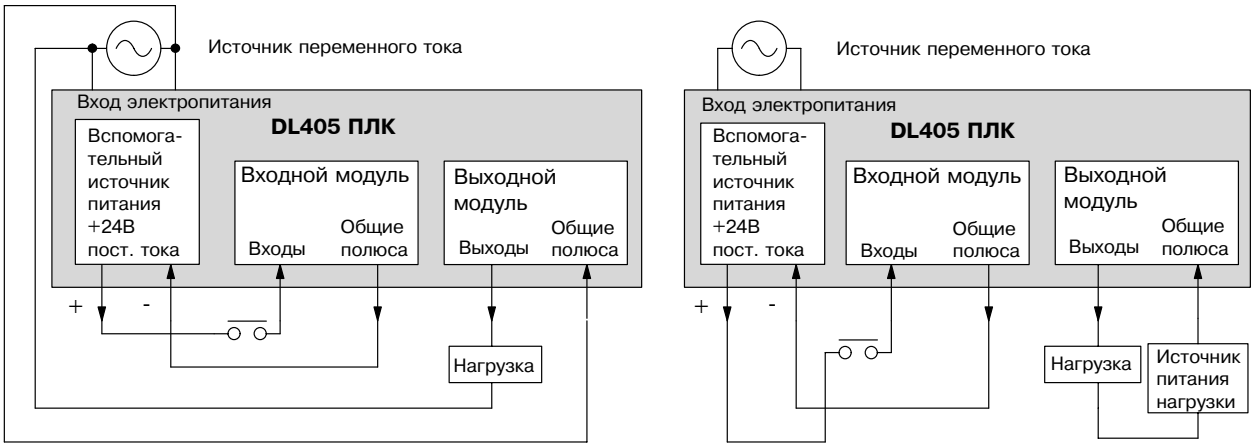
Каркасы DL405 с питанием от постоянного тока проектируется для такой среды приложений, в которой низковольтное электропитание постоянного тока более доступно, чем питание переменного тока. Эта среда включает широкий диапазон приложений с питанием от батарей, например, дистанционное управление, средства передвижения, переносные механизмы и др. В таких типах приложений для входных устройств и выходной нагрузки обычно используется один и тот же источник питания постоянного тока. Типовая схема соединений для приложений с питанием от постоянного тока показана на следующем рисунке.



Электропитание цепей ввода/вывода от отдельных источников

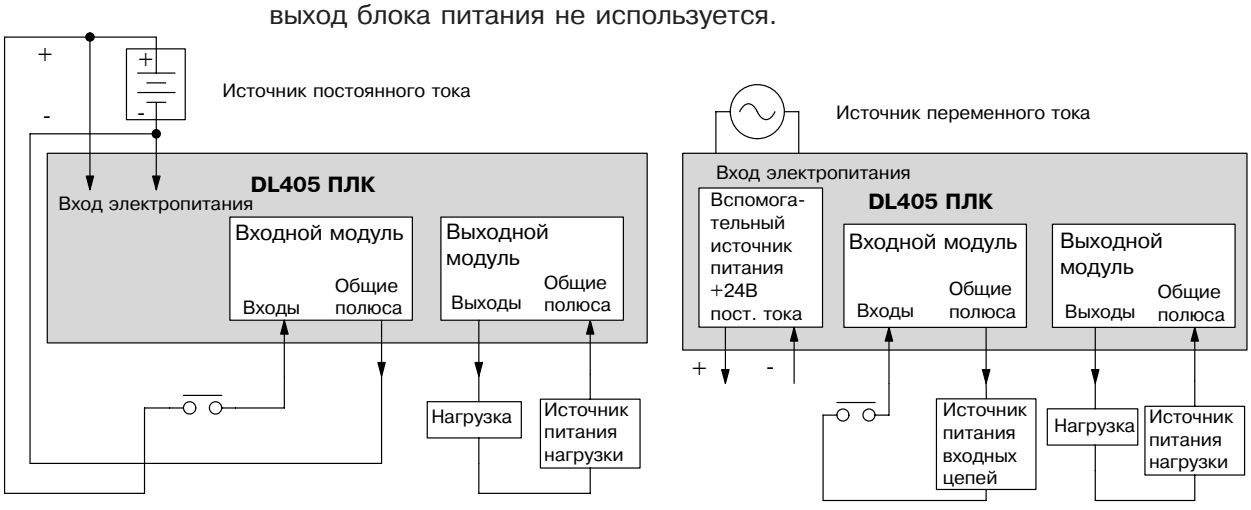
В большинстве приложений входные устройства необходимо снабжать электроэнергией от одного источника, выходную нагрузку — от другого. Нагрузка часто требует высоковольтного питания переменного тока, в то время как для чувствительных элементов на входе можно использовать постоянный ток с более низким напряжением. Оператор механизмов случайно может прикоснуться к входным клеммам, поэтому условия безопасности также требуют изоляции от высоковольтных выходных цепей. Наиболее удобным является случай, когда для нагрузки можно использовать тот же источник питания, что и для ПЛК, а для чувствительных элементов на входе — использовать вспомогательное электропитание, как показано на левом рисунке ниже.

Когда нагрузка не может питаться от источника питания ПЛК, тогда необходимо использовать отдельный источник питания, как показано на правом рисунке ниже.



В некоторых приложениях можно использовать внешний источник питания контроллера для электропитания входной цепи. Это обычно имеет место для ПЛК с блоком питания постоянного тока, как показано на левом рисунке ниже. Входные модули совместно с ПЛК используют один источник питания, а выходы имеют собственный источник.

Наихудшим вариантом по стоимости и сложности является приложение, в котором требуются отдельные источники питания для ПЛК, входных устройств и выходной нагрузки. Изображенный на правом рисунке ниже пример соединений показывает, как работает эта схема, вспомогательный выход блока питания не используется.



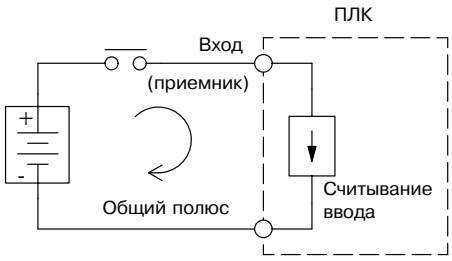
Понятия приемник/ источник тока

Перед дальнейшим изучением схем подключения вы должны иметь полное понимание понятий ”приемник” и ”источник” тока. Эти термины часто используются при описании входных и выходных цепей. Все цепи ввода/вывода дискретных модулей DL405 пассивны и требуют внешних источников для поддержания в них сигнального тока. Поэтому понятия «цепь-источник» и «цепь-приемник» условны и относятся только к направлению тока в сигнальном проводе цепи:

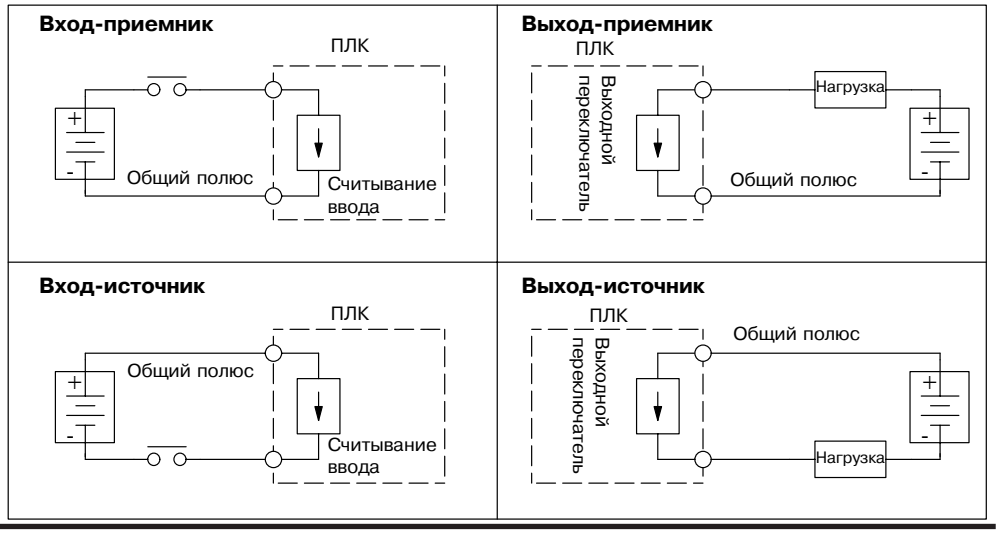
- направление «в модуль» характеризует «цепь-приемник»
- направление «из модуля» характеризует «цепь-источник»

Необходимо отметить, что эти понятия относятся только к цепям постоянного тока, так как имеется указание полярности (+) и (-). Следовательно, терминология Приемник и Источник применима только к входным и выходным цепям постоянного тока. Входные и выходные точки, которые являются приемниками либо источниками, могут проводить ток только в одном направлении. Это означает, что можно подсоединить внешний источник питания и полевые устройства к точке ввода/вывода с обратным направлением тока, и эти цепи не будут работать. Таким образом, вы можете приступить к монтажу только при понимании понятий ”Приемник” и ”Источник”.

Например, на рисунке справа показан вход — ”приемник”. Для правильного подсоединения внешнего источника питания, вы должны осуществить это соединение таким образом, чтобы вход обеспечивал проводимость к заземлению (-). Начните с входной клеммы ПЛК, продолжайте через цепи считывания входа ПЛК и закончите на общем полюсе, соедините источник питания (-) с общим полюсом. После добавления выключателя между источником питания (+) и входом вы закончите цепь. Если замкнуть выключатель, ток потечет в направлении, указанном стрелками.

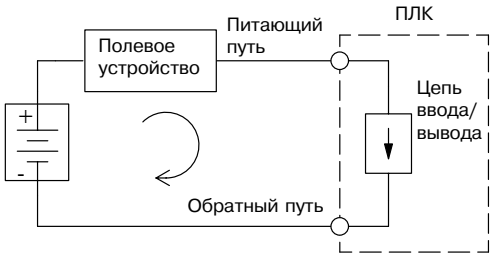


Применяя указанный принцип построения цепи, получим показанные ниже четыре возможных схемы входного/выходного приемника/источника. В конце данного раздела приводятся спецификации модулей ввода/вывода для перечисленных типов входов и выходов.

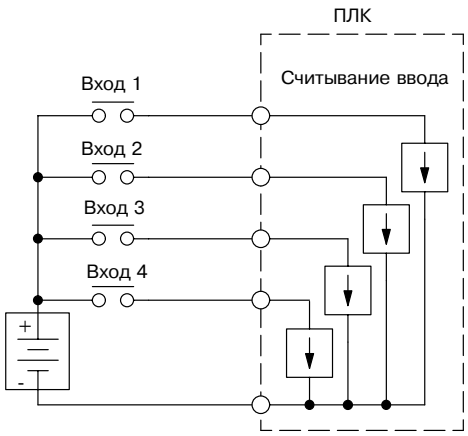


Понятия "Общего полюса" Входа/Выхода

Для того чтобы цепи ввода/вывода работали, необходимо, чтобы ток входил в одну клемму и выходил через другую. Поэтому с каждой точкой ввода/вывода связаны, по крайней мере, две клеммы. На рисунке справа клемма для входа или выхода предназначена для питающего провода тока. Еще одна клемма необходима для обратного провода (к источнику питания).

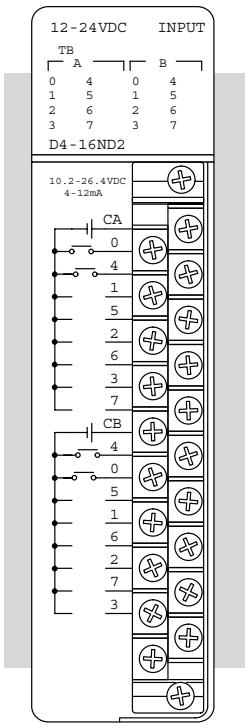


Каждая точка ввода/вывода должна иметь две выделенных клеммы, как показано на рисунке выше. Однако обеспечение такого уровня гибкости не практично и даже не является необходимым для большинства приложений. Так, большая часть точек ввода или вывода ПЛК объединена в группы, в которой совместно используется обратный провод (называемый общим). На рисунке справа показана группа (или блок) из 4 входных точек, которые совместно используют обратный общий провод. В этом случае четыре входа требуют только пять клемм вместо восьми.



ПРИМЕЧАНИЕ. В приведенных цепях ток в общем проводе в 4 раза больше любого канального входного тока, когда все входы подключены. Это особенно важно для выходных цепей, где иногда требуется более мощный общий провод.

Большинство входных и выходных модулей DL405 объединяют их точки ввода/вывода в блоки, которые совместно используют обратный провод. Хорошим указателем такой группы ввода/вывода являются монтажные метки, показанные на рисунке справа. Миниатюрная схема показывает два блока цепей по восемь входных точек в каждом. Общая клемма для каждого из них помечена соответственно "CA" и "CB".



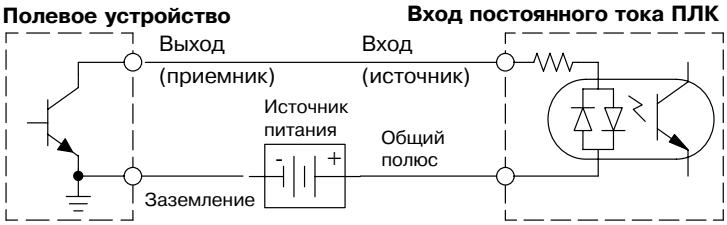
В данном примере положительный вывод источника питания постоянного тока соединяется с общими клеммами. Обозначения, принятые в схемах соединений, приведены ниже.

Соединение точек ввода/вывода постоянного тока с "полупроводниковыми" полевыми устройствами

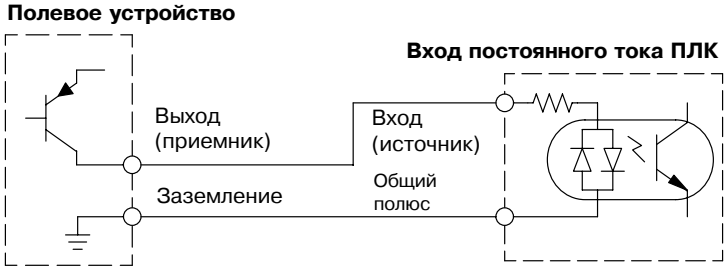
В предыдущем разделе понятия Приемник и Источник цепи ввода/вывода постоянного тока объяснялись возможностью тока течь только в одном направлении. Это справедливо также для многих рабочих устройств, которые имеют полупроводниковую (транзисторную) аппаратуру сопряжения. Другими словами, рабочие устройства также могут быть "приемниками" и "источниками". Когда два устройства соединяются последовательно в цепи постоянного тока, то один из них должен соединяться как "источник", а другой — как "приемник" тока.

Полупроводниковые входные чувствительные элементы

Некоторые входные модули постоянного тока DL405 могут принимать ток любого направления, поэтому они могут подсоединяться либо как "источники", либо как "приемники". В следующей цепи полевое устройство имеет выход с транзистора с открытым коллектором NPN. К нему подается ток от входной точки ПЛК, которая является "источником". Питание может подаваться от вспомогательного источника +24 В, либо от любого другого источника (+12 или +24 В постоянного тока), который удовлетворяет техническим условиям.



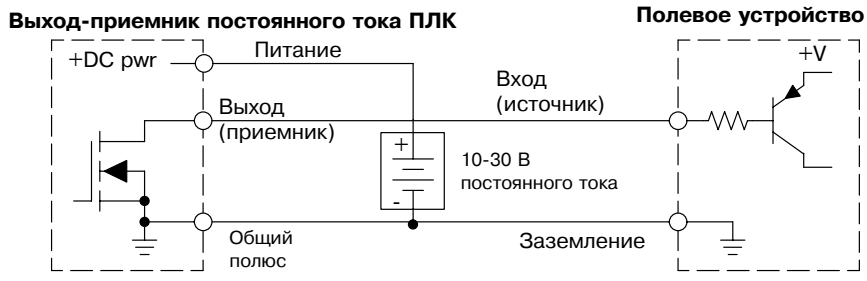
В следующей цепи полевое устройство имеет выход с транзистора с открытым эмиттером. С него ток подается на входной модуль ПЛК, с которого — на заземление. Поскольку полевое устройство является источником тока, то никакие дополнительные источники питания не требуются.



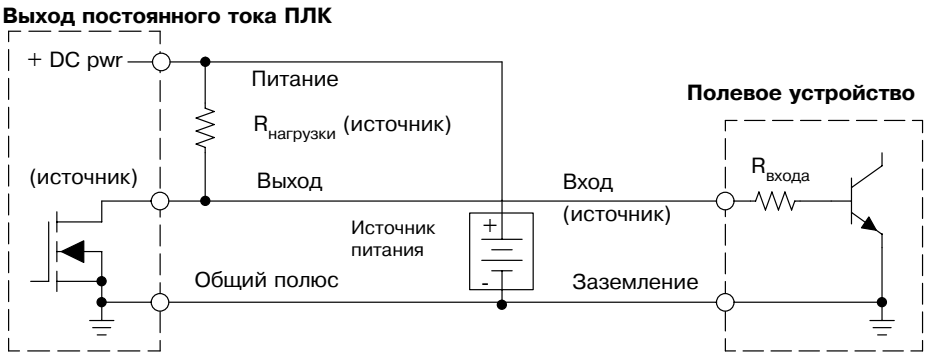
Полупроводниковая выходная нагрузка

Иногда в приложении требуется соединить выходную точку ПЛК с полупроводниковым входом полевого устройства. Этот тип соединения обычно используется для передачи управляющего сигнала низкого уровня, не посылая питание постоянного тока на исполнительный механизм.

Некоторые выходные модули постоянного тока ПЛК относятся к типу "Приемник". Это означает, что каждый такой выходной модуль при подключении питания будет проводить ток к заземлению. В следующей цепи от выходной точки ПЛК при подключении питания ток подается на общий полюс выходного модуля. Эта выходная точка соединена с входом-источником полевого устройства.



В следующем примере выходная точка-приемник ПЛК постоянного тока соединена с входом-приемником полевого устройства. Это — небольшая хитрость, так как и выход ПЛК, и вход полевого устройства имеют тип "приемник". Поскольку цепь должна иметь одно устройство-источник и одно устройство-приемник, то к выходу ПЛК необходимо добавить возможности источника посредством использования нагрузочного резистора. В цепи, показанной ниже, Rнагрузки соединяет выходную точку с входом цепи питания выходного модуля постоянного тока.



ПРИМЕЧАНИЕ 1. НЕ пытайтесь использовать большую нагрузку (>25 мА) в этой схеме подключения нагрузки.

ПРИМЕЧАНИЕ 2. Применение нагрузочного резистора для реализации выхода-источника имеет своим результатом инвертирование логики выходной точки. Другими словами, с точки зрения многоступенчатой логики, на вход рабочего устройства подается питание, когда выход ПЛК отключен. Ваша программа должна учитывать это и вырабатывать инвертированный выход. Или вы можете отменить инверсию везде, например, на полевом устройстве.

Важно правильно выбрать значение Rнагрузки. Для этого вам необходимо знать номинальный входной ток полевого устройства (Iвхода) при его включении. Если он не известен, то его можно рассчитать, как показано ниже (типичное его значение 15мА). Далее используйте (Iвхода) и напряжение внешнего источника питания для вычисления Rнагрузки. Затем вычислите мощность $P_{нагрузки}$ (в ваттах) для оценки правильного значения $R_{нагрузки}$.

$$I_{входа} = \frac{V_{входа\ (при\ включении)}}{R_{входа}}$$

$$R_{нагрузки} = \frac{V_{истояника} - 0.7}{I_{входа}} - R_{входа} \qquad P_{нагрузки} = \frac{V_{истояника}^{2}}{R_{нагрузки}}$$

Конечно, наиболее простой способ подвести ток на вход-приемник рабочего устройства, как показано ниже, состоит в использовании выходного модуля — источника постоянного тока. Каскад NPN Дарлингтона может дать около 1.5 В при насыщении после включения, но этого достаточно для слаботочной полупроводниковой нагрузки.

Руководство по релейному выходу

Четыре выходных модуля в серии модулей ввода/вывода DL405 характеризуются следующими релейными выходами: D4-08TR, F4-08TRS-1, F4-08TRS-2, D4-16TR. Реле являются наилучшим решением для следующих приложений:

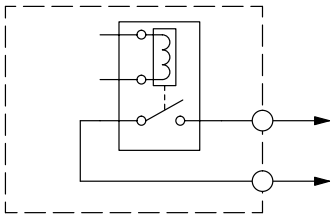
- При нагрузке, для которой требуется более высокий ток, чем может дать полупроводниковый выход.
- В приложениях, для которых важна стоимость.
- Когда некоторые выходные каналы требуют изоляции от других выходов (например, когда некоторые нагрузки требуют напряжений, отличных от других нагрузок).

Некоторые приложения, в которых НЕЛЬЗЯ использовать реле:

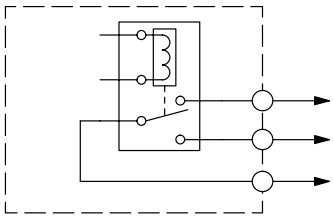
- Нагрузки, которые требуют ток менее 10мА.
- Нагрузки, которые должны переключаться с высокой скоростью либо в цикле с тяжелым режимом.

Релейные выходы выходных модулей DL405 имеют контакты двух типов, как показано справа. Тип A или тип SPST (один полюс, один ход) является нормально открытым, он наиболее прост в применении. Тип C или тип SPDT (один полюс, два направления) является переключающим. Он обеспечивает как нормально закрытый контакт, так и нормально открытый контакт.

Реле с контактами типа A



Реле с контактами типа C

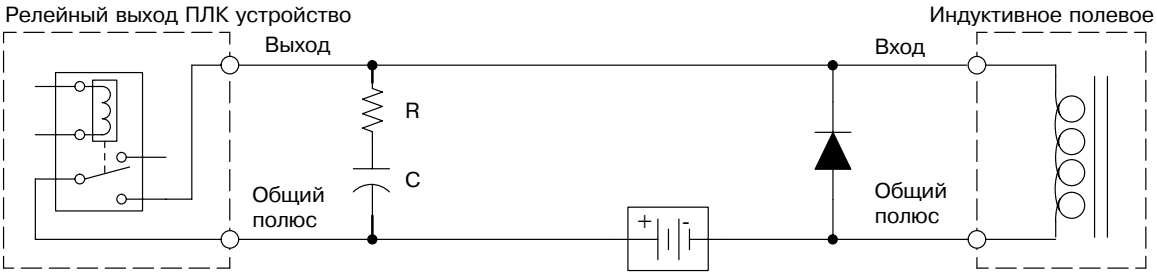


Некоторые реле модулей с релейным выходом совместно используют общие клеммы, которые соединены с подвижным контактом каждого реле блока. В других релейных модулях реле полностью изолированы друг от друга. Во всех случаях модуль возбуждает обмотку реле, когда включается соответствующая выходная точка.

Продление срока службы контактов реле

Контакты реле изнашиваются в зависимости от числа переключений реле, количества искровых разрядов, создаваемых при его открытии и закрытии, и от присутствия примесей в воздухе. Однако существуют некоторые действия, позволяющие продлить срок службы контактов реле:

- Включение и выключение реле должно происходить только тогда, когда этого требует приложение.
- Если вы имеете варианты, то включение или отключение нагрузки необходимо проводить в тот момент, когда она потребляет наименьший ток.
- Примите меры по подавлению пиков индуктивного напряжения от индуктивной нагрузки постоянного тока, такой как контакторы и соленоиды (соответствующая цепь показана ниже).



Добавляя внешнюю защиту контактов, можно увеличить срок их службы по сравнению с данными (числом циклов), которые приведены в спецификации релейных модулей. Индуктивные нагрузки с большим током такие, как муфты сцепления, тормоза, двигатели, клапаны с соленоидами прямого действия и стартеры двигателей, в наибольшей степени выиграют от внешней защиты контактов.
Рядом с выходными разъемами релейного модуля необходимо установить RC-схемы. Для определения значений для сглаживающей RC-схемы сначала найдите напряжение на контактах при открытии и ток на них при закрытии. Если питание нагрузки производится от переменного тока, то преобразуйте ток и напряжение к их пиковым значениям.
Значения R и C можно рассчитать по следующим формулам:

$$C\ (\mu F) = \frac{I^2}{10}$$

$$R\ (\Omega) = \frac{V}{10 \times I^x} \qquad \text{где } x = 1 + \frac{50}{V}$$

C минимум = 0.001 μF, номинальное напряжение на C должно быть ≥ V, не поляризованное.
R минимум = 0.5Ω (Ом), 1/2 ватта, точность равна ± 5%.
Например, пусть контакты реле управляют нагрузкой при 120 В переменного тока, 0.5 А. Поскольку в данном примере используется питание от переменного тока, то сначала надо вычислить пиковые значения:

$$I_{пик} = I_{ср.квадр.} \times 1.414 = 0.5 \times 1.414 = 0.707 \text{ Ампер}$$

$$V_{пик} = V_{ср.квадр.} \times 1.414 = 120 \times 1.414 = 169.7 \text{ Вольт}$$

Далее находятся значения R и C:

$$C\ (\mu F) = \frac{I^2}{10} = \frac{0.707^2}{10} = 0.05\ \mu F, \text{ номинальное напряжение} \geq 170 \text{ Вольт}$$

$$R\ (\Omega) = \frac{V}{10 \times I^x} \qquad \text{где } x = 1 + \frac{50}{V}$$

$$x = 1 + \frac{50}{169.7} = 1.29$$

$$R\ (\Omega) = \frac{169.7}{10 \times 0.707^{\ 1.29}} = 26\ \Omega \text{ (Ом), 1/2 ватта, } \pm 5\%.$$

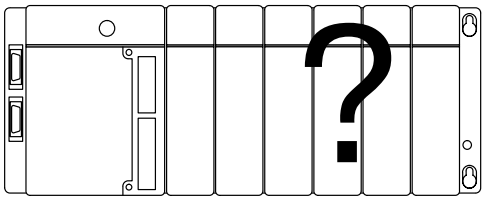
Если с помощью контактов переключается индуктивная нагрузка постоянного тока, то следует добавить диод параллельно нагрузке по возможности близко к обмотке нагрузки. Когда нагрузка включается, диод находится в непроводящем состоянии. Когда нагрузка отключается, то энергия, накопленная в ее обмотке, освобождается в виде отрицательного пика напряжения. В этот момент диод находится в проводящем состоянии и шунтирует энергию на заземление. Это защищает контакты реле от высоковольтной дуги, которая может иметь место при открытии контактов.
Для получения наилучших результатов следуйте следующим указаниям при использовании диода подавления помех:

- НЕ ИСПОЛЬЗУЙТЕ эту схему при источнике переменного тока.
- Устанавливайте диод по возможности близко к индуктивному полевому устройству.
- Используйте диод с номиналом, по крайней мере, 100 PIV (пиковых инверсных значений напряжения), 3А и более для прямого тока. Используйте тип диода с быстрым восстановлением (например, тип диода Шотки). НЕ используйте диоды для слабых сигналов, такие как 1N914, 1N941 и др.
- Убедитесь перед работой, что диод находится в правильном положении, Если он установлен в обратном направлении, то он замкнет источник питания при включении реле.

Монтаж и спецификации модулей ввода/вывода

Размещение модулей

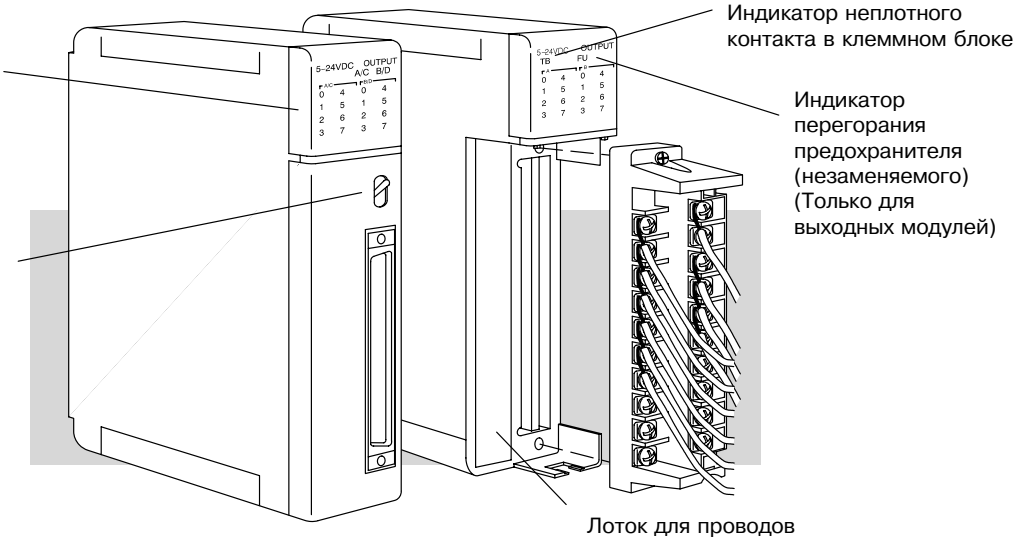
Перед подключением модулей ввода/вывода к полевым устройствам вашей системы необходимо убедиться в том, что каждый модуль ввода/вывода находится в надлежащем слоте и каркасе. Можно избежать дорогостоящих ошибок при монтаже, если выполнить следующее:



- Провести расчет потребляемой мощности для каждого каркаса, чтобы проверить, может ли блок питания каркаса обеспечить все модули каркаса. Информацию о том, как это делается, можно получить в главе 4 ”Проектирование и конфигурирование системы”.
- Некоторые специальные модули ввода/вывода могут устанавливаться только в конкретные слоты (в противном случае они не будут правильно работать). Поэтому перед их установкой и монтажом просмотрите соответствующие руководства.
- По мере возможности размещайте модули с высоким уровнем напряжения и тока подальше от чувствительных аналоговых модулей.

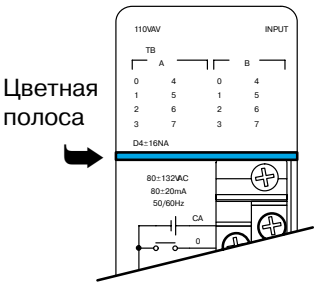
Индикаторы состояния модулей ввода/вывода

На рисунке ниже показано расположение индикаторов состояния для общих модулей ввода/вывода.



Цветовая кодировка модулей ввода/вывода

На лицевой панели модулей ввода/вывода семейства DL405 имеется полоса с цветовой кодировкой, которая помогает быстро определить тип модуля — либо это входной, либо выходной модуль либо специальный модуль. Схема цветовой кодировки указана ниже:

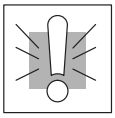


Тип модуля	**Цвет полосы**
Дискретный/Аналоговый Выход	Красный
Дискретный/Аналоговый Вход	Голубой
Другие	Белый

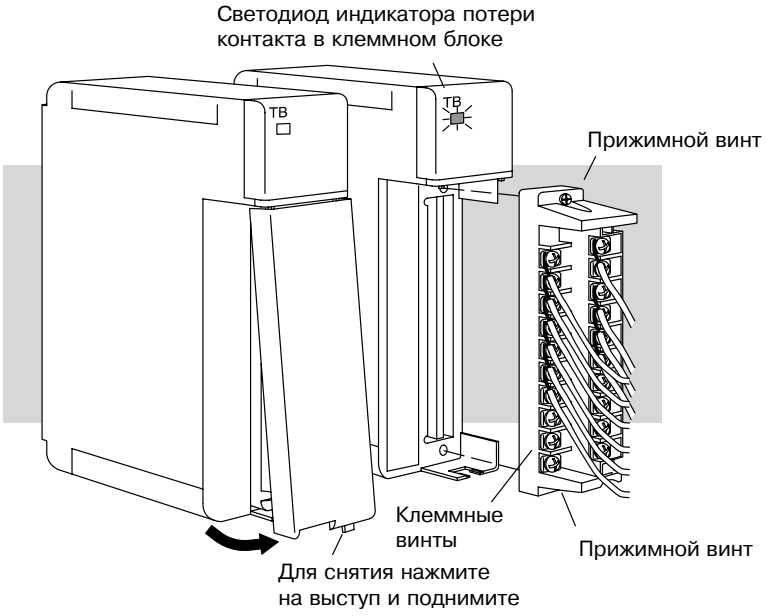
Монтаж модуля с клеммным блоком

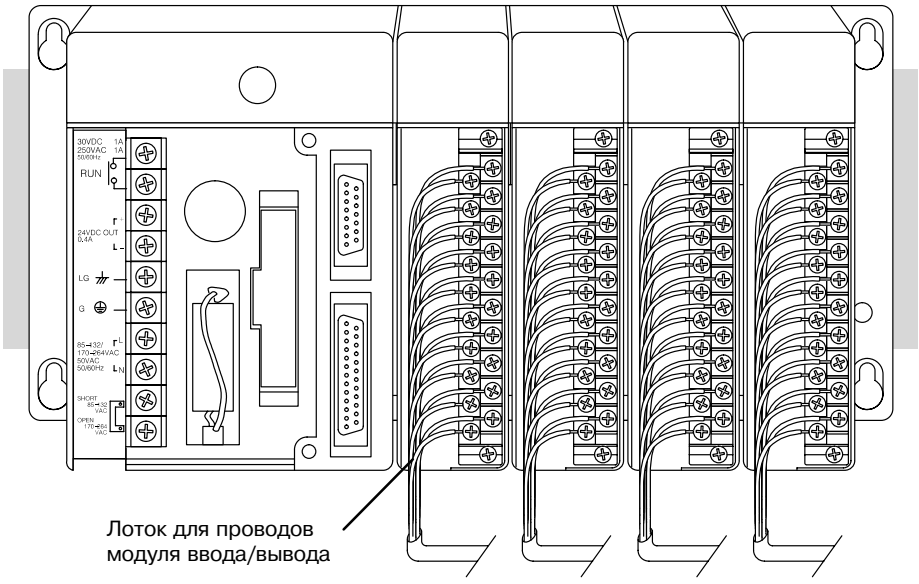
Перед монтажом сначала снимите переднюю крышку модуля. Чтобы снять крышку, нажмите на ее нижний выступ, поднимите ее нижний край и отсоедините крышку от модуля.

Для вашего удобства клеммные блоки всех модулей ввода/вывода DL405 сделаны съемными. Для снятия клеммного блока ослабьте прижимные винты и вытяните блок из модуля. При возвращении клеммного блока в модуль убедитесь, что он плотно установлен. Для уверенности затяните прижимные винты. Вы можете также проверить, что светодиод индикатора потери контакта в клеммном блоке остается отключенным при подаче питания в систему.



ПРЕДУПРЕЖДЕНИЕ. В некоторых модулях напряжение на полевых устройствах может присутствовать на контактах, даже когда система ПЛК уже отключена. Для минимизации риска электрического удара отключите питание всех полевых устройств перед тем, как снять разъем.



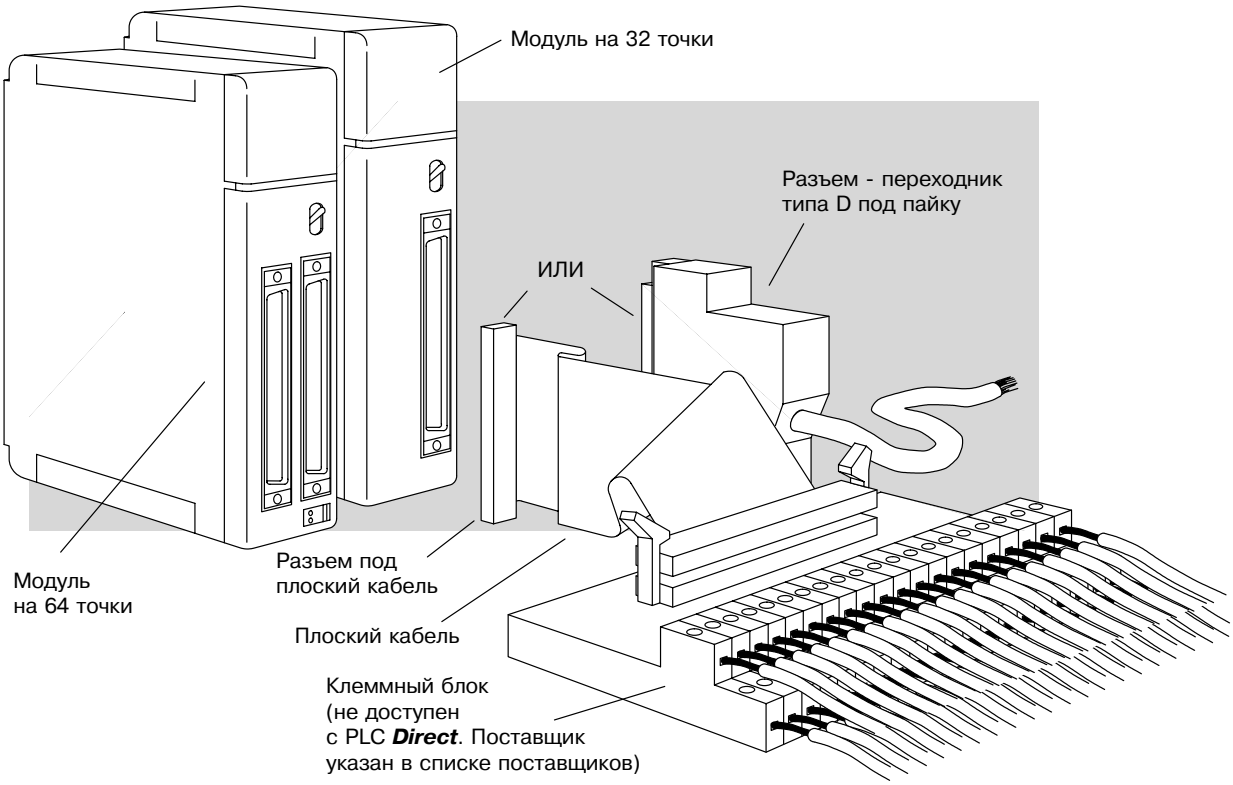


Монтаж модуля с использованием разъемов под плоский кабель/ под пайку

Из-за большого числа точек ввода/вывода в модулях на 32 и 64 точки используются другие типы разъемов. Для таких модулей предусмотрено два типа разъемов. Один — разъем типа D, требующий пайки. Другой — разъем под плоский кабель, который просто впрессован в плоский кабель.

Для модулей на 64 точки Вы должны использовать либо разъемы под плоский кабель, либо особые разъемы под пайку, специально разработанные для этих модулей. При заказе используйте следующие номера деталей: D4-IO3264S, который имеет в упаковке 2 разъема типа D под пайку; D4-IO32R, который имеет в упаковке 2 разъема под плоский кабель.

На следующем рисунке показаны примеры этих двух типов разъемов.



Номера деталей для разъемов модулей

Оба типа разъемов могут использоваться для соединения с PLCDirect. Их можно также применять для других устройств Fujitsu Microelectronic, Inc. При заказе этих разъемов используйте следующие номера деталей

*Номера деталей PLC**Direct***

- D4-IO3264R — Разъемы под плоский кабель, 2 в упаковке. Могут применяться для модулей либо на 32, либо на 64 точки.
- D4-IO3264S — Разъемы под пайку, 2 в упаковке. Могут применяться для модулей либо на 32, либо на 64 точки.

Номера деталей Fujitsu

- FCN-367J040-AU/F, или AG/F — Разъем под плоский кабель на 32 / 64 точки,
- FCN-361J040-AU, или AG — Разъем под пайку на 32 / 64 точки.

(В разъемах AU используется золотое покрытие сверх палладиевого. В разъемах AG используется серебряное покрытие.)

Поставщики деталей, используемых при компоновке клеммных блоков

Если Вы желаете применять клеммные блоки для модулей на 32 или 64 точки, то воспользуйтесь следующим списком поставщиков, которые могут обеспечить вас всеми деталями, необходимыми для указанной выше схемы компоновки, состоящей из плоского кабеля, разъема под плоский кабель и клеммного блока.

Vendors	
3M Electronic Products Division 6801 River Place Blvd. Austin, TX 78726-9000 800-225-5373	DuPont Connector Systems Barley Mill Plaza Wilmington, DE 19898-0019 800-237-2374
Augat/RDI 525 Randy Rd. Carol Stream, IL 60188 708-682-4100	Phoenix Contacts Products P.O. Box 4100 Harrisburg, PA 17111-0100 717-944-1300
AMP Incorporated P.O. Box 3608 Harrisburg, PA 17105-3608 717-564-0100	Thomas & Betts Electronics Div. 200 Executive Center Drive Greenville, SC 29616 803-676-2900
Cooper Industries, Belden Div. P.O. Box 1980 Richmond, IN 47375 317-983-5200	Weidmuller, Inc. 821 Southlake Blvd. Richmond, VA 23236 804-794-2877
Newark Electronics 4108 North Ravenswood A ve. Chicago, Il 60640 312-784-5100	(Newark Electronics is a distributor for all of the above product manufacturers except for Phoenix Contacts Products)

Плоский кабель

В следующей таблице перечислены кабели, которые могут использоваться для подсоединения клеммного блока к модулю ввода/вывода на 32 точки. Это — поливинилхлоридные 40 -жильные кабели с шагом 1.27 мм.

Описание/Тип	Поставщик	Номер детали
Серый / 0.13 мм^2	3M	3801 / 40
Серый / 0.13 мм^2	Belden	9L260 40
Серый / 0.1 мм^2	Belden	9L280 40
Серый / 0.1 мм^2	DuPont	76825-040
Серый / 0.1 мм^2	AMP	4991 16-5
С цветовой кодировкой/ 0.13 мм^2	3M	3811 / 40
С цветовой кодировкой/ 0.1 мм^2	Belden	9R280 40
С цветовой кодировкой/ 0.1 мм^2	DuPont	76177-040

Разъемы под плоский кабель

Эти разъемы под плоский кабель предназначены для соединения плоского кабеля с клеммным блоком. Это — плоские разъемы типа ”розетки” 2 ’ 20 с центральными контактами 2.54 x 2.54 мм.

Описание/Тип	Поставщик	Номер детали
Разъем	3M	3417-7640
Strain Relief	3M	3448-3040

Клеммный блок для сопряжения

Ниже приведены клеммные блоки, которые могут использоваться для сопряжения 40-жильного плоского кабеля с 40 отдельными проводами полевых устройств. Характеристики клеммного блока: головные клеммы (центры 5 мм) с разъемом типа ”вилки” со штырьками 2 x 20 2.54 x 2.54 мм, подсоединяемые провода 0.33 — 3.1 мм^2, без предохранителя.

Описание/Тип	Поставщик	Номер детали
Монтируется на панели	W eidmuller	RI-40A /914897
Монтируется на DIN-рейке		RI-40A /914908
Монтируется на DIN-рейке	Phoenix Contacts	FLKM 40 / 2281076
Монтируется на DIN-рейке	Augat/RDI	2M40FC

Контрольная таблица при монтаже ввода/вывода

Используйте следующие указания при монтаже модулей ввода/вывода в вашей системе.

1. Существует предел по размеру провода, который допускают модули. В приводимой ниже таблице приводятся максимальные сечения проводов для каждого типа модулей. Для подсоединения каждого модуля следует использовать по возможности большее сечение.

Тип модуля	Минимальное сечение
На 8 точек	3.1 мм^2
На 16 точек	2.0 мм^2
На 32 точки - общий провод	0.5 мм^2
На 32 точки - другие	0.2 мм^2
На 64 точки	0.2 мм^2 (требуется плоский кабель)
F4-08THM-X	4.1 мм^2 (провод от термопар)

ПРИМЕЧАНИЕ. Для модулей на 8 точек можно использовать провода сечением 3.1 мм^2 типа TFFN или типа MTW. Для модулей на 16 точек можно использовать провода сечением 2.0 мм^2 типа TFFN или типа MTW. Можно применять другие типы проводов, но их применение фактически зависит от толщины изоляции провода. Если изоляция чересчур толста, и вы используете все точки ввода/вывода, то Вы не сможете надлежащим образом закрыть пластиковую крышку клеммника.

2. Всегда используйте целый провод, не используйте скрученные провода для получения нужной длины.
3. Используйте самую короткую допустимую длину провода.
4. При прокладке проводов используйте по возможности желоба.
5. Избегайте прокладки проводов рядом с высокоэнергетическими линиями.
6. По возможности избегайте прокладки входных проводов близко к выходным проводам.
7. Минимизируйте падение напряжение при прокладке проводов на большое расстояние, рассмотрите возможность применения многожильных проводов для обратной линии.
8. По возможности избегайте прокладки проводов постоянного тока в непосредственной близости к проводам переменного тока.
9. Избегайте остроугольных изгибов при прокладке проводов.
10. ВАЖНО! Предлагаем вам добавить внешние предохранители в схему соединений модулей ввода/вывода, чтобы уменьшить риск перегорания предохранителя внутри модуля. Можно добавить в каждом общем проводе быстродействующий предохранитель с более низким токовым номиналом, чем у предохранителя модуля ввода/вывода, либо добавить в каждую выходную цепь предохранитель с номиналом, который немного меньше, чем максимальный ток в каждой выходной точке.

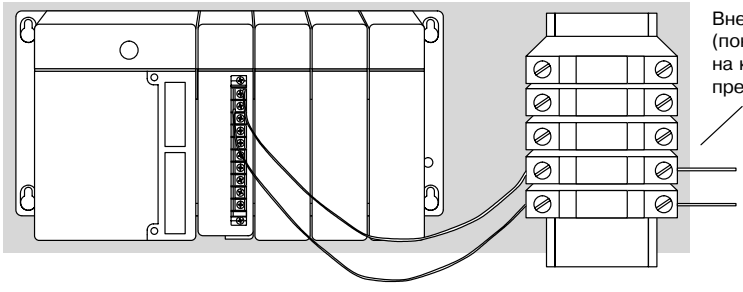


Таблица входных модулей DL405

В следующей таблице перечислены входные модули DL405. Их спецификации начинаются со следующей страницы.

Тип входных модулей DL405	Число входных точек	Вход — приемник постоянного тока	Вход — источник постоянного тока	Вход переменного тока
D4-16ND2	16		✔	
D4-16ND2F	16		✔	
D4-32ND3-1	32	✔	✔	
D4-32ND3-2	32	✔	✔	
D4-64ND2	64	✔		
D4-08NA	8			✔
D4-16NA	16			✔
D4-16NE3	16	✔	✔	✔
F4-08NE3S	8	✔	✔	✔
D4-08ND3S	8	✔	✔	

Таблица выходных модулей DL405

В следующей таблице перечислены выходные модули DL405. Их спецификации начинаются после спецификаций входных модулей.

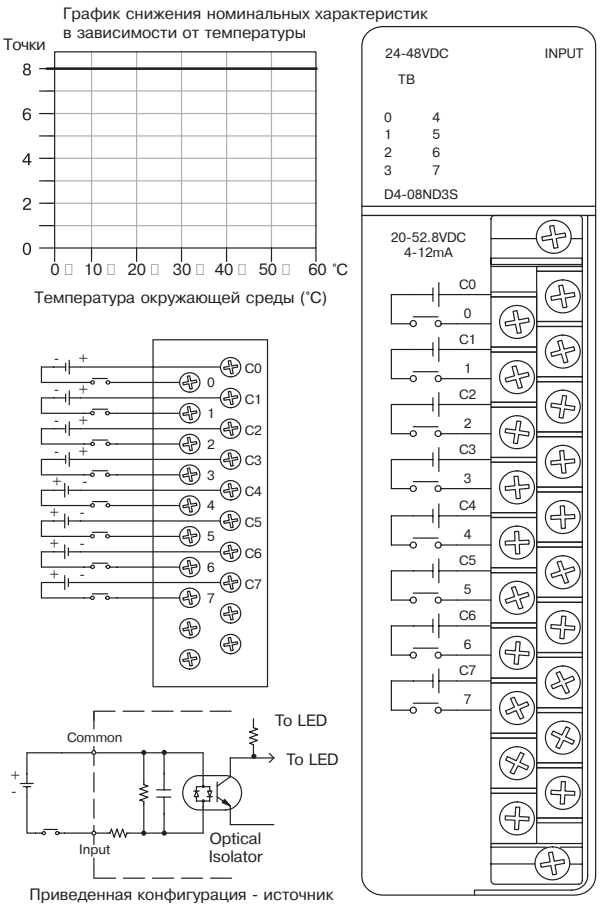
Тип выходных модулей DL405	Число выходных точек	Выход — приемник постоянного тока	Выход — источник постоянного тока	Выход переменного тока
D4-08TD1	8	✔		
F4-08TD1S	8	✔		
D4-16TD1	16	✔		
D4-16TD2	16		✔	
D4-32TD1	32	✔		
D4-32TD1-1	32	✔		
D4-32TD2	32		✔	
D4-64TD1	64	✔		
D4-08T A	8			✔
D4-16T A	16			✔
D4-08TR	8	✔	✔	✔
F4-08TRS-1	8	✔	✔	✔
F4-08TRS-2	8	✔	✔	✔
D4-16TR	16	✔	✔	✔

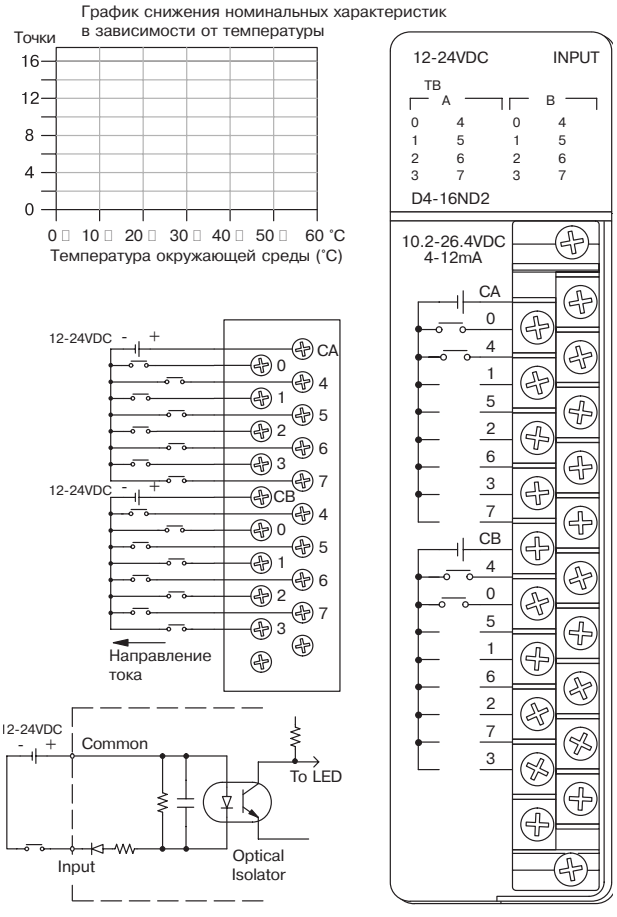
D4-08ND3S входы постоянного тока

Число входов в модуле	8 (потребитель/источник)
Число общих проводов	8 (изолированные)
Напряжение на входах	20 - 52.8 В
Допустимое кратковременное напряжение	52.8 В
Уровень в состоянии «Вкл»	> 18 В
Уровень в состоянии «Выкл»	< 7 В
Входное сопротивление	4.8 КОм
Входной ток	5 мА при 24 В 10 мА при 48 В
Ток в состоянии «Вкл»	3.5 мА
Ток в состоянии «Выкл»	1.5 мА
Потребляемый от каркаса ток 5 В	макс. 100 мА
Время перехода из «Выкл» → «Вкл»	3 - 10 мсек.
Время перехода из «Вкл» → «Выкл»	3 - 12 мсек.
Клеммный блок	съемный
Индикаторы состояния	есть, на логической части
Вес	250 гр.

D4-16ND2 входы постоянного тока

Число входов в модуле	16 (источник)
Число общих проводов	2 (изолированные)
Напряжение на входах	10.2 - 26.4 В
Допустимое кратковременное напряжение	26.4 В
Уровень в состоянии «Вкл»	> 9.5 В
Уровень в состоянии «Выкл»	< 4 В
Входное сопротивление	3.2 КОм при 12В 2.9 КОм при 24В
Входной ток	3.8 мА при 12 В 8.3 мА при 24 В
Ток в состоянии «Вкл»	3.5мА
Ток в состоянии «Выкл»	1.5 мА
Потребляемый от каркаса ток 5В	макс. 150 мА
Время перехода из «Выкл» → «Вкл» стандарт)	1 - 7 мсек. (2.3
Время перехода из «Вкл» → «Выкл»	2 - 12 мсек. (4.6 стандарт)
Клеммный блок	съемный
Индикаторы состояния	есть, на логической части
Вес	250 гр.



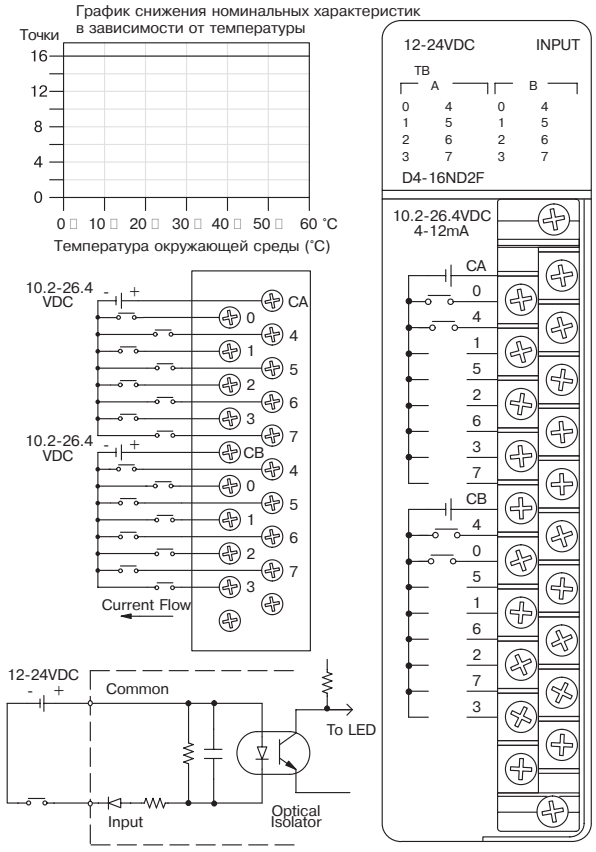


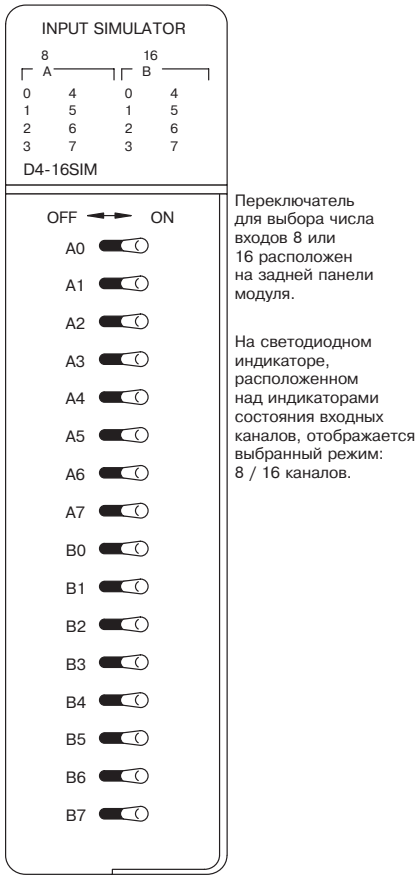
D4-16ND2F входы постоянного тока

Число входов в модуле	16 (потребитель)
Число общих проводов	2 (изолированные)
Напряжение на входах	10.2 - 26.4 В
Допустимое кратковременное напряжение	26.4 В
Уровень в состоянии «Вкл»	> 9.5 В
Уровень в состоянии «Выкл»	< 4 В
Входное сопротивление	3.2 КОм при 12В 2.9 КОм при 24В
Входной ток	3.8 мА при 12 В 8.3 мА при 24 В
Ток в состоянии «Вкл»	3.5мА
Ток в состоянии «Выкл»	1.5 мА
Потребляемый от каркаса ток 5В	макс. 150 мА
Время перехода из «Выкл» → «Вкл»	1мсек.
Время перехода из «Вкл» → «Выкл»	1мсек.
Клеммный блок	съемный
Индикаторы состояния	есть, на логической части
Вес	250 гр.

D4-16SIM имитатор входов

Число входов в модуле	8 или 16, устанавливается внутренним переключателем
Потребляемый от каркаса ток	макс. 150 мА
Клеммный блок	Отсутствует
Индикаторы состояния	есть, на логической части
Вес	250 гр.





Переключатель для выбора числа входов 8 или 16 расположен на задней панели модуля.

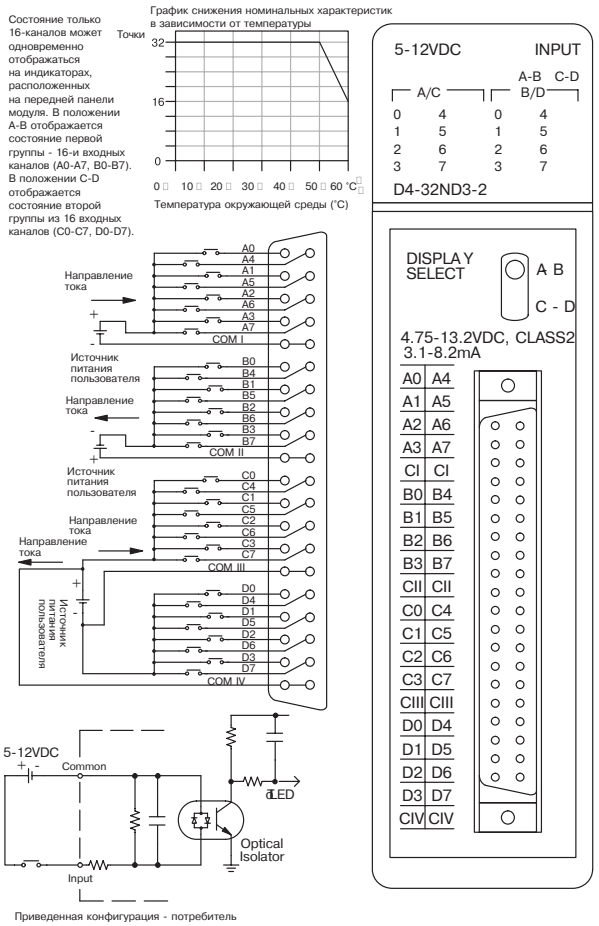
На светодиодном индикаторе, расположенном над индикаторами состояния входных каналов, отображается выбранный режим: 8 / 16 каналов.

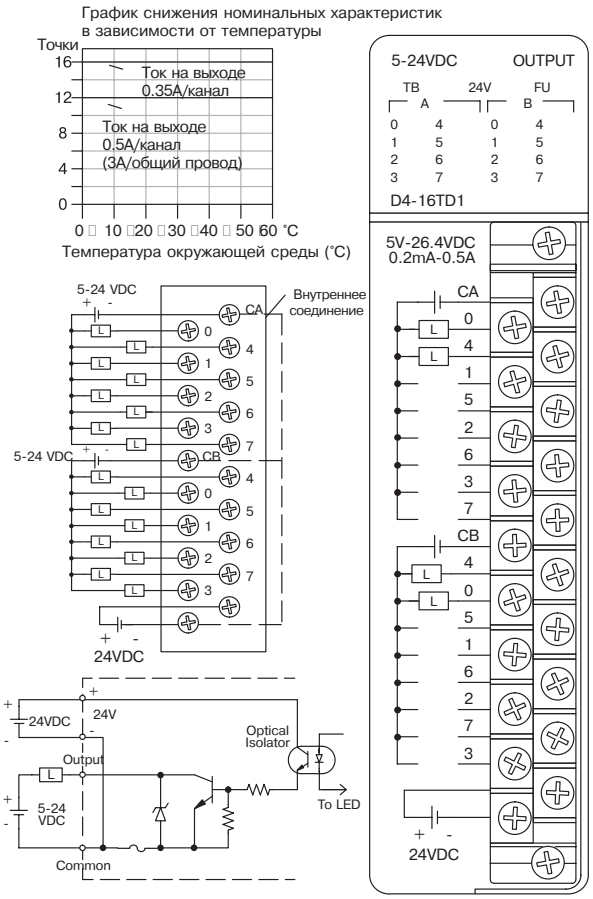
D4-32ND3-2 входы постоянного тока

Число входов в модуле	32 (потребитель/источник)
Число общих проводов	4 (изолированные)
Напряжение на входах	4.75 - 13.2 В
Допустимое кратковременное напряжение	15 В
Уровень в состоянии «Вкл»	> 4 В
Уровень в состоянии «Выкл»	< 2 В
Входное сопротивление	2 КОм при 5В 1.6 КОм при 12В
Входной ток	2.5 мА при 5 В 7.5 мА при 12 В
Ток в состоянии «Вкл»	1.8 мА
Ток в состоянии «Выкл»	0.8 мА
Потребляемый от каркаса ток 5 В	макс. 150 мА
Время перехода из «Выкл» → «Вкл»	1- 4 мсек.
Время перехода из «Вкл» → «Выкл»	1- 4 мсек.
Клеммный блок Помимо разъемов, показанных на предыдущей странице, рекомендуем ознакомиться с компонентами системы быстрого монтажа ZIPLink.	Съемный **Внимание.** Разъемы, приобретаемые отдельно D4-IO3264R - плоский обжимной, D4-IO3264S - с паянными контактами D4-IOCBL-1 - гибкий кабель 3м.
Индикаторы состояния	есть, на логической части
Вес	190 гр.

D4-1TD1 выходы постоянного тока

Число выходов	16 (потребитель)
Число общих проводов	2 соединены внутри
Рабочее напряжение	=4.5 - 26.4 В
Тип вывода	n-p-n транзистор (открытый коллектор)
Наибольшее кратковременное напряжение	=40 В
Падение напряжения во включенном состоянии макс.	0.5В при 0.5А 0.2В при 0.1А
Максимальный ток (нагрузка)	0.5А /канал 3А / общий провод
Максимальная утечка тока	0.1мА при 40В
Кратковременная перегрузка	2А при 10мс 1А при 100мс
Минимальная нагрузка	0.2мА
Потребляемый от каркаса ток (при 5В)	макс. 200 мА
Требуемое внешнее питание	24В ±10%, 125мА
Время перехода из «Выкл» → «Вкл»	0.5мсек.
Время перехода из «Вкл» → «Выкл»	0.5мсек.
Клеммный блок	съемный
Индикаторы состояния	есть, на логической части
Вес	270 гр.
Предохранители	1 (5А) на общем проводе, несъемный



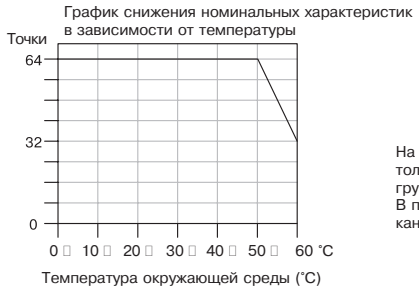


D4-64ND2 входы постоянного тока

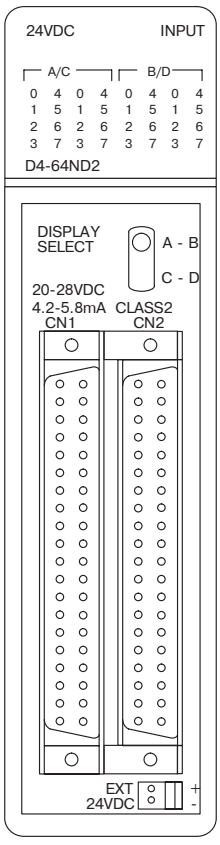
Расположение модуля	Только в каркасе процессора *
Число входов в модуле	64 (потребитель/источник)
Число общих проводов	8 (изолированные)
Напряжение на входах	20 - 28 В
Допустимое кратковременное напряжение	30 В
Уровень в состоянии «Вкл»	> 20 В
Уровень в состоянии «Выкл»	< 13 В
Входное сопротивление	4.8 КОм
Входной ток	5.0 мА при 24 В
Ток в состоянии «Вкл»	3.6 мА
Ток в состоянии «Выкл»	2.6 мА
Потребляемый от каркаса ток 5 В	макс. 300 мА

Внешнее питание (дополнительно)	24В ±10%, макс. 320мА
Время перехода из «Выкл» → «Вкл»	2.5 мсек. (стандарт)
Время перехода из «Вкл» → «Выкл»	5 мсек. (стандарт)
Клеммный блок	Съемный **Внимание.** Разъемы, приобретаемые отдельно D4-IO3264R - плоский обжимной, D4-IO3264S - с паяными контактами D4-IOCBL-1 - гибкий кабель 3м.
Индикаторы состояния	есть, на логической части
Вес	220 гр.

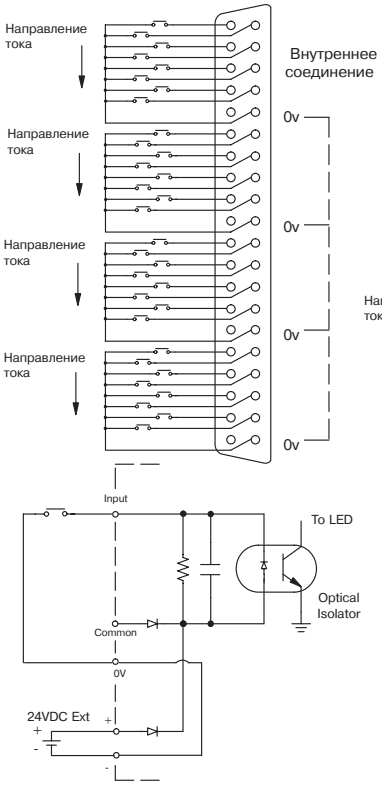
* Это ограничение действительно только для процессоров DL430/440. Модули могут быть установлены в каркасах расширения DL450, новой модификации (-1).



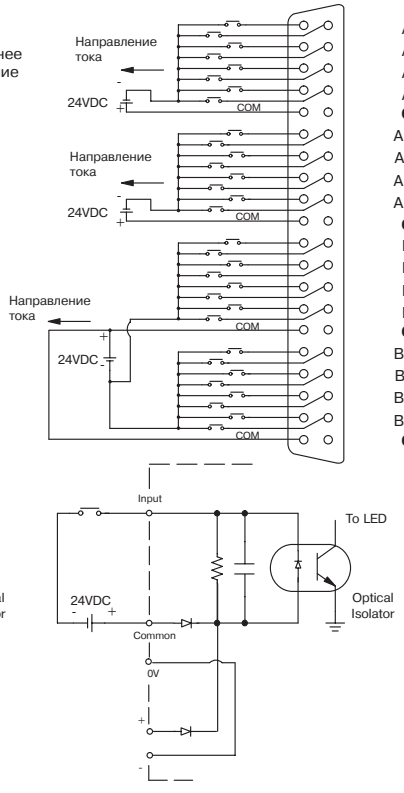
На 32 индикаторах на модуле может одновременно отображаться состояние только 32 каналов. В положении A-B отображается состояние первой группы - 32 входных каналов (A0-A17, B0-B17) (разъем 1).
В положении C-D отображается состояние второй группы из 32 входных каналов (C0-C17, D0-D17) (разъем 2).



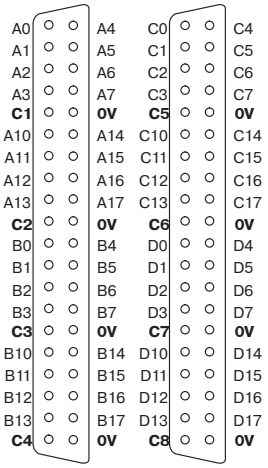
Присоединение 32-х каналов
с использованием Разъема EXT 24В



Присоединение 32-х каналов
при 24В на разъеме



Контакты на разъеме

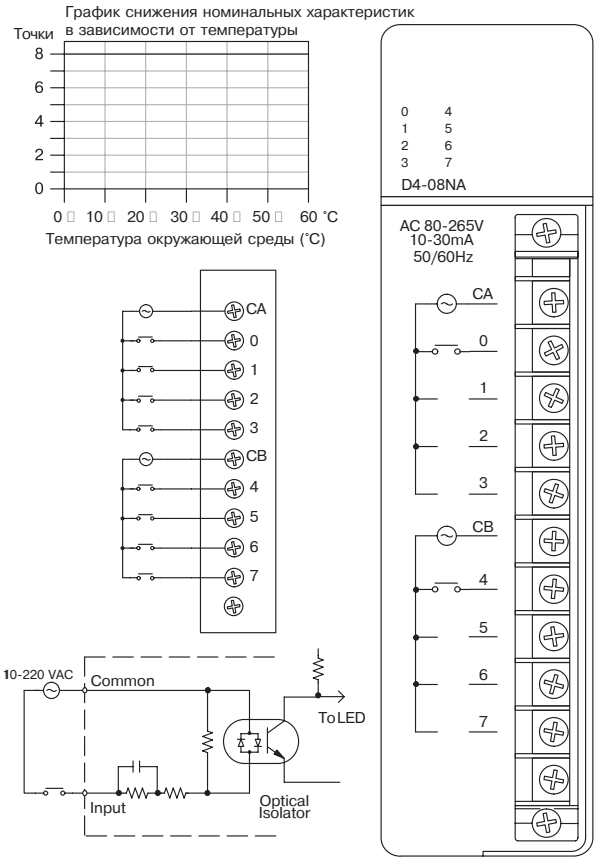


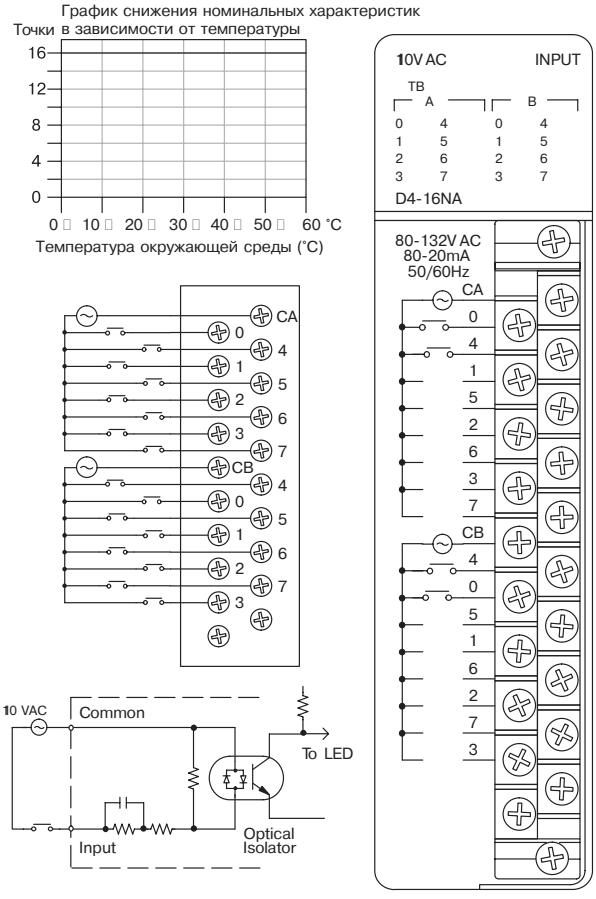
D4-08NA входы переменного тока 220В

Число входов в модуле	8
Число общих проводов	2 (изолированные)
Напряжение на входах	~80-265 В
Допустимое кратковременное напряжение	~265 В
Сигнал переменного тока	47-63Гц
Уровень в состоянии «Вкл»	> 70 В
Уровень в состоянии «Выкл»	< 30 В
Входное сопротивление	12 КОм
Входной ток	8.5 мА при ~100 В 20 мА при ~230 В
Ток в состоянии «Вкл»	5 мА
Ток в состоянии «Выкл»	2 мА
Потребляемый от каркаса ток	макс. 100 мА
Время перехода из «Выкл» → «Вкл»	5 - 30 мсек.
Время перехода из «Вкл» → «Выкл»	10 - 50 мсек.
Клеммный блок	Съемный
Индикаторы состояния	есть, на логической части
Вес	240 гр.

D4-16NA входы переменного тока 110В

Число входов в модуле	16
Число общих проводов	2 (изолированные)
Напряжение на входах	~80-132 В
Допустимое кратковременное напряжение	132 В
Сигнал переменного тока	47-63Гц
Уровень в состоянии «Вкл»	> 70 В
Уровень в состоянии «Выкл»	< 20 В
Входное сопротивление	8 КОм
Входной ток	14.5 мА при ~120 В
Ток в состоянии «Вкл»	7 мА
Ток в состоянии «Выкл»	2 мА
Потребляемый от каркаса ток	макс. 150 мА
Время перехода из «Выкл» → «Вкл»	5 - 30 мсек.
Время перехода из «Вкл» ® «Выкл»	10 - 50 мсек.
Клеммный блок	Съемный
Индикаторы состояния	есть, на логической части
Вес	270 гр.



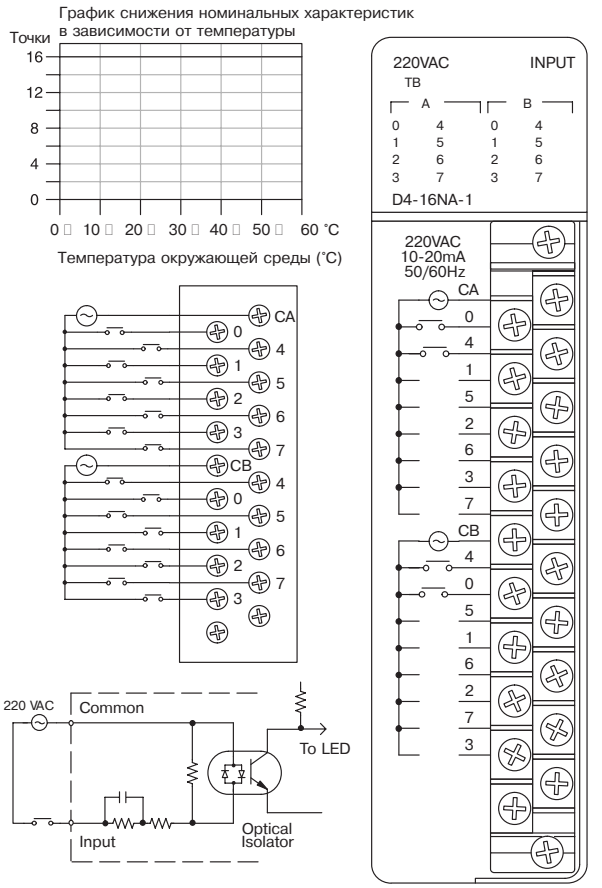


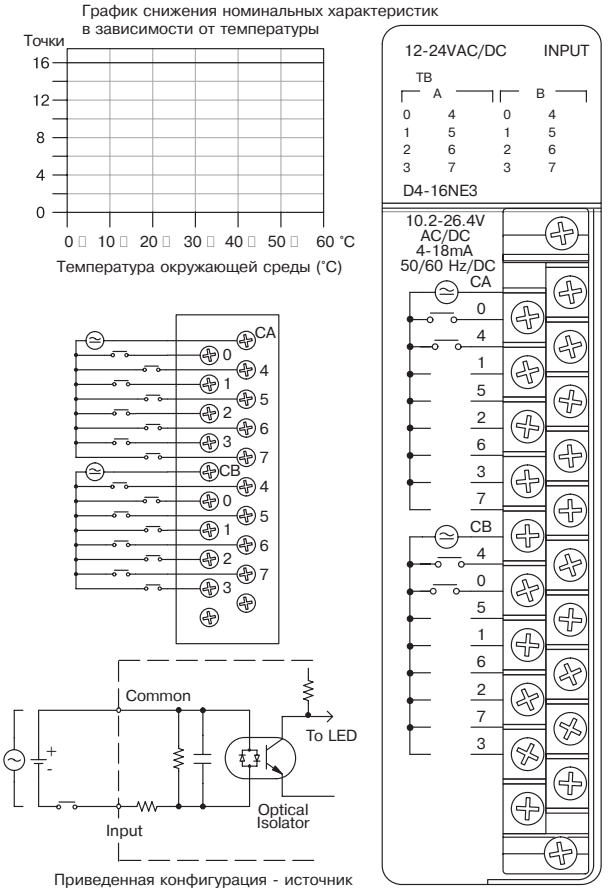
D4-16NA-1 входы переменного тока 220В

Число входов в модуле	16
Число общих проводов	2 (изолированные)
Напряжение на входах	~187-238 В
Допустимое кратковременное напряжение	~265 В
Сигнал переменного тока	47-63Гц
Уровень в состоянии «Вкл»	> 150 В
Уровень в состоянии «Выкл»	< 40 В
Входное сопротивление	22 КОм
Входной ток	10.0 мА при ~220 В
Ток в состоянии «Вкл»	7 мА
Ток в состоянии «Выкл»	2 мА
Потребляемый от каркаса ток	макс. 150 мА
Время перехода из «Выкл» → «Вкл»	5 - 30 мсек.
Время перехода из «Вкл» → «Выкл»	10 - 50 мсек.
Клеммный блок	Съемный
Индикаторы состояния	есть, на логической части
Вес	260 гр.

D4-16NE3 входы переменного/постоянного тока

Число входов в модуле	16 (потребитель/источник)
Число общих проводов	2 (изолированные)
Напряжение на входах	~10.2-26.4В / 10.2-26.4В
Допустимое кратковременное напряжение	~37.5В / 37.5В
Сигнал переменного тока	47-63Гц
Уровень в состоянии «Вкл»	> 9.5 В
Уровень в состоянии «Выкл»	< 3 В
Входное сопротивление	3.2 КОм при 12В 2.9 КОм при 24В
Входной ток	3.8 мА при 12 В 8.3 мА при 24 В
Ток в состоянии «Вкл»	4 мА
Ток в состоянии «Выкл»	1.5 мА
Потребляемый от каркаса ток 5 В	макс. 150 мА
Время перехода из «Выкл» → «Вкл»	5- 40 мсек.
Время перехода из «Вкл» → «Выкл»	10 - 50 мсек.
Клеммный блок	Съемный
Индикаторы состояния	есть, на логической части
Вес	250 гр.





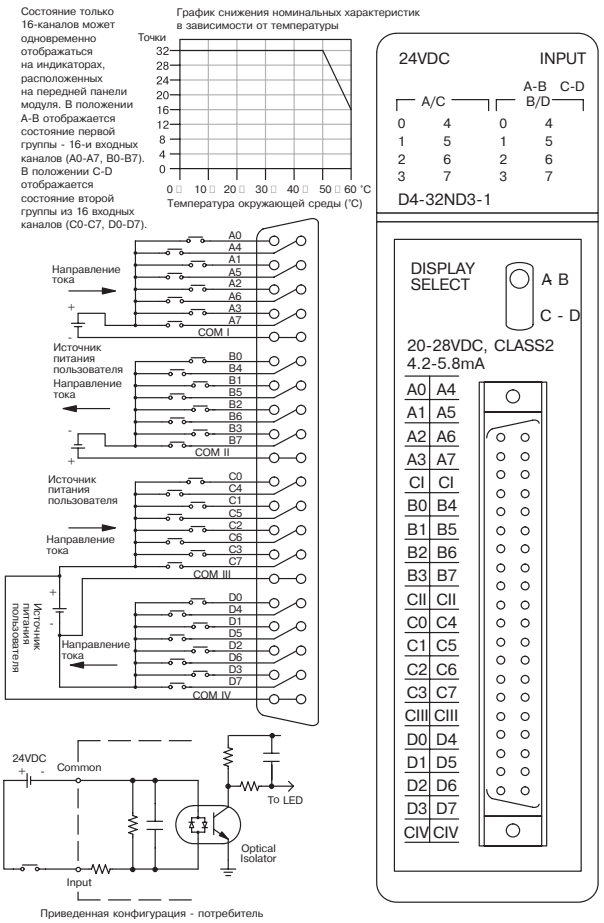
Приведенная конфигурация - источник

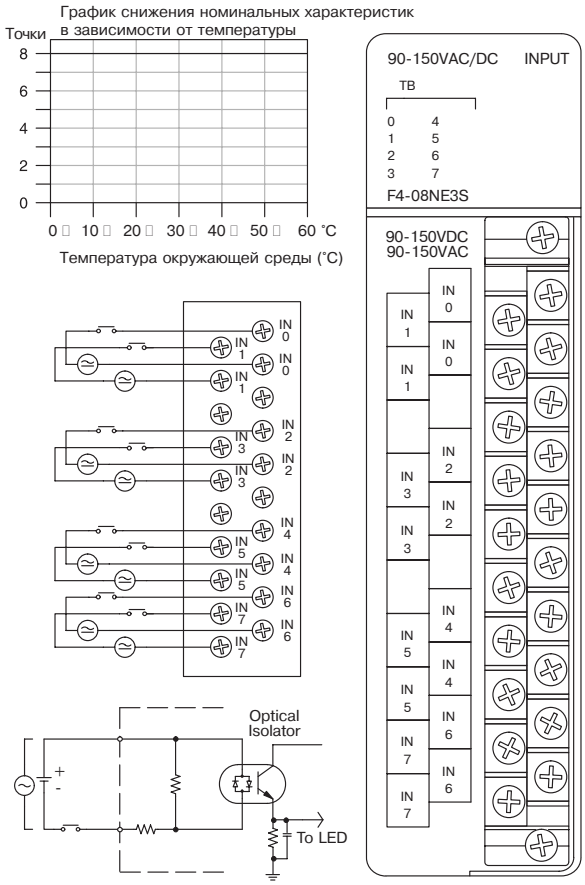
D4-32ND3-1 входы постоянного тока

Число входов в модуле	32 (потребитель/источник)
Число общих проводов	4 (изолированные)
Напряжение на входах	20 - 28 В
Допустимое кратковременное напряжение	30 В
Уровень в состоянии «Вкл»	> 19 В
Уровень в состоянии «Выкл»	< 10 В
Входное сопротивление	4.8 КОм
Входной ток	5 мА при 24 В
Ток в состоянии «Вкл»	3.5 мА
Ток в состоянии «Выкл»	1.6 мА
Потребляемый от каркаса ток 5 В	макс. 150 мА
Время перехода из «Выкл» → «Вкл»	2 - 10 мсек.
Время перехода из «Вкл» → «Выкл»	2 - 10 мсек.
Клеммный блок Помимо разъемов, показанных на предыдущей странице, рекомендуем ознакомиться с компонентами системы быстрого монтажа ZIPLink.	Съемный **Внимание.** Разъемы, приобретаемые отдельно D4-IO3264R - плоский обжимной, D4-IO3264S - с паянными контактами D4-IOCBL-1 - гибкий кабель 3м.
Индикаторы состояния	есть, на логической части
Вес	190 гр.

D4-08NE3S входы переменного/постоянного тока

Число входов в модуле	8 (потребитель/источник)
Число общих проводов	8 (изолированные)
Напряжение на входах	~90-150В / 90-150В
Допустимое кратковременное напряжение	350, < 1 мсек.
Сигнал переменного тока	47-63Гц
Уровень в состоянии «Вкл»	> 90В / ~75В
Уровень в состоянии «Выкл»	< 60В / ~45В
Входное сопротивление	22 КОм
Входной ток	5.5 мА при 120В
Ток в состоянии «Вкл»	4 мА
Ток в состоянии «Выкл»	2 мА
Потребляемый от каркаса ток 5 В	макс. 90 мА
Время перехода из «Выкл» → «Вкл»	8 мсек.
Время перехода из «Вкл» → «Выкл»	15 мсек.
Клеммный блок	Съемный
Индикаторы состояния	есть, на логической части
Вес	256 гр.



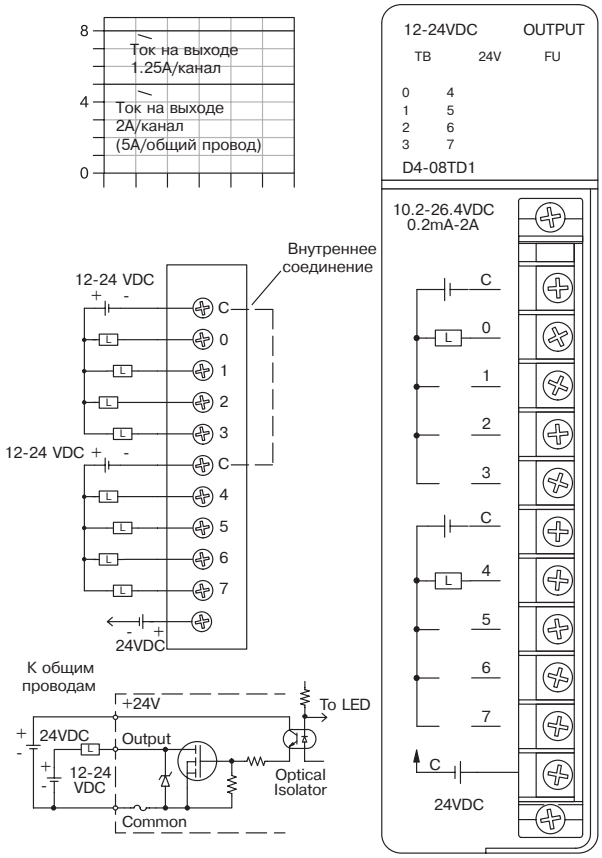


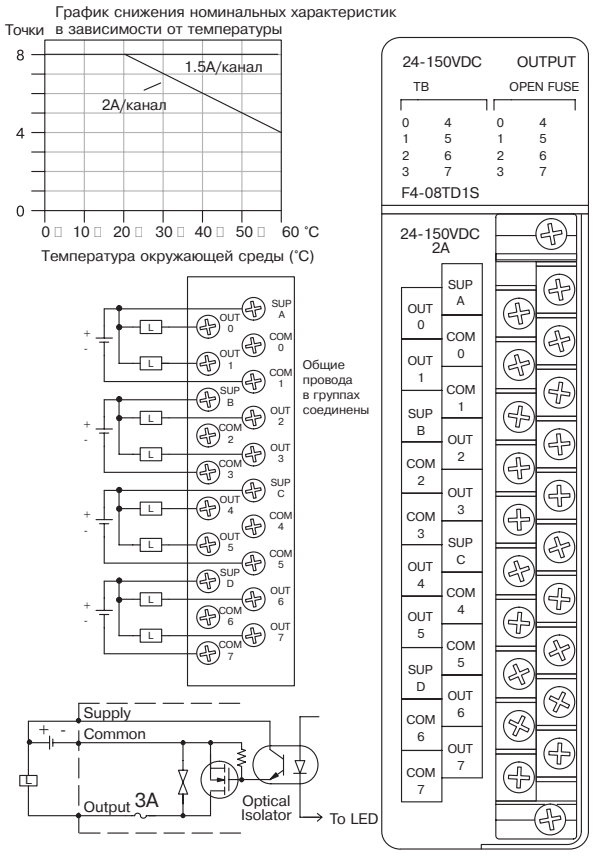
D4-08TD1 выходы постоянного тока

Число выходов	8 (потребитель)
Число общих проводов	2 соединены внутри
Рабочее напряжение	=10.2 - 26.4 В
Тип вывода	n-канальный МОП-полевой транзистор (открытый сток)
Наибольшее кратковременное напряжение	40 В
Падение напряжения во включенном состоянии макс	=0.5В при 2А =0.2В при 1 А
Максимальный ток (нагрузка)	2А /канал 5А / общий провод
Максимальная утечка тока	0.1мА при 40В
Кратковременная перегрузка	12А при 10мс, 6А при 100мс
Минимальная нагрузка	0.2мА
Потребляемый от каркаса ток (при 5В)	макс. 150 мА
Требуемое внешнее питание	24В ±10%, 35мА
Время перехода из «Выкл» → «Вкл»	1мсек.
Время перехода из «Вкл» → «Выкл»	1мсек.
Клеммный блок	съемный
Индикаторы состояния	есть, на логической части
Вес	240 гр.
Предохранители	1 (7А) на общем проводе, несъемный

F4-08TD1S выходы постоянного тока

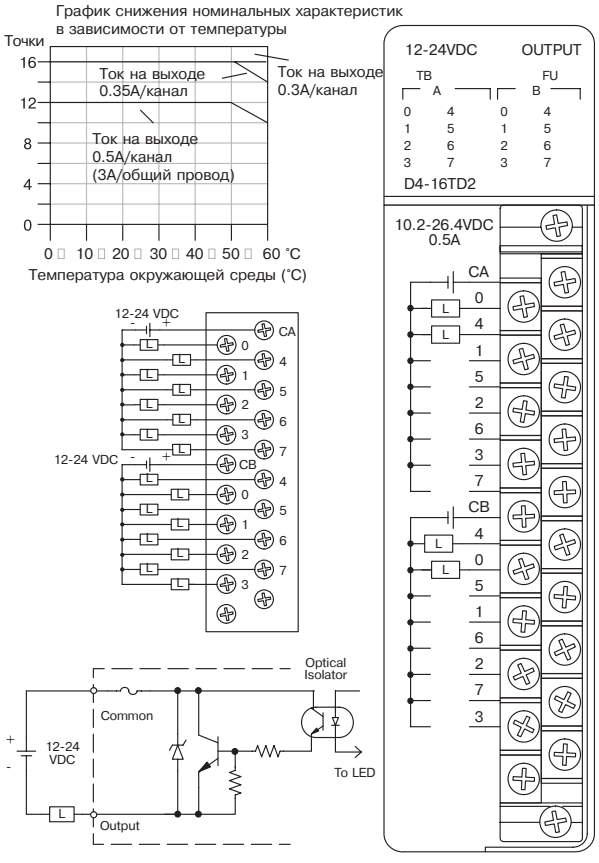
Число выходов	8 (потребитель)
Число общих проводов	4 (изолированные, 8 контактов)
Рабочее напряжение	=24 - 150 В
Тип вывода	МОП- полевой транзистор
Наибольшее кратковременное напряжение	=200В < 1сек.
Падение напряжения во включенном состоянии макс.	=0.5В при 2А
Максимальный ток (нагрузка)	2А /канал 4А / общий провод
Максимальная утечка тока	1мкА
Кратковременная перегрузка	30А при 1мс 19А при 10мс
Минимальная нагрузка	Не определена
Потребляемый от каркаса ток (при 5В)	макс. 295 мА
Требуемое внешнее питание	Нет
Время перехода из «Выкл» → «Вкл»	25 мксек.
Время перехода из «Вкл» → «Выкл»	25 мксек.
Клеммный блок	съемный
Индикаторы состояния	есть, на логической части
Вес	282 гр.
Предохранители	1 (3А) на каждом выходе (см. схему), несъемный





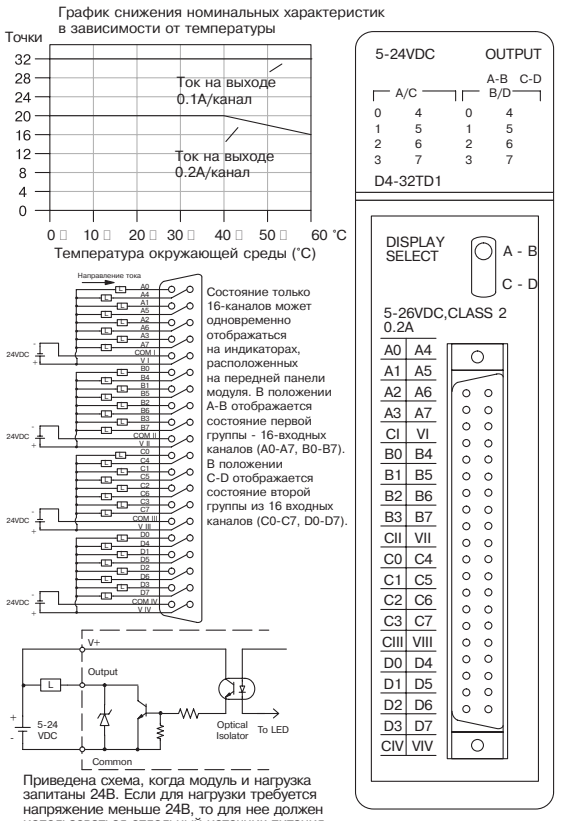
D4-16TD2 выходы постоянного тока

Число выходов	16 (потребитель)
Число общих проводов	2 (изолированные)
Рабочее напряжение	=10.2 - 26.4 В
Тип вывода	n-p-n транзистор (эмиттерный повторитель)
Наибольшее кратковременное напряжение	=40 В
Падение напряжения во включенном состоянии макс.	=1.5В при 0.5А
Максимальный ток (нагрузка)	0.5А /канал 3А / общий провод при 50°С 2.5А / общий провод при 60°С
Максимальная утечка тока	0.1мА при 40В
Кратковременная перегрузка	2А при 10мс 1А при 100мс
Минимальная нагрузка	0.2мА
Потребляемый от каркаса ток (при 5В)	макс. 400 мА
Требуемое внешнее питание	Нет
Время перехода из «Выкл» → «Вкл»	1мсек.
Время перехода из «Вкл» → «Выкл»	1мсек.
Клеммный блок	съемный
Индикаторы состояния	есть, на логической части
Вес	280 гр.
Предохранители	1 (5А) на общем проводе, несъемный



D4-32TD1 выходы постоянного тока

Число выходов	32 (потребитель)
Число общих проводов	4 (изолированные)
Рабочее напряжение	4.75 - 26.4 В
Тип вывода	n-p-n транзистор (открытый коллектор)
Наибольшее кратковременное напряжение	36 В
Падение напряжения во включенном состоянии	макс. 0.6В при 0.2А
Максимальный ток (нагрузка)	0.2А /канал 1.6А / общий провод
Максимальная утечка тока	0.1мА при 36В
Кратковременная перегрузка	1А при 10мсек 0.5А при 100мсек
Минимальная нагрузка	0.1мА
Потребляемый от каркаса ток (при 5В)	макс. 250 мА
Требуемое внешнее питание	24В ±10%, макс.140мА
Время перехода из «Выкл» → «Вкл»	0.1мсек.
Время перехода из «Вкл» → «Выкл»	0.1мсек.
Клеммный блок Помимо разъемов, показанных на предыдущей странице, рекомендуем ознакомиться с компонентами системы быстрого монтажа ZIPLink.	Съемный **Внимание.** Разъемы, приобретаемые отдельно D4-IO3264R — плоский обжимной, D4-IO3264S — с паяными контактами D4-IOCBL-1 — гибкий кабель 3м.
Индикаторы состояния	есть, на логической части
Вес	190 гр.
Предохранители	Нет



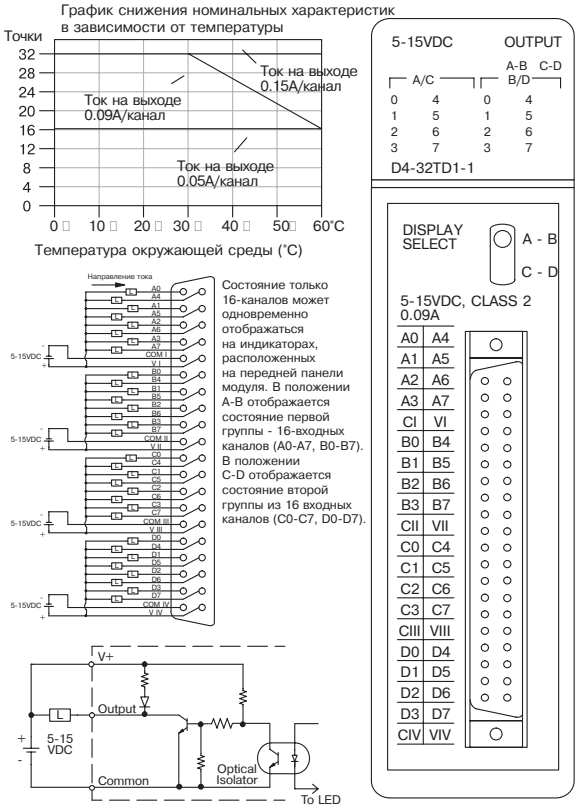
Приведена схема, когда модуль и нагрузка запитаны 24В. Если для нагрузки требуется напряжение меньше 24В, то для нее должен использоваться отдельный источник питания.

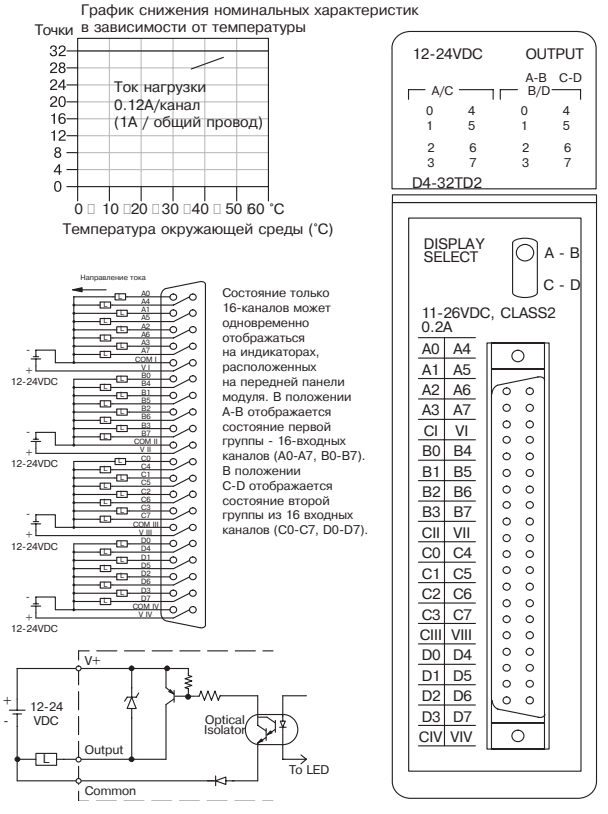
D4-32TD1-1 выходы постоянного тока

Число выходов	32 (потребитель)
Число общих проводов	4 (изолированные)
Рабочее напряжение	=5 - 15 В
Тип вывода	n-p-n транзистор (открытый коллектор)
Наибольшее кратковременное напряжение	=16.5 В
Падение напряжения во включенном состоянии макс.	0.4В при 0.1А
Максимальный ток (нагрузка)	0.09А /канал 0.72А / общий провод 2.88А / модуль
Максимальная утечка тока	0.01мА при 16.5В
Кратковременная перегрузка	0.5А при 10мсек 0.2А при 100мсек
Минимальная нагрузка	0.15мА
Потребляемый от каркаса ток (при 5В)	макс. 250 мА
Требуемое внешнее питание	=5-15В ±10%, макс. 150мА
Время перехода из «Выкл» → «Вкл»	0.1мсек.
Время перехода из «Вкл» → «Выкл»	0.1мсек.
Клеммный блок Помимо разъемов, показанных на предыдущей странице, рекомендуем ознакомиться с компонентами системы быстрого монтажа ZIPLink.	Съемный **Внимание.** Разъемы, приобретаемые отдельно D4-IO3264R — плоский обжимной, D4-IO3264S — с паяными контактами D4-IOCBL-1 — гибкий кабель 3м.
Индикаторы состояния	есть, на логической части
Вес	190 гр.
Предохранители	Нет

D4-32TD1-2 выходы постоянного тока

Число выходов	32 (источник)
Число общих проводов	4 (изолированные)
Рабочее напряжение	=10.8 - 26.4 В
Тип вывода	p-n-p транзистор (открытый коллектор)
Наибольшее кратковременное напряжение	=30 В
Падение напряжения во включенном состоянии макс.	0.6В при 0.2А
Максимальный ток (нагрузка)	0.2А/канал 1.0А/общий провод 4.0А/модуль
Максимальная утечка тока	0.01мА при 26.4В
Кратковременная перегрузка	500мА при 10мсек
Минимальная нагрузка	0.2мА
Потребляемый от каркаса ток (при 5В)	макс. 350 мА
Требуемое внешнее питание	=10.8-26.4В, 1А/общий провод, включая нагрузку
Время перехода из «Выкл» → «Вкл»	< 0.2 мсек.
Время перехода из «Вкл» → «Выкл»	< 0.2 мсек.
Клеммный блок Помимо разъемов, показанных на предыдущей странице, рекомендуем ознакомиться с компонентами системы быстрого монтажа ZIPLink.	Съемный **Внимание.** Разъемы, приобретаемые отдельно D4-IO3264R — плоский обжимной, D4-IO3264S — с паяными контактами D4-IOCBL-1 — гибкий кабель 3м.
Индикаторы состояния	есть, на логической части
Вес	190 гр.
Предохранители	Нет



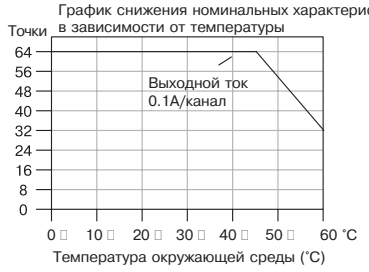


D4-64TD1 выходы постоянного тока

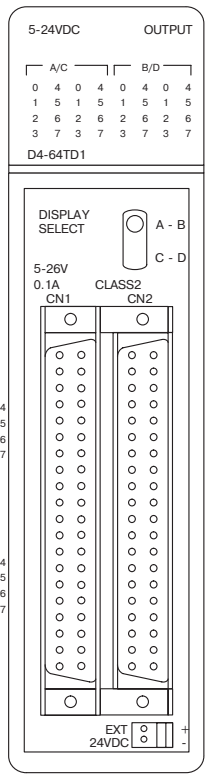
Расположение модуля:	Только в каркасе процессора *
Число выходов	64 (потребитель)
Число общих проводов	8 (изолированные)
Рабочее напряжение	=4.75 - 26.5 В
Тип вывода	n-p-n транзистор (открытый коллектор)
Наибольшее кратковременное напряжение	=36 В
Падение напряжения во включенном состоянии макс.	=0.6В при 0.1А
Максимальный ток (нагрузка)	0.1А /канал 1А / общий провод 8А / модуль
Максимальная утечка тока	0.01мА при 36В
Кратковременная перегрузка	1А при 1мсек 700мА при 100мсек
Минимальная нагрузка	0.1мА

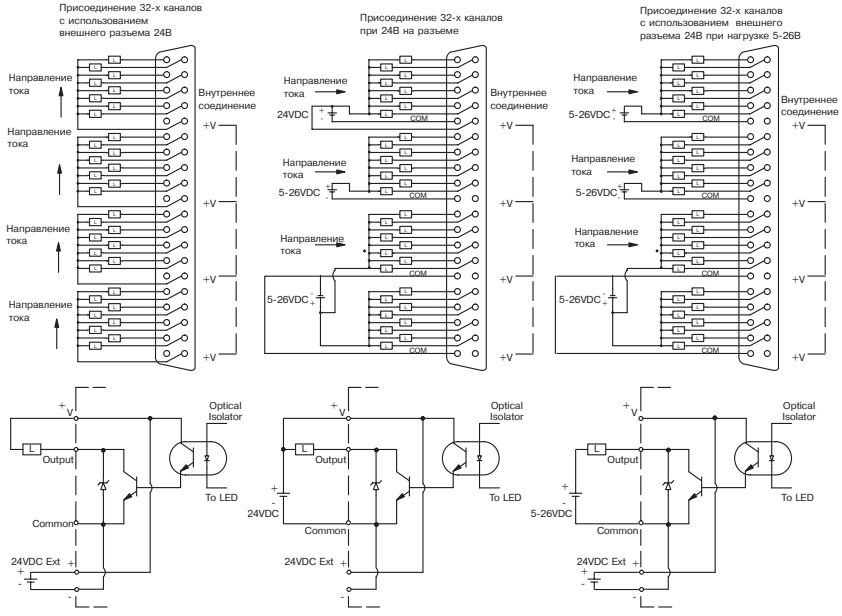
* Это ограничение действительно только для процессоров DL430/440. Модули могут быть установлены в каркасах расширения DL450, новой модификации (-1).

Потребляемый от каркаса ток (при 5В)	макс. 800 мА
Требуемое внешнее питание	=24В ±10%, (850мА + 50мА на общий провод) макс. общая мощность 7.0А
Время перехода из «Выкл» → «Вкл»	0.1мсек.
Время перехода из «Вкл» → «Выкл»	0.2мсек.
Клеммный блок	Съемный **Внимание.** Разъемы, приобретаемые отдельно D4-IO3264R — плоский обжимной, D4-IO3264S — с паяными контактами D4-IOCBL-1 — гибкий кабель 3м.
Индикаторы состояния	есть, на логической части
Вес	210 гр.
Предохранители	Нет



На 32-х индикаторах на модуле может одновременно отображаться состояние только 32-х каналов. В положении A-B отображается состояние первой группы - 32-х входных каналов (A0-A17, B0-B17) (разъем 1). В положении C-D отображается состояние второй группы из 32-х входных каналов (C0-C17, D0-D17) (разъем 2).



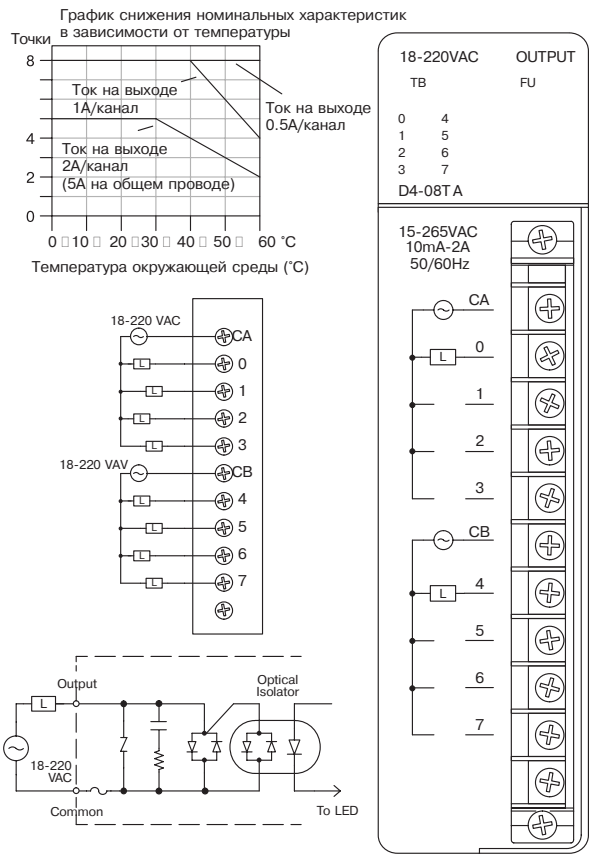


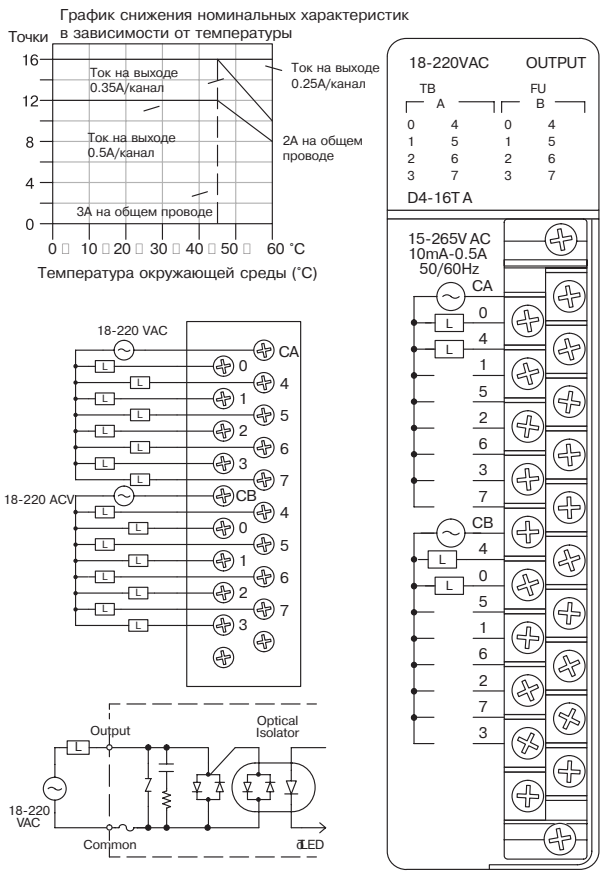
D4-08TA выходы переменного тока

Число выходов	8
Число общих проводов	2 (изолированные)
Рабочее напряжение	~15 - 265 В
Тип вывода	полупроводниковое реле (триак)
Наибольшее кратковременное напряжение	~265 В
Сигнал переменного тока	47 - 63 Гц
Падение напряжения во включенном состоянии макс.	~1.5 В при 2А
Максимальный ток	2А /канал 5А / общий провод при 30°С 2А / общий провод при 60°С
Максимальная утечка тока	5мА при ~265В
Кратковременная перегрузка	30А при 10мсек. 10А при 100мсек.
Минимальная нагрузка	10мА
Потребляемый от каркаса ток (5В)	макс. 250 мА
Время перехода из «Выкл» → «Вкл»	1мсек.
Время перехода из «Вкл» → «Выкл»	1 мсек. + 1/2 цикла
Клеммный блок	съемный
Индикаторы состояния	есть, на логической части
Вес	330 гр.
Предохранители	1 (8А) на общий провод, несъемный

D4-16TA выходы переменного тока

Число выходов	16
Число общих проводов	2 (изолированные)
Рабочее напряжение	~15 - 265 В
Тип вывода	полупроводниковое реле (триак)
Наибольшее кратковременное напряжение	~265 В
Сигнал переменного тока	47 - 63 Гц
Падение напряжения во включенном состоянии макс.	~1.5 В при 2А
Максимальный ток	0.5А/канал 3А/общий провод при 45°С 2А/общий провод при 60°С
Максимальная утечка тока	4мА при ~264В
Кратковременная перегрузка	15А при 10мсек. 10А при 100 мсек.
Минимальная нагрузка	10мА
Потребляемый от каркаса ток (5В)	макс. 450 мА
Время перехода из «Выкл» → «Вкл»	1мсек.
Время перехода из «Вкл» → «Выкл»	1 мсек. + 1/2 цикла
Клеммный блок	съемный
Индикаторы состояния	есть, на логической части
Вес	350 гр.
Предохранители	1 (5А) на общий провод, несъемный



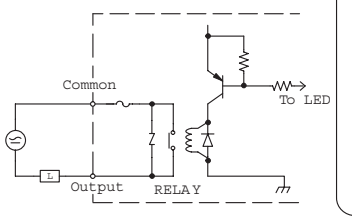


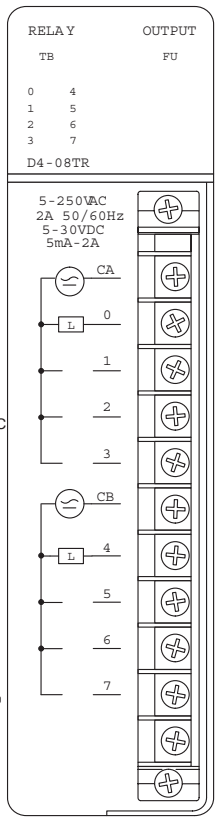
D4-08TRS релейные выходы

Число выходов	8 релейные
Число общих проводов	2 (изолированные)
Рабочее напряжение	=5-30В / ~5-440В
Тип вывода	Реле, тип А однополюсное на одно направление, нормально разомкнутое (SPST-NO)
Наибольшее кратковременное напряжение	=30В / ~265В
Сигнал переменного тока	47 - 63 Гц
Падение напряжения во включенном состоянии	Нет
Максимальный ток (активная нагрузка)	2А/канал 5А/модуль (активная нагрузка)
Максимальная утечка тока	0.1мА при ~265В
Кратковременная перегрузка	2А
Минимальная нагрузка	5мА
Потребляемый от каркаса ток	макс. 550мА
Требуемое внешнее питание	нет
Время перехода из «Выкл» → «Вкл»	12 мсек.
Время перехода из «Вкл» → «Выкл»	12 мсек.
Клеммный блок	съемный
Индикаторы состояния	есть, на логической части
Вес	260 гр.
Предохранители	1 (8А), несъемный, на общем проводе

Стандартный ресурс реле (1000 операций)

Максимальная кратковременная активная или индуктивная нагрузка	Рабочее напряжение		
	30В	125В	250В
2А активная	100К	300К	200К
2А индуктивная	100К	80К	60К
0.5А активная	800К	1М	800К
0.5А индуктивная	300К	300К	200К



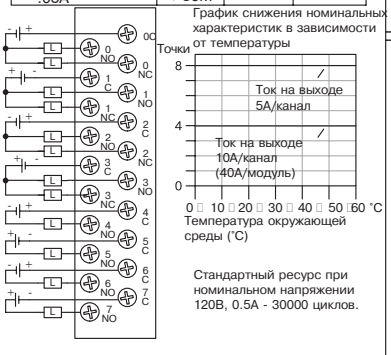


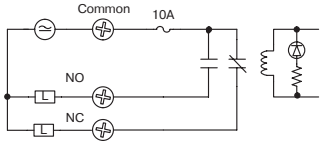
F4-08TRS-1 релейные выходы

Число выходов	8 реле
Число общих проводов	8 (изолированные)
Рабочее напряжение *(требуются внешние предохранители)	=12-30В / ~12-125В *~125-250В
Тип вывода	4 реле, тип С однополюсное на два направления (SPDT) 4 реле, тип А однополюсное на одно направление, нормально разомкнутое (SPST-NO)
Наибольшее кратковременное напряжение	=30В / ~250В при 10А
Сигнал переменного тока	47 - 63 Гц
Падение напряжения во включенном состоянии	Нет
Максимальный ток (активная нагрузка)	10А/канал 40А/модуль
Максимальная утечка тока	нет
Кратковременная перегрузка	10А
Минимальная нагрузка	100мА при 12В
Потребляемый от каркаса ток	макс. 575мА
Требуемое внешнее питание	нет
Время перехода из «Выкл» → «Вкл»	7 мсек.
Время перехода из «Вкл» → «Выкл»	9 мсек.
Клеммный блок	съемный
Индикаторы состояния	есть, на логической части
Вес	374 гр.
Предохранители	1 (10А) на общем проводе, несъемный,

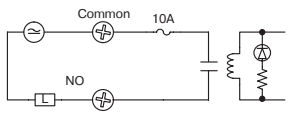
Стандартный ресурс реле (1000 операций)

Максимальная кратковременная активная или индуктивная нагрузка	Рабочее напряжение		
	28В	120В	240В
1/4HP		25К	
10.0А	50К	50К	
5.0А	200К	100К	
3.0А	325К	125К	50К
.05А	>50М		

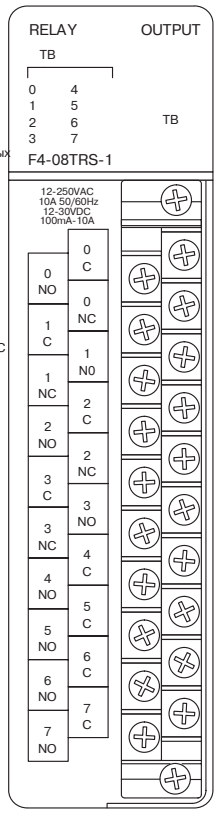




Пример схемы с релейным выходом (выходы 1 - 4)



Пример схемы с релейным выходом (выходы 5 - 8)

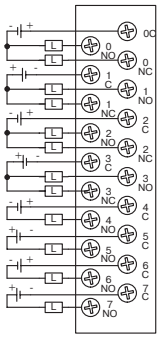


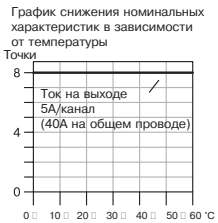
F4-08TRS-2 релейные выходы

Число выходов	8 реле
Число общих проводов	8 (изолированные)
Рабочее напряжение	=12-30В / ~12-250
Тип вывода	4 Реле, тип C однополюсное на два направления (SPDT) 4 Реле, тип A однополюсное на одно направление, нормально разомкнутое (SPST-NO)
Наибольшее кратковременное напряжение	=30В / ~250В при 5А
Сигнал переменного тока	47 - 63 Гц
Падение напряжения во включенном состоянии	Нет
Максимальный ток (активная нагрузка)	5А/канал 40А/модуль
Максимальная утечка тока	нет
Кратковременная перегрузка	10А
Минимальная нагрузка	100мА при 12В
Потребляемый от каркаса ток 60мА/канал	макс. 575мА,
Требуемое внешнее питание	нет
Время перехода из «Выкл» → «Вкл»	7 мсек.
Время перехода из «Вкл» → «Выкл»	9 мсек.
Клеммный блок	съемный
Индикаторы состояния	есть, на логической части
Вес	390 гр.
Предохранители 19379-K-10A Wickmann	1 (10А, 250В) на общем проводе Съемный

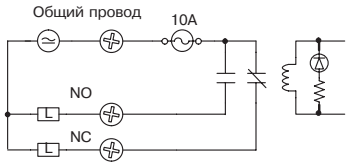
Стандартный ресурс реле (1000 операций)

Максимальная кратковременная активная или индуктивная нагрузка	Рабочее напряжение		
	28В	120В	240В
5.0 A 3.0 A 0.05 A	200K 325K >50M	100K 125K	50K

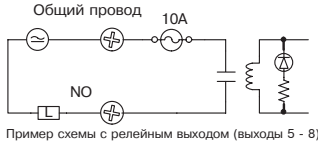




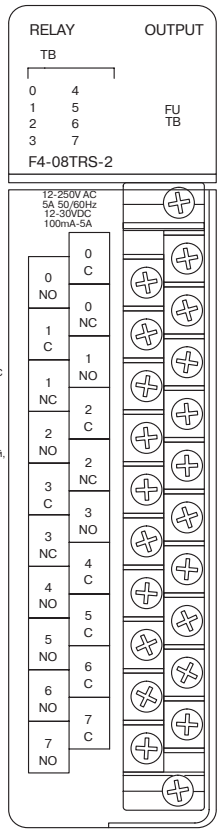
Стандартный ресурс при номинальном напряжении 120В, 0.5А - 30000 циклов. Данный модуль можно использовать в пусковых устройствах электродвигателей, вплоть до типа NEMA, размер 3.



Пример схемы с релейным выходом (выходы 1 - 4)



Пример схемы с релейным выходом (выходы 5 - 8)

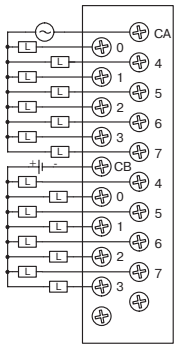


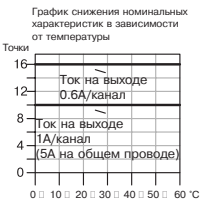
D4-16TR релейные выходы

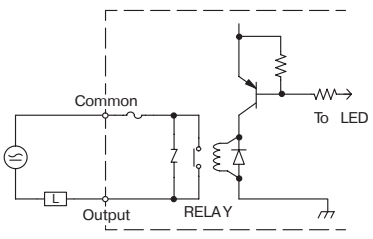
Число выходов	16 реле
Число общих проводов	2 (изолированные)
Рабочее напряжение	=5-30В / ~5-250В
Тип вывода	Реле, тип A однополюсное на одно направление, нормально разомкнутое (SPST-NO)
Наибольшее кратковременное напряжение	=30В / ~256В
Сигнал переменного тока	47 - 63 Гц
Падение напряжения во включенном состоянии	Нет
Максимальный ток (активная нагрузка)	1А/канал 5А модуль (активная нагрузка)
Максимальная утечка тока	0.1мА при ~265В
Кратковременная перегрузка	4А
Минимальная нагрузка	5мА
Потребляемый от каркаса ток 60мА/канал	макс. 1000мА,
Требуемое внешнее питание	нет
Время перехода из «Выкл» → «Вкл»	10 мсек.
Время перехода из «Вкл» → «Выкл»	10 мсек.
Клеммный блок	съемный
Индикаторы состояния	есть, на логической части
Вес	310 гр.
Предохранители	1 (8А), несъемный,

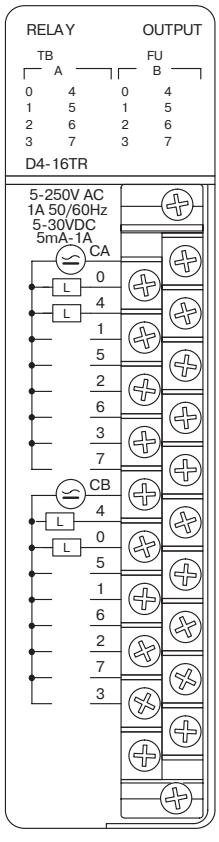
Стандартный ресурс реле (1000 операций)

Максимальная кратковременная активная или индуктивная нагрузка	Рабочее напряжение		
	30В	125В	250В
1А активная	>1M	500K	300K
1А индуктивная	400K	200K	100K
0.5А активная	2M	800K	500K
0.5А индуктивная	1M	300K	200K







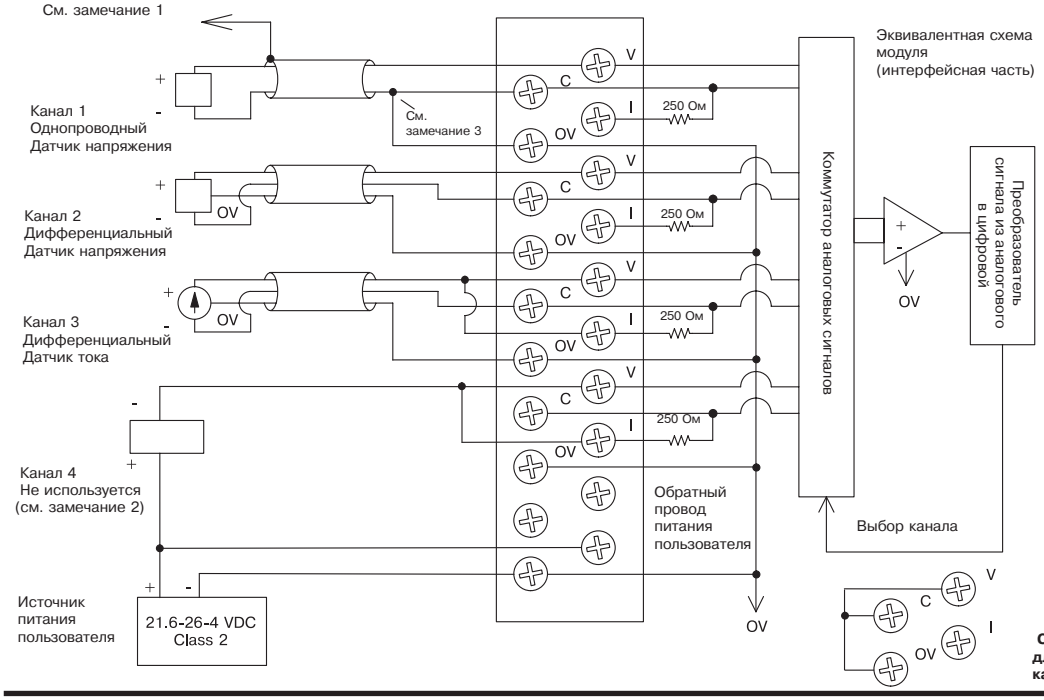


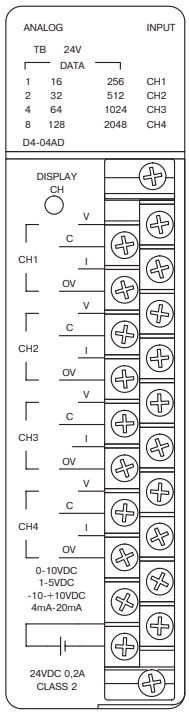
F4-04AD 4-канальный модуль аналоговых входов

Число каналов	4
Тип входов	С общим проводом или дифференциальные напряжения, токовые
Диапазоны входных сигналов	0 - 5В, 1 - 5В, 0 - 10В, ±5В, ±10В, 0 - 20 мА, 4 - 20 мА
Число каналов, конфигурируемых индивидуально	Диапазон выбирается одновременно для всех каналов. Однако каждый канал может быть присоединен как вход напряжения или как токовый вход.
Разрешение	12 бит (0 из 4095) однополярный режим 13 бит (-4095 - +4095) биполярный режим
Входное сопротивление напряжения	мин. 20 МОм для входов 250Ом, 1/2Вт, ±0.1%, $\pm 25 \bullet 10^{-6}$/°C для токовых входов
Максимальная длительная перегрузка напряжения, напряжения	±50В, вход ±45мА токовый вход
Рекомендуемый внешний предохранитель	0.032А, Серии 217 быстродействующие, на токовые входы
Максимальное напряжение	макс. ±10В
Ошибка линеаризации	макс. ±0.025% диапазона (±1 импульс счета в однополярном режиме)
Входная стабильность	±1/2 импульса счета
Перекрестная наводка	-80дБ, макс. 1/2 имп. счета
Полная величина ошибки	±12 импульсов счета, входы напряжения ±16 импульсов счета, токовые входы при 20мА

Ошибка смещения	±1 импульс счета, входы напряжения ±2 импульса счета, токовые входы при 4мА
Наибольшая погрешность	±0.4% при 25°C ±0.55% при 0 - 60°C
Время преобразования	< 6 мсек. / канал
Ослабление шума по сигналу: по общему проводу:	 -3дБ/50Гц, -6дБ/октаву -70дБ /12КГц
Скорость обновления каналов контроллера	1 канал за один цикл сканирования, макс. - 4 канала
Число используемых	16(X) входных каналов (12 двоичных бит данные, 2 канала для битов идентификатора, 1 для знака, 1 - разрыва цепи датчика) Дополнительно для совместимости с D4-04AD 32 операции ввода канала
Потребляемый от каркаса ток (5В)	85мА
Параметры внешнего питания	24В, ±10%, 100мА, класс 2
Точность в зависимости от температуры	макс. $\pm 45 \bullet 10^{-6}$/°C от величины полного диапазона (включая максимальное изменение смещения равное 2 импульсам)
Рабочая температура	0 - 60°C

Замечание 1: Оплетка должна быть заземлена у источника сигнала.
Замечание 2: Неиспользуемые каналы должны быть закорочены с целью улучшения защиты от шума.
Замечание 3: Если дифференциальный вход не используется, то контакт 0В должен быть закорочен с контактом С соответствующего канала.





Соединение клемм для неиспользуемых каналов (см. Замечание 2)

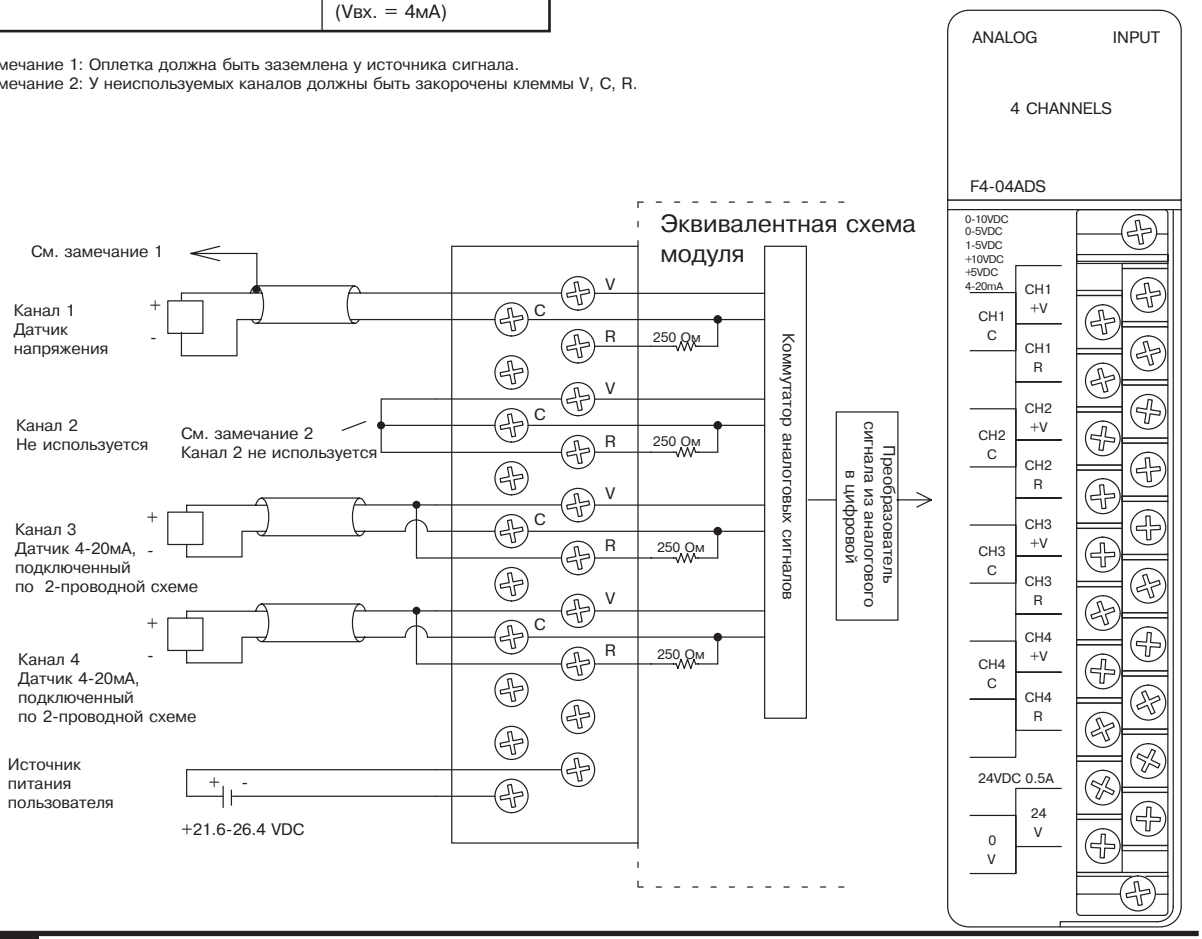
F4-04ADS 4-канальный модуль изолированных аналоговых входов

Число каналов	4
Диапазоны входных 5В, сигналов	0 - 5В, 0 - 10В, 1 - 5В, ±5В, ±10В, 0 - 20 мА, 4 - 20 мА
Возможность индивидуального конфигурирования каналов	Да
Разрешение	12 бит (1 из 4096)
Способ преобразования сигнала	Последовательное приближение
Тип входов	Дифференциальные
Максимальное напряжение	±750В выдерживаемое изоляцией трансформатора
Ослабление шума	по общему проводу: -100дБ при 60Гц
Активная низкочастотная фильтрация	-3дБ при 20Гц, -12дБ на октаву
Входное сопротивление	200 КОм для входов напряжения 250Ом, 1/2Вт, ±0.1% для токовых входов
Максимальные значения характеристик	-45мА - +45мА, токовый вход ±100В вход напряжения
Время преобразования	1 мсек./канал
Ошибка линеаризации в однополярном режиме	макс. ±1 импульс счета (±0.025% диапазона)
В биполярном режиме	макс. ±2 импульса счета (±0.025% диапазона)
Полная величина ошибки калибровки	макс. ±8 импульсов счета (Vвх. = 20мА)
Ошибка калибровки смещения	макс. ±8 импульсов счета (Vвх. = 4мА)

Скорость обновления каналов контроллера	1 канал за один цикл сканирования
Число используемых	16(X) входных каналов 12 двоичных бит данных, 4 бита - признак активного состояния канала
Точность в зависимости от температуры	макс. $\pm 100 \bullet 10^{-6}$/°C от величины полного диапазона (включая максимальное изменение смещения)
Потребляемый от каркаса ток (5В)	270мА
Параметры внешнего питания	24В, ±10%, 100мА, класс 2
Рекомендуемый внешний предохранитель	0.032А, Серии 217 быстродействующие, на токовые входы
Рабочая температура	0 - 60°C

В таблице характеристик один импульс счета соответствует нижнему значащему биту аналоговой величины (1 из 4096)

Замечание 1: Оплетка должна быть заземлена у источника сигнала.
Замечание 2: У неиспользуемых каналов должны быть закорочены клеммы V, C, R.



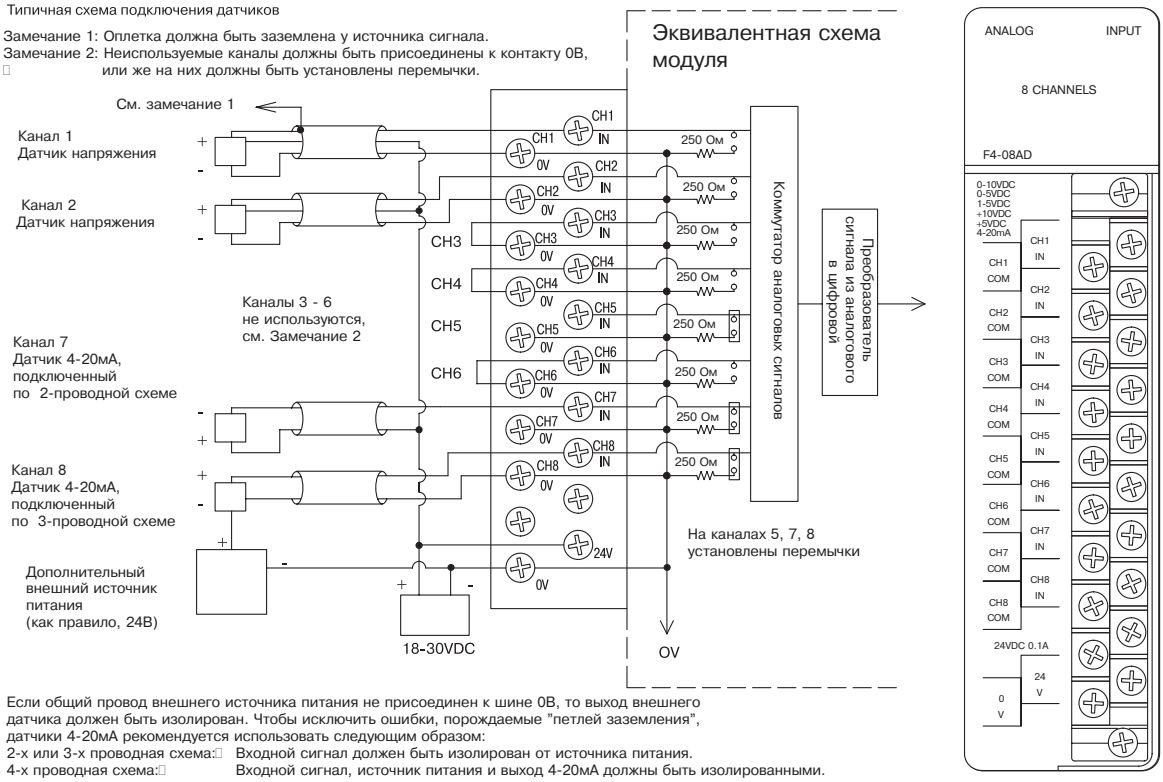
F4-08AD 8-канальный модуль аналоговых входов

Число каналов	8
Тип входов	С общим проводом, дифференциальные напряжения, токовые
Диапазоны входных сигналов	0 - 5В, 0 - 10В, 1 - 5В, ±5В, ±10В, 0 - 20 мА, 4 - 20 мА
Возможность индивидуального конфигурирования каналов	Диапазон выбирается одновременно для всех каналов. Однако каждый канал может быть подсоединен как вход напряжения или как токовый вход.
Разрешение	12 бит (1 из 4096)
Активная низкочастотная фильтрация	-3дБ при 20Гц, -12дБ на октаву
Входное сопротивление	> 20 МОм для входов напряжения 1МОм мин. 250Ом, 1/2Вт, ±0.1%, 25u10-6/°C для токовых входов
Время преобразования	0.4 мсек./канал (время преобразования модулем), 1 мсек./канал (время процессора)
Ошибка линеаризации (при сквозной передаче данных)	макс. ±1 импульс счета (0.025% от истинной величины)
Входная стабильность	±1/2 импульс счета
Полная величина ошибки	±12 импульсов счета, входы напряжения ±12 импульсов счета, токовые входы при 20мА
Ошибка смещения	±2 импульса счета, однополюсные входы напряжения ±4 импульса счета, двухполюсные входы напряжения ±4 импульса счета, токовые входы при 4мА
Скорость обновления каналов контроллера	мин. 1 канал за один цикл сканирования, макс. - 8 каналов
Число используемых	16(X) входных канала (12 двоичных бит данных, 3 бита - признак активного состояния канала, 1 бит - неиспользуемые)
Потребляемый от каркаса ток	75мА
Параметры внешнего питания	8-30, 90мА, класс 2
Рекомендуемый внешний предохранитель	0.032А, Серии 217 быстродействующие, на токовые входы
Рабочая температура	0 - 60°C
Точность в зависимости от температуры	макс. ±100u10-6/°C от величины полного диапазона (включая максимальное изменение смещения)

В таблице характеристик один импульс счета соответствует нижнему значащему биту аналоговой величины (1 из 4096)

Типичная схема подключения датчиков

Замечание 1: Оплетка должна быть заземлена у источника сигнала.
Замечание 2: Неиспользуемые каналы должны быть присоединены к контакту 0В, или же на них должны быть установлены перемычки.



Если общий провод внешнего источника питания не присоединен к шине 0В, то выход внешнего датчика должен быть изолирован. Чтобы исключить ошибки, порождаемые "петлей заземления", датчики 4-20мА рекомендуется использовать следующим образом:
2-х или 3-х проводная схема: Входной сигнал должен быть изолирован от источника питания.
4-х проводная схема: Входной сигнал, источник питания и выход 4-20мА должны быть изолированными.

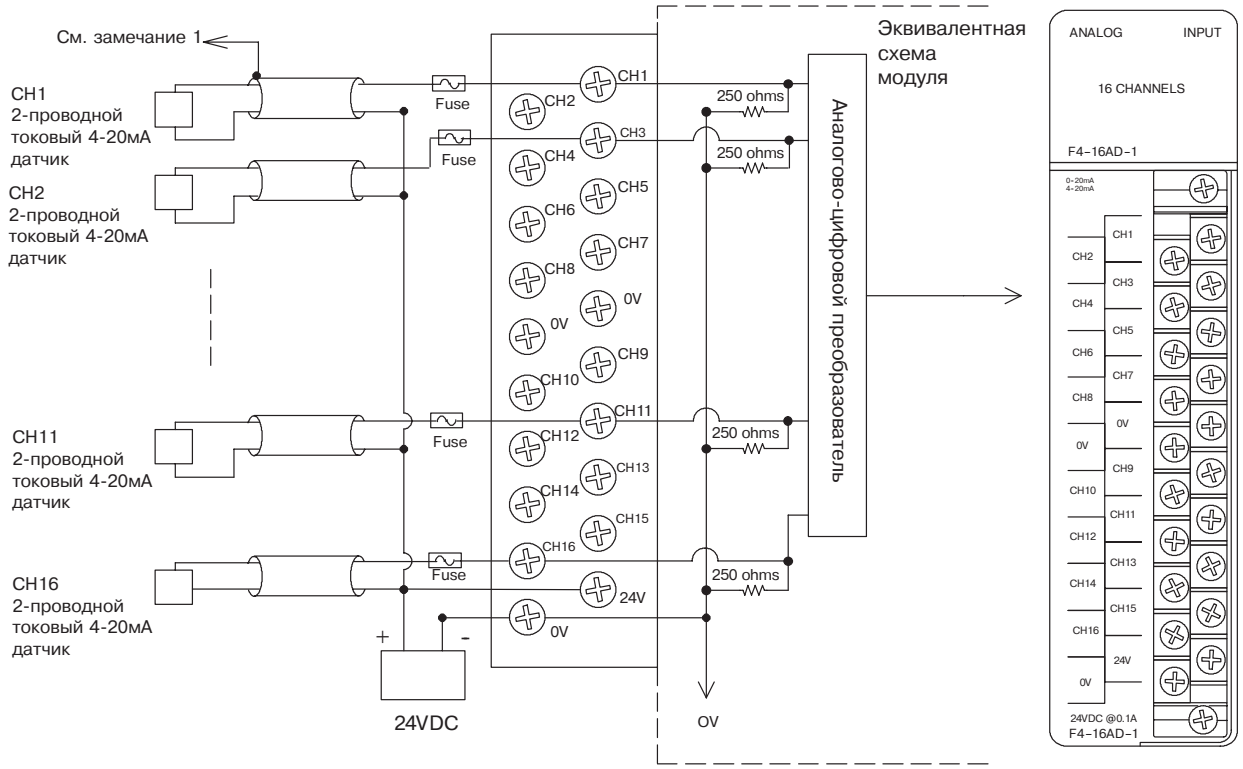
F4-16AD-1 16-канальный аналоговый входной модуль

Число каналов	16, несимметричных (один общий)
Диапазоны входных сигналов	0-20 мА, 4-20 мА
Разрешение	12 бит(1 из 4096)
Активный низкочастотный фильтр	-3 dB при 20Гц, -6 dB на октаву
Входное сопротивление	250 Ом ± 0.1%, 1/2 Вт на токовый вход
Абсолютное максимальное значение	± 45 мА, входного тока
Время преобразования	2.0 мс на канал
Ошибка лианеризации	Максимум ± 2 единицы отсчета
Стабильность входа	± 1 единица отсчета
Ошибка калибровки полного диапазона (Ошибка смещения не включена)	Максимум ± 12 единиц отсчета, при входном токе 20мА
Ошибка смещения калибровки	Максимум ± 3 единицы отсчета, при входном токе 4мА

Одна единица отсчета в таблицах характеристик соответствует одному значащему биту аналоговой величины (1 из 4096)

Скорость обновления в PLC сканирования процессора	Максимум 16 каналов за один цикл
Требуемые количество точек дискретного ввода	Всего 16 (X) входных точек, из них 12 бит двоичных данных, 4 бита активного канала
Требуемая мощность	75 мА (питание от каркаса)
Внешнее питание	= 21.6-26.4 В, 100 мА, класс 2
Рекомендуемые предохранители	0.032 А, Серии 217, быстродействующие для токовых входов
Точность в зависимости от температуры	Максимум ± 50 ppm/°C на полном диапазоне (включая максимальную ошибку смещения равную 3 единицам отсчета)
Рабочая температура	От 0 до 60°C
Температура хранения	От -20 до 70°C
Влажность	От 5 до 95% (без конденсата)

ПРИМЕЧАНИЕ 1: Экраны должны быть заземлены на стороне источника сигнала



Рекомендуется использовать быстродействующий плавкий предохранитель серии 217,для токовых входов на 0.032А.
Если общий провод источника питания не подключен к 0V на модуле, то выход внешнего датчика должен быть изолирован.
Чтобы избежать ошибок в контуре заземления рекомендуется использовать следующие типы датчиков на 4-20мА:

- 2 или 3-проводные с изоляцией между входным сигналом и питанием.
- 4-проводные с изоляцией между входным сигналом, питанием и выходом 4-20мА.

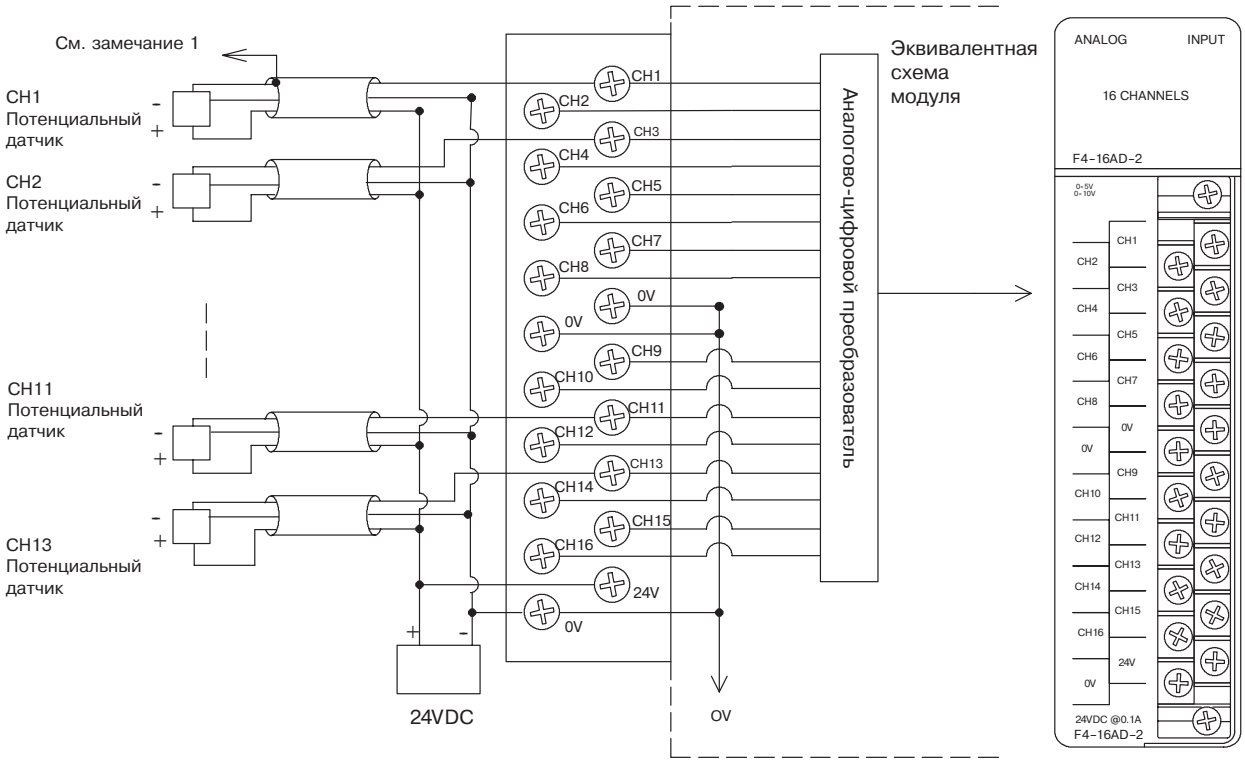
F4-16AD-2 16-канальный аналоговый входной модуль

Число каналов	16, несимметричных (один общий)
Диапазоны входных сигналов	0-5В, 0-10В
Разрешение	12 бит(1 из 4096)
Активный низкочастотный фильтр	-3 dB при 20Гц, -6 dB на октаву
Входное сопротивление	Минимум 100 кОм
Абсолютное максимальное значение	± 45 мА, входного тока
Время преобразования	0.4 мс на канал (для всего модуля) минимум 2 мс на подключенный канал (процессор)
Ошибка лианеризации	Максимум ±2 единицы отсчета (0.05% от полного диапазона)
Стабильность входа	± 1 единица отсчета
Ошибка калибровки полного диапазона (Ошибка смещения не включена)	Максимум ± 12 единиц отсчета
Ошибка смещения калибровки	Максимум ±3 единицы отсчета, при однополярном напряжении

Скорость обновления в PLC	Максимум 16 каналов за один цикл сканирования процессора
Требуемые количество точек дискретного ввода	Всего 16 (X) входных точек, из них 12 бит двоичных данных, 4 бита активного канала
Требуемая мощность	75 мА (питание от каркаса)
Внешнее питание	= 21.6-26.4 В, 100 мА, класс 2
Точность в зависимости от температуры	Максимум ± 50 ppm/°C на полном диапазоне (включая максимальную ошибку смещения равную 3 единицам отсчета)
Рабочая температура	От 0 до 60°C
Температура хранения	От -20 до 70°C
Влажность	От 5 до 95% (без конденсата)

Одна единица отсчета в таблицах характеристик соответствует одному значащему биту аналоговой величины (1 из 4096)

ПРИМЕЧАНИЕ 1: Экраны должны быть заземлены на стороне источника сигнала.



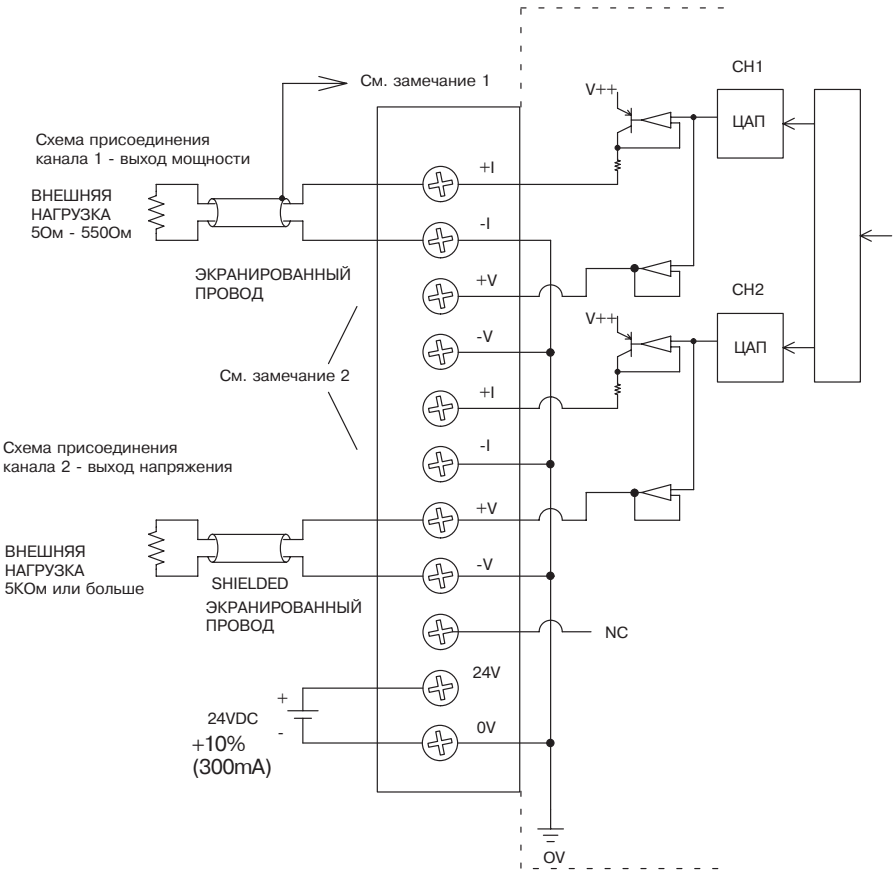
Если общий провод источника питания не подключен к 0V на модуле, то выход внешнего датчика должен быть изолирован.

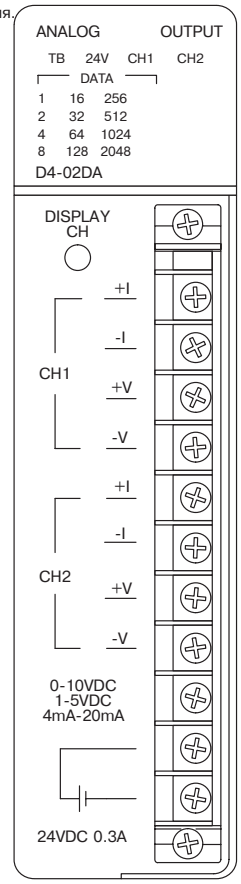
D4-02DA 2-канальный модуль аналоговых выходов

Число каналов	2 (независимые)
Диапазоны выходных сигналов	0 - 10В, 1 - 5В, 4 - 20 мА
Возможность индивидуального конфигурирования каналов	Диапазон определяется способом присоединения каждого канала
Разрешение	12 бит (1 из 4096)
Тип вывода	С общим проводом
Сопротивление выходов	макс. 0.5Ом, выход напряжения
Ток на выходе	макс. 5мА, выход напряжения
Сопротивление нагрузки	макс. 550Ом, мин. 5.0Ом, выход мощности, мин. 2КОм, выход напряжения
Ошибка линеаризации	макс. 0.1%
Точность в зависимости от температуры	макс. $\pm 70 \bullet 10^{-6}$/°C
Наибольшая погрешность	±0.2% при 25°C
Время преобразования	С момента начала очередного шага сканирования, 30мксек. + продолжительность одного шага сканирования

Скорость обновления каналов контроллера	1 или 2 канала за один цикл сканирования
Занятые цифровые выходы	32(Y) выходных каналов
Информация на выходах	два раза по 12 двоичных бит данных, 8 неиспользуемых бит
Потребляемый от каркаса ток (5В)	250мА
Параметры внешнего питания	24В, ±10%, 300мА, класс 2
Рабочая температура окружающей среды	0 - 60°C

Замечание 1: Оплетки экранированных кабелей должны быть присоединены к контакту 0В модуля или к 0В источника питания.
Замечание 2: Неиспользуемые токовые выходы / выходы напряжения должны быть разомкнуты (соединения отсутствуют).





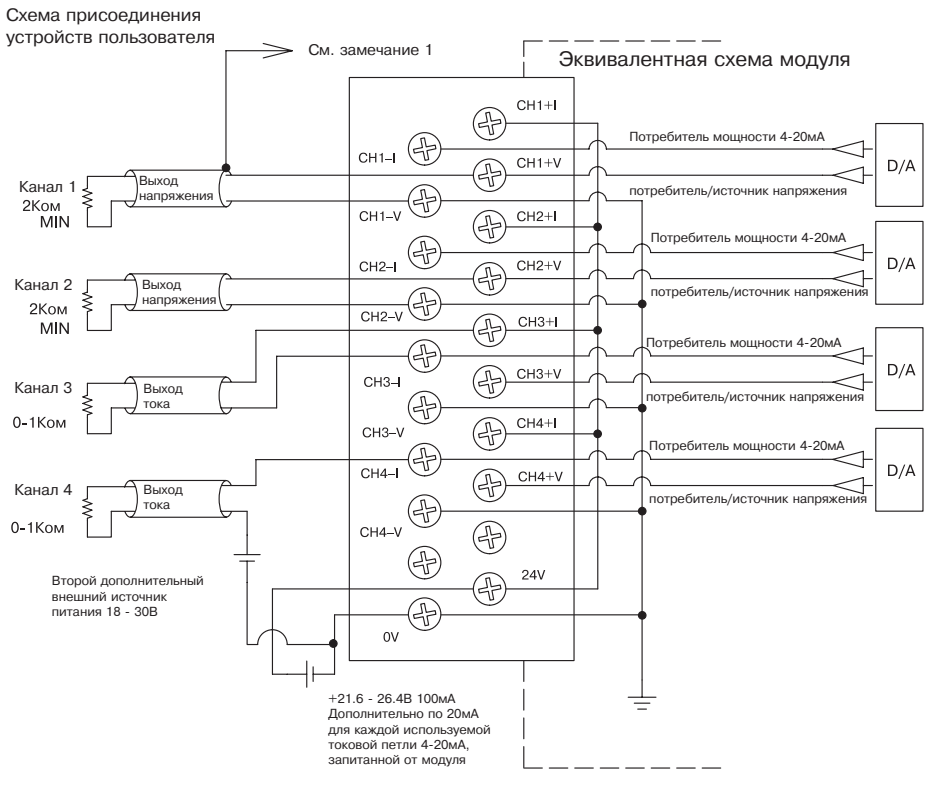
F4-04DA 4-канальный модуль аналоговых входов

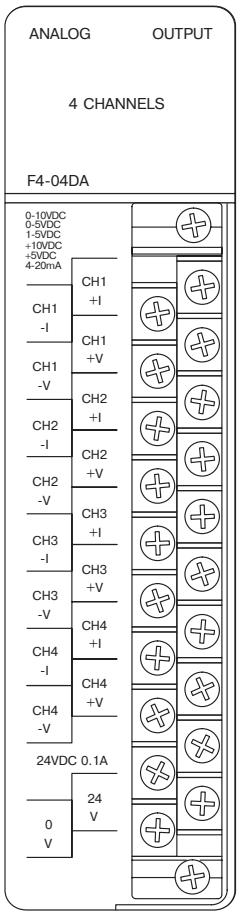
Число каналов	4
Диапазоны выходных сигналов	0 - 5В, 0 - 10В, ±5В, ±10В, 4 - 20 мА
Возможность индивидуального конфигурирования каналов	Да*
Разрешение	2 бит (1 из 4096)
Способ преобразования сигнала	Последовательное приближение
Тип вывода	С общим проводом
Сопротивление выходов	0.2Ом станд., выход напряжения
Сопротивление нагрузки	мин. 0 Ом, выход токовый, мин. 2КОм, выход напряжения
Максимальная нагрузка/нпряжения	680Ом/18В, 1КОм/24В, 1.5КОм/36В, выход токовый
Ток на выходе напряжения	5мА, потребитель или источник
Ток короткого замыкания	15мА стандартно, выход напряжения
Ошибка линеаризации	макс. ±1 импульс счета (0.025%)
Коэффициент усиления ошибки калибровки	макс. ±8 импульсов счета, выход напряжения
мощности	макс. -8 - +11 импульсов счета, выход
Ошибка калибровки смещения счета,	макс. ±2 импульса
	выход напряжения макс. -5 - +9 импульсов счета, выход мощности

Время преобразования	макс. 5мксек., время установки 0.3 мсек. макс. при преобразовании цифрового выхода в аналоговый выход
занятые цифровые выходы	16(Y) выходных каналов (12 бит данных и 4 бита выбора канала)
Потребляемый от каркаса ток (5В)	120мА
Параметры внешнего питания	+24В(±10%), 100мА, класс 2 (+ по 20мА для каждой используемой токовой петли)
Точность в зависимости от температуры	макс. $\pm 50 \bullet 10^{-6}$/°C полного диапазона макс. $\pm 25 \bullet 10^{-6}$/°C величины смещения
Рабочая температура	0 - 60°C

В таблице характеристик один импульс счета соответствует нижнему значащему биту аналоговой величины (1 из 4096)

* В любой из возможных комбинаций недопустимо использование режимов 4 - 20 мА и -10В - +10В.
Замечание 1: Оплетки экранированных кабелей должны быть присоединены к контакту 0В модуля или к 0В источника питания.
Замечание 2: Неиспользуемые токовые выходы и выходы напряжения должны быть разомкнуты (соединения отсутствуют).





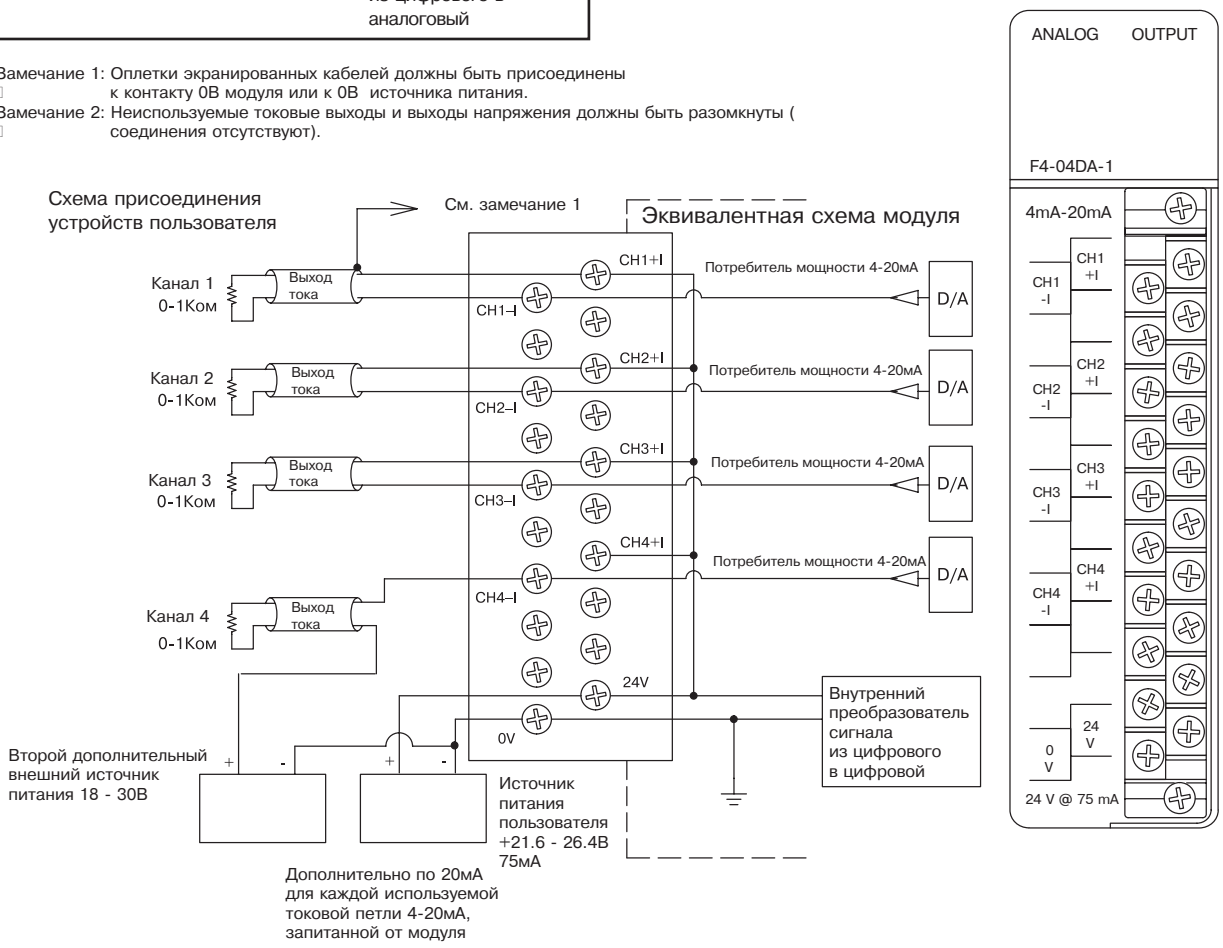
F4-04DA-1 4-канальный модуль токовых аналоговых выходов

Число каналов	4, с общим проводом (один общий провод)
Диапазон выходных сигналов	4 - 20 мА
Разрешение	12 бит (1 из 4095)
Тип вывода	Выходы 4 - 20 мА, с питанием от внешнего источника (макс. =30В)
Сопротивление выходов	мин. 0 Ом
Наибольшее кратковременное напряжение на выходе	=40В (фиксированное, подавитель помех)
Максимальная нагрузка/Параметры питания	620Ом/18В, 910Ом/24В, 1.2КОм/30В
Ошибка линеаризации (наилучшее приближение)	макс. ±1 импульс счета (0.025%)
Коэффициент усиления ошибки калибровки	макс. ±5 импульсов счета
Ошибка калибровки смещения счета	макс. ±3 импульса
Наибольшая погрешность	±0.1% при 25°C ±0.3% при 0 - 60°C
Время преобразования время	макс. 100мксек. - установки макс. 2.0 мсек. - преобразование сигнала из цифрового в аналоговый

Занятые цифровые выходы	16(Y) выходных каналов (12 бит данных и 4 бита - признак активного канала)
Потребляемый от каркаса ток (5В)	70мА
Параметры внешнего питания	=21.6 - 26.4В, 75мА, класс 2 (+ по 20мА для каждой используемой токовой петли)
Точность в зависимости от температуры	макс. $\pm 57 \bullet 10^{-6}$/°C полного диапазона (включая максимальное изменение смещения, 2 импульса)
Рабочая температура	0 - 60°C

В таблице характеристик один импульс счета соответствует нижнему значащему биту аналоговой величины (1 из 4096)

Замечание 1: Оплетки экранированных кабелей должны быть присоединены к контакту 0В модуля или к 0В источника питания.
Замечание 2: Неиспользуемые токовые выходы и выходы напряжения должны быть разомкнуты (соединения отсутствуют).



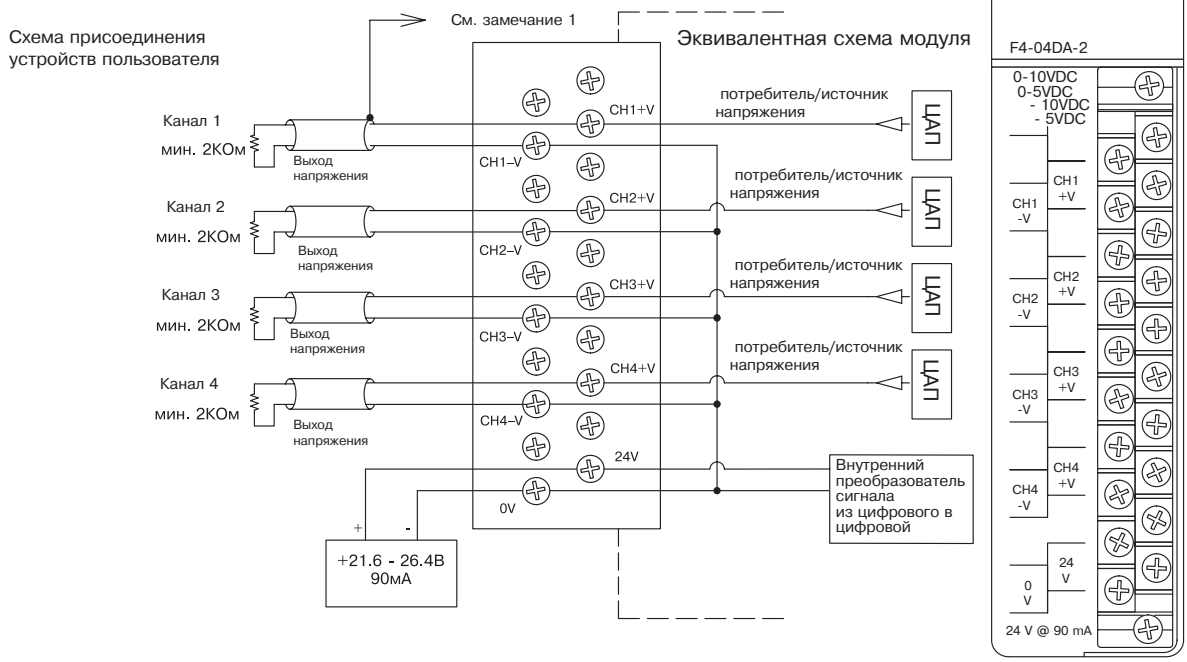
F4-04DA-2 4-канальный модуль аналоговых выходов напряжения

Число каналов	4, с общим проводом (один общий провод)
Диапазон выходных сигналов	0 - 5В, 0 - 10В, ±5В, ±10В
Возможность индивидуального конфигурирования каналов	Да
Разрешение	12 бит (1 из 4095)
Сопротивление выходов	мин. 2КОм
Емкость нагрузки	не более 0.01 мкФ
Ток на выходе	макс. 5мА, потребитель или источник
Ток короткого замыкания	15мА стандартно
Ошибка линеаризации (при сквозной передаче данных) и относительная точность	± 1 импульс счета (макс. 0.025%)
Ошибка калибровки смещения	макс. ±3 импульса счета при униполярном режиме, макс. ±4 импульса счета при биполярном режиме
Полная величина ошибки калибровки	макс. ±8 импульсов счета (ошибка калибровки включена)
Наибольшая погрешность	±0.2% при 25°C ±0.4% при 0 - 60°C

В таблице характеристик один импульс счета соответствует нижнему значащему биту аналоговой величины (1 из 4096)

Время преобразования	макс. 5мксек., время установки 2.0 мсек. при преобразовании цифрового выхода в аналоговый выход
Занятые цифровые выходы	16(Y) выходных каналов (12 бит данных, 4 или бита признак активности канала, один бит - признак биполярого режима)
Потребляемый от каркаса ток (5В)	90мА
Параметры внешнего питания	+21.6 - 26.4В, 90мА, класс 2 (при полной нагрузке на каналах)
Точность в зависимости от температуры	макс. $\pm 57 \bullet 10^{-6}$/°C полного диапазона (включая максимальное изменение смещения, 2 импульса)
Рабочая температура	0 - 60°C
Температура хранения	-20 - 70°C
Относительная влажность	5 - 95% (без конденсации)
Атмосфера	Отсутствие агрессивных газов
Устойчивость к вибрации	Стандарт MIL 810C, метод 514.2
Устойчивость к удару	Стандарт MIL 810C, метод 516.2
Шумоустойчивость	NEMA (ICS3-304)

Замечание 1: Оплетки экранированных кабелей должны быть присоединены к контакту 0В модуля или к 0В источника питания.
Замечание 2: Неиспользуемые токовые выходы и выходы напряжения должны быть разомкнуты (соединения отсутствуют).



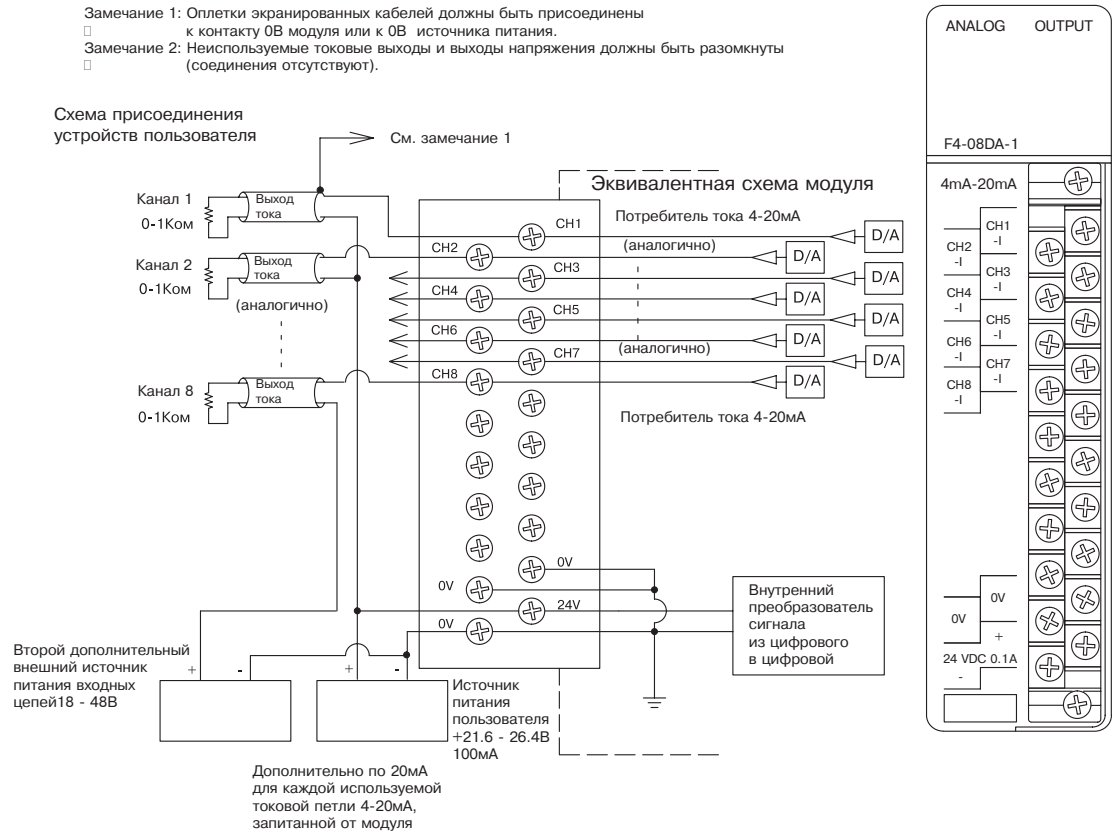
F4-08DA-1 8-канальный модуль токовый аналоговых выходов

Число каналов	8, с общим проводом (один общий провод)
Диапазон выходных сигналов	4 - 20 мА
Разрешение	12 бит (1 из 4095)
Тип вывода	Токовый 4 - 20 мА, с питанием от внешнего источника
Наибольшее кратковременное напряжение на выходе	=40В (подавитель импульсных помех отсутствует)
Параметры нагрузки	0 - 480 Ом/18В, 220-740 Ом/24В, 1550-1760 Ом/48В
Максимальное напряжение питания	48В (Сопротивление нагрузки в допустимом диапазоне)
Перекрестная помеха	макс. -70дБ, ±1 импульс счета
Ошибка линеаризации (при сквозной передаче данных) и относительная точность	макс. ±1 импульс счета
Полная величина ошибки калибровки (ошибка калибровки включена)	макс. ±8 импульсов счета
Ошибка калибровки смещения	макс. ±3 импульса счета (4мА при 25°С)
Наибольшая погрешность	±0.2% при 25°С ±0.4% при 0 - 60°С

Время преобразования	макс. 400мксек. при изменении сигнала в полном диапазоне 2.25 - 4.5 мсек. - преобразование сигнала из цифрового в аналоговый
Занятые цифровые выходы	16(Y) выходных каналов (12 бит данных, 3 бита - признак выбора канала, 1 бит - признак активности вывода)
Потребляемый от каркаса ток (5В)	90мА
Параметры внешнего питания	21.6 - 26.4В, 75мА, класс 2 (+ по 20мА для каждой используемой токовой петли)
Точность в зависимости от температуры	макс. $\pm 57 \bullet 10^{-6}$/°С полного диапазона (включая максимальное изменение смещения - 2 импульса)
Рабочая температура	0 - 60°С

В таблице характеристик один импульс счета соответствует нижнему значащему биту аналоговой величины (1 из 4096)

Замечание 1: Оплетки экранированных кабелей должны быть присоединены к контакту 0В модуля или к 0В источника питания.
Замечание 2: Неиспользуемые токовые выходы и выходы напряжения должны быть разомкнуты (соединения отсутствуют).



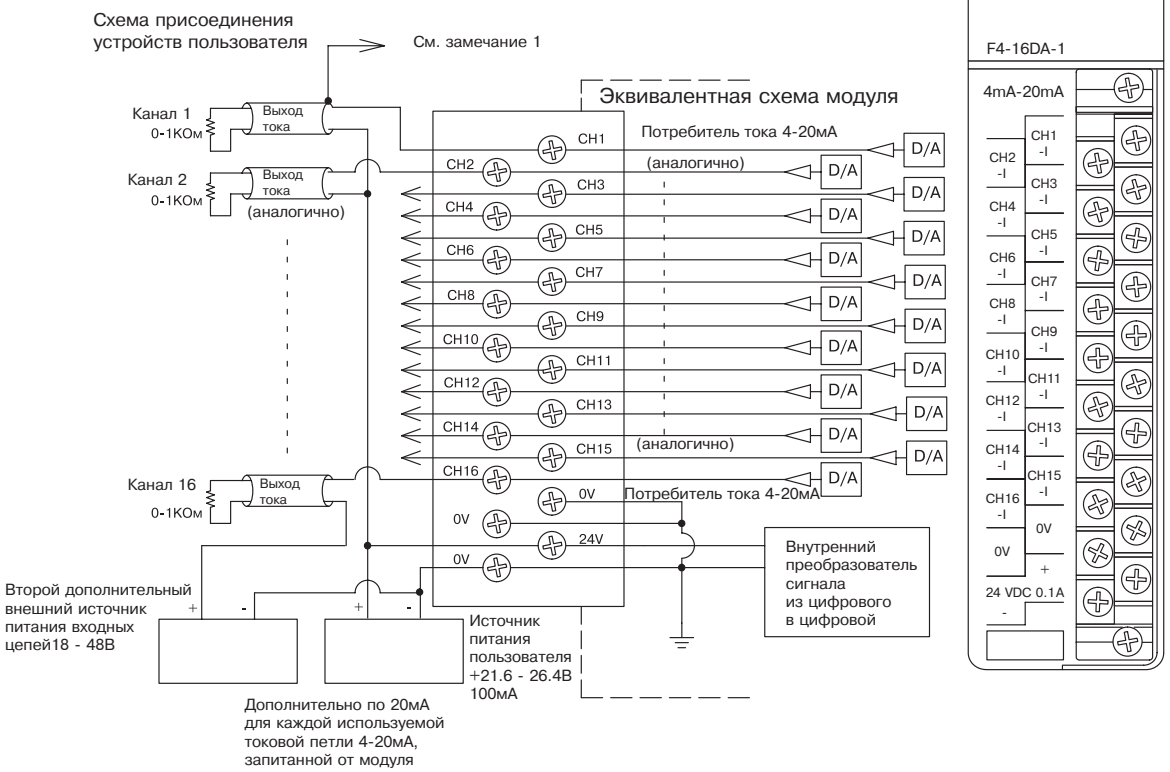
F4-16DA-1 16-канальный модуль аналоговых выходов тока

Число каналов проводом	16, с общим (один общий провод)
Диапазон выходных сигналов	4 - 20 мА
Разрешение	12 бит (1 из 4095)
Тип вывода	Выходы источников тока 4 - 20 мА, с питанием от внешнего источника
Наибольшее кратковременное напряжение на выходе	40В (подавитель импульсных помех отсутствует)
Параметры нагрузки	0-480 Ом/18В, 220-740 Ом/24В, 1550-1760 Ом/48В
Максимальное напряжение питания	48В (Сопротивление нагрузки в допустимом диапазоне)
Перекрестная помеха	макс. -70дБ, ±1 импульс счета
Ошибка линеаризации (при сквозной передаче данных) и Относительная точность	макс. ±1 импульс счета (20мА при 25°С)
Полная величина ошибки калибровки (ошибка калибровки включена)	макс. ±8 импульсов счета (20мА при 25°С)
Ошибка калибровки смещения	макс. ±3 импульса счета (4мА при 25°С)
Наибольшая погрешность	±0.2% при 25°С ±0.4% при 0 - 60°С

Время преобразования	макс. 400мксек. при изменении сигнала в полном диапазоне 4.5 - 9 мсек. - преобразование сигнала из цифрового в аналоговый
Занятые цифровые выходы каналов	32(Y) выходных (два блока по: 12 бит данных, 3 бита - признак выбора канала, 1 бит - признак активности вывода)
Потребляемый от каркаса ток (5В)	90мА
Параметры внешнего питания	21.6 - 26.4В, 75мА, класс 2 (+ по 20мА для каждой используемой токовой петли)
Точность в зависимости от температуры	макс. $\pm 57 \bullet 10^{-6}$/°С полного диапазона (включая максимальное изменение смещения - 2 импульса)
Рабочая температура	0 - 60°С

В таблице характеристик один импульс счета соответствует нижнему значащему биту аналоговой величины (1 из 4096)

Замечание 1: Оплетки экранированных кабелей должны быть присоединены к контакту 0В модуля или к 0В источника питания.
Замечание 2: Неиспользуемые токовые выходы и выходы напряжения должны быть разомкнуты (соединения отсутствуют).



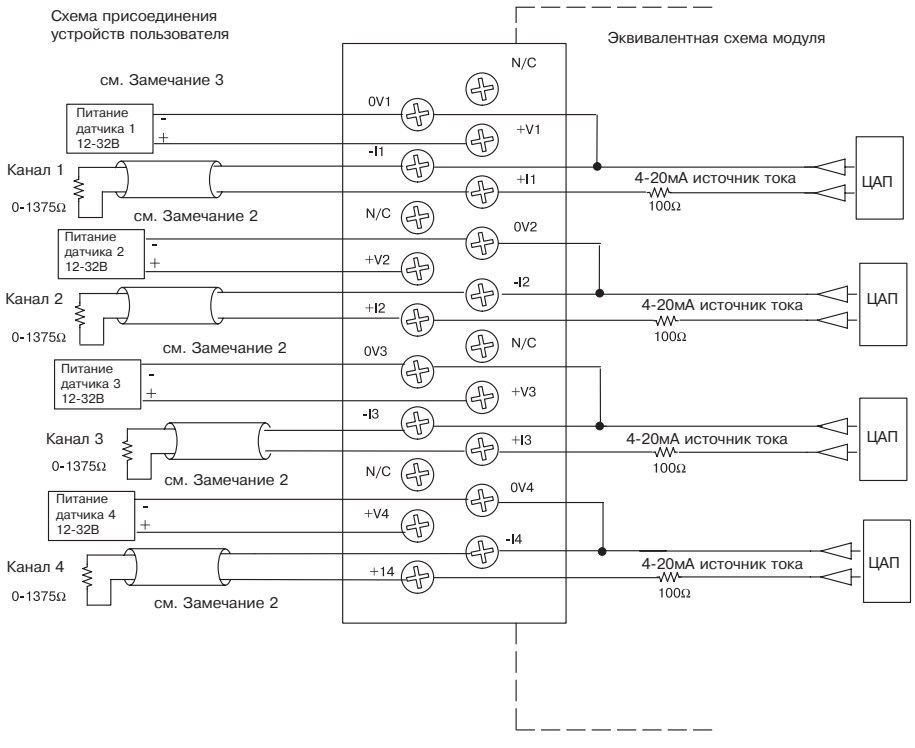
F4-04DAS-1 4-канальный модуль изолированных аналоговых выходов тока

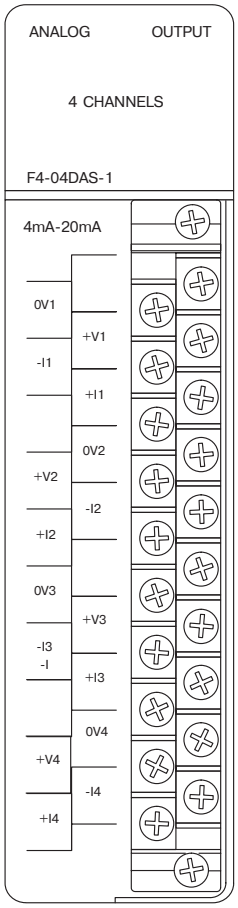
Число каналов	4, изолированные выходы тока
Диапазон выходных сигналов	4 - 20 мА
Разрешение	16 бит (1 из 65536)
Тип вывода источника	Токовый 4 - 20 мА, с питанием от внешнего
Макс. напряжения, выдерживаемое изоляцией	±750В в течение длительного времени, между каналами, между каналом и логической частью
Параметры питания цепей	=12 - 32В
Внутреннее падение напряжения выходной цепи	2.5В
Сопротивление нагрузки	0 - 1375Ом (при 32В)
Максимальная нагрузка/ параметры питания	375Ом/12В, 975Ом/24В, 1375Ом/32В
Скорость обновления каналов контроллера	мин. 1, макс 4 канала за один цикл сканирования
Занятые цифровые выходы каналы	32(Y) выходные
Информация на выходах	16 двоичных бит данных, 2 канала для битов идентификатора, 1 бит - признак активности выхода
Потребляемый от каркаса ток	60мА при 5В
Параметры внешнего питания	50мА / канал

Ошибка линеаризации (при сквозной передаче данных)	макс. ±10 импульсов счета (0.015% от истинной величины)
Продолжительность переходного процесса	3 мсек для 0.1% от полного диапазона
Коэффициент усиления ошибки калибровки	±32 импульса счета (±0.05%)
Смещение ошибки калибровки	±13 импульсов счета (±0.02%)
Уход величины на выходе	$50 \bullet 10^{-6}$/°C
Наибольшая погрешность	±0.07% при 25°C ±0.18% при 0 - 60°C
Рабочая температура окружающей среды	0 - 60°C

В таблице характеристик один импульс счета соответствует нижнему значащему биту аналоговой величины (1 из 4096)

Замечание 1: Оплетки экранированных кабелей должны быть присоединены к контакту 0В модуля.
Замечание 2: Параметры нагрузки должны соответствовать параметрам питания цепей.
Замечание 3: При работе с неизолированными выходами необходимо соединить:
- между собой все контакты 0V (0V1 ... 0V4);
- между собой все контакты +V (+V1 ... +V4).



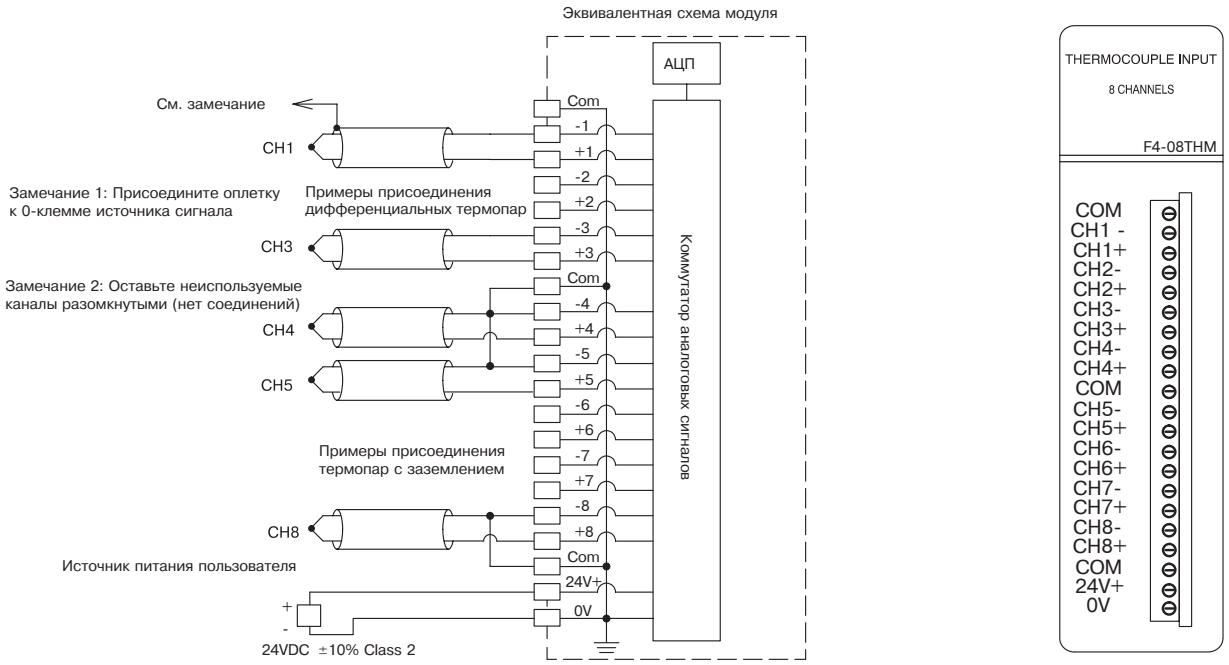


F4-08THM 8-канальный модуль входов термопар

Число каналов	8, дифференциальные
Диапазон сигнала в нормальном режиме	±5В
Ослабление синфазного сигнала	мин. 90дБ для постоянного тока, мин. 150дБ для переменного тока 50/60Гц
Входное сопротивление	1МОм
Максимальный сигнал	Защита входов при ±50В
Точность в зависимости от температуры	макс. $\pm 5 \bullet 10^{-6}$/°C от величины диапазона (включая максимальное изменение смещения)
Скорость обновления каналов контроллера	макс. 8 каналов за один цикл сканирования
Занятые цифровые входы	16 двоичных бит данных, 2 канала для битов идентификатора, 4 бита диагностики
Необходимые входные каналы	32 (X) модуль входов
Параметры внешнего питания	макс. 60мА, 18 - 26.4В
Потребляемый от каркаса ток	макс. 110мА, 5В
Рабочая температура окружающей среды	0 - 60°C

Диапазон сигнала на входах*	Тип J -190 - 760°C Тип E -210 - 1000°C Тип K -150 - 1372°C Тип R 65 - 1768°C Тип S 65 - 1768°C Тип T -230 - 400°C Тип B 529 - 1820°C Тип N -70 - 1300°C Тип C 65 - 2320°C
Разрешение на выходе	±0.1°C
Компенсация холодного спая	Автоматическая
Время прогрева	30 мин. стандартное, (повторяемость ±1°C)
Ошибка линеаризации (сквозная)	±0.05°C макс., ±0.01°C стандартная
Наибольшая погрешность	±3°C (без учета погрешности термопар)
Характеристики входов напряжения	
Диапазон сигнала на входах	0 - 5В, ±5В, 0-156.25мВ, ±156.25мВ
Разрешение	16 бит (1 из 65535)
Полная величина ошибки калибровки (ошибка смещения учтена)	±13 импульса счета стандартная величина, ±33 - максимальная
Ошибка смещения	±1 импульс счета (при 0В на входе)
Ошибка линеаризации (при сквозной передаче данных)	макс. ±1 импульс счета
Наибольшая погрешность	±0.02% при 25°C

* Тип термопары определяется установкой внутренних переключателей



F4-08THM-n 8-канальный модуль входов термопар

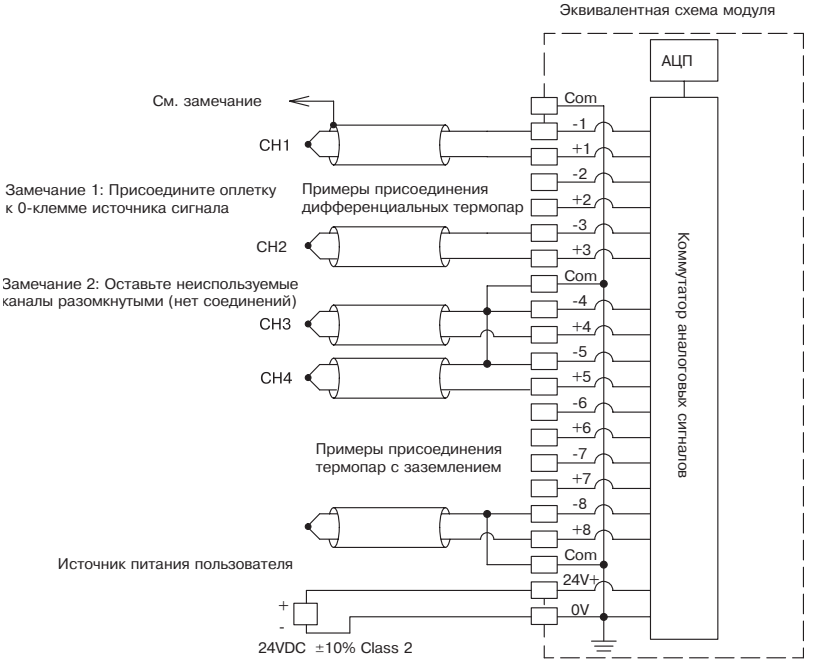
При заказе модуля на место буквы «n» подставьте индекс, соответствующий типу необходимой Вам термопары. Например, если Вам требуется термопара типа J, то заказывать нужно устройство F4-08-THM-J.

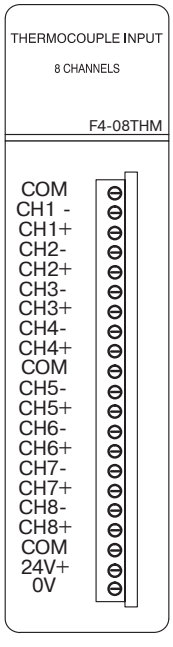
Число каналов	8, дифференциальные
Диапазон сигнала на входах*	Тип B 529 - 1820°C Тип C 65 - 2320°C Тип E -210 - 1000°C Тип J -240 - 760°C Тип K -150 - 1372°C Тип R 0 - 1768°C Тип S 0 - 1768°C Тип T -270 - 400°C -1: 0 - 50 мВ -2: 0 - 100 мВ -3: 0 - 25 мВ
Разрешение	12 бит (1 из 4096)
Входное сопротивление	27КОм для постоянного тока
Максимальный сигнал	Защита входов, 100В
Компенсация холодного спая	Автоматическая
Время преобразования	мин. 15 мсек./канал 1 канал за один цикл сканирования
Способ преобразования	Последовательное приближение

Ошибка линеаризации	макс. ±1 импульс счета (0.03% от истинной величины)
Полная величина ошибки калибровки	±0.35% от истинной величины
Наибольшая погрешность	±1°C для термопар типа E, J, K, и T ±3°C для термопар типа B, C, R, и S
Скорость обновления каналов контроллера	мин. 1, макс. 8 каналов за один цикл сканирования
Занятые цифровые входы	16(X) входные каналы
Информация на входах	(12 двоичных бит данных, 3 канала для битов идентификатора, 1 для знака)
Потребляемый от каркаса ток	макс. 120мА, 5В
Параметры внешнего питания	24В, ±10°C, 50мА
Рабочая температура окружающей среды	0 - 60°C

* Величина характеристики наибольшей погрешности не гарантируется для температур меньших, чем

-220°C для типов E и T
-200°C для типов J и K
+100°C для типов R и S

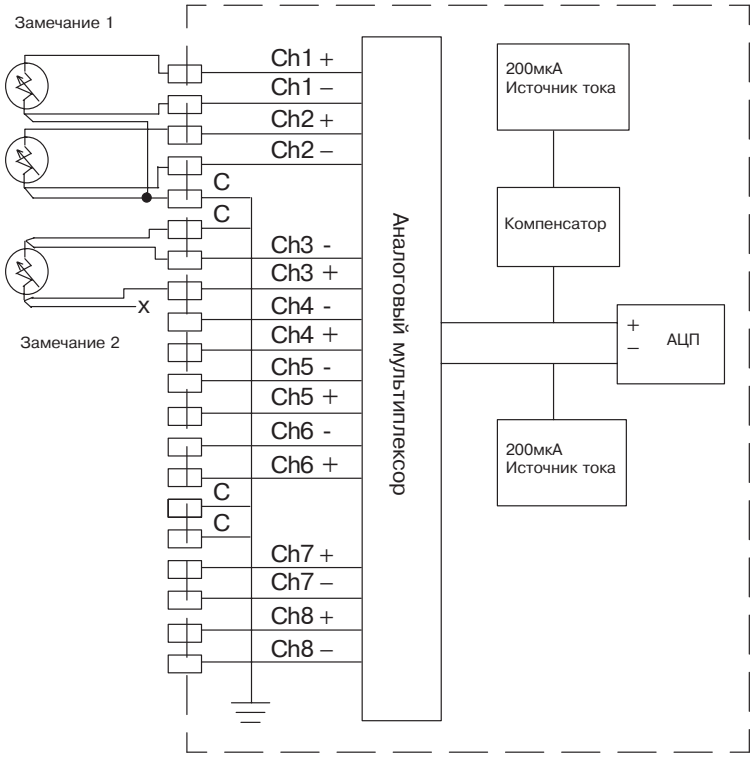


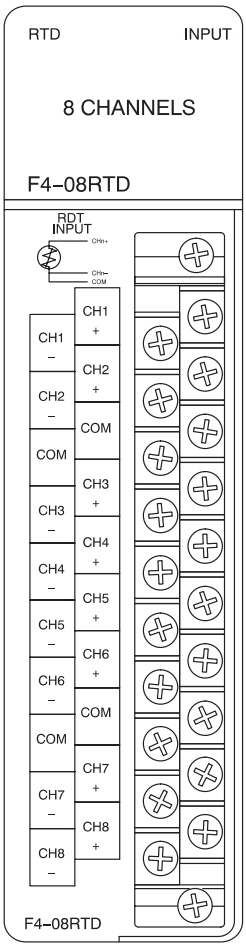


F4-08RTD входы терморезисторов

Число каналов	8
Диапазон сигнала на входах	Тип Pt100: -200.0/850.0 °C Тип Pt1000: -200.0/595.0 °C Тип jPt100: -38.0/450.0 °C Тип CU-10/25Ом: -200.0/260.0 °C
Разрешение	15 бит (1 из 32768)
Сопротивление входов	27КОм при постоянном токе
Разрешение на выходе	±0.1°C (±3276.7)
Ток питания терморезисторов	200мкА
Тип входного сигнала	Дифференциальный
Фильтр	> 100дБ при 50/60 Гц -3дБ = 13.1 Гц
Максимальное время измерения	100мсек (полный диапазон на входе)
Диапазон сигнала в нормальном режиме	0-5В

Максимальный сигнал	Защита входов при ±50В
Способ преобразования сигнала	Компенсационный
Ошибка линеаризации	±0.05°C макс., ±0.01°C стандартная
Полная величина ошибки калибровки	±1 °C
Скорость обновления каналов контроллера сканирования	мин. 1, макс. 4 канала за один цикл
Занятые цифровые входные каналы двоичных	32 (X) входа, 15 бит данных, 3 канала битов идентификации, 1 бит - знак, 8 бит - признак ошибки
Потребляемый от каркаса ток	90мА, 5В
Рабочая температура окружающей среды	0 - 60°C
Уход температуры	Отсутствует (самокалибровка)





Замечания:
1: Три провода, соединяющие терморезисторы с модулем, должны быть одинаковыми и иметь одну и ту же длину.
В качестве третьего провода нельзя использовать экран провода или заземление.
2: Если терморезистор имеет 4 контакта, то дополнительный вывод следует оставить свободным.

Словарь терминов в спецификациях

Входы и выходы на модуль	Указывает число электрических входных и выходных точек на модуль и определяет, является ли он приемником тока, источником тока, либо и тем и другим.
Число общих проводов на модуль	Число общих проводов на модуль. Общим проводом является соединение входного или выходного модуля, которое совместно используется многими цепями ввода/вывода. Обычно им является обратный провод источников питания цепей ввода/вывода.
Диапазон входного напряжения	Диапазон рабочего напряжения входной цепи, измеряемого от входной точки до общего провода при включенном входе.
Диапазон входного напряжения	Диапазон рабочего напряжения выходной цепи, измеряемого от выходной точки до общего провода при выключенном выходе.
Пиковое напряжение	Максимальное напряжение, разрешенное для входной или выходной цепи в течение короткого времени.
Частота переменного тока (AC)	Модули переменного тока разработаны для работы в определенном диапазоне частот.
Порог срабатывания по напряжению (логическая ”1”)	Уровень напряжения, при котором входная точка принимает значение ”1”.
Порог отпускания по напряжению (логический ”0”)	Уровень напряжения, при котором входная точка принимает значение ”0”.
Входное сопротивление	Электрическое сопротивление, измеренное между входной точкой и ее общим проводом. Поскольку это сопротивление не линейное, то оно должно задаваться для различных входных токов.
Входной ток	Рабочий ток, типичный для активного (ВКЛЮЧЕННОГО) входа.
Минимальный ток включения	Минимальный ток во входной цепи для надежной работы в состоянии ”ВКЛЮЧЕНО”.
Максимальный ток выключения	Максимальный ток во входной цепи для надежной работы в состоянии ”ВЫКЛЮЧЕНО”.
Минимальная нагрузка	Минимальный ток нагрузки в выходной цепи, необходимый для надлежащей работы выходных устройств.
Требуемый внешний источник постоянного тока	Для некоторых выходных модулей необходим внешний источник питания для выходных цепей.
Падение напряжения при включении	Иногда называемое ”напряжением насыщения”, это — напряжение, измеряемое между выходной точкой и ее общим проводом, когда выход включен на максимальную нагрузку.
Максимальный ток утечки	Максимальный ток, который подсоединенная максимальная нагрузка будет получать при выключенной выходной точке.

Максимальный бросок тока	Максимальный ток, используемый нагрузкой в течение короткого времени при переходе выходной точки от состояния ”ВЫКЛЮЧЕНО” в состояние ”ВКЛЮЧЕНО”. Он больше тока в нормальном состоянии ”ВКЛЮЧЕНО” и характеризуется индуктивными нагрузками в цепях переменного тока.
Потребление от источника 5В	Питание +5 В постоянного тока от блока питания каркаса, необходимое для работы модуля. Убедитесь, что соблюдается расчет потребляемой каркасом мощности.
Задержка при включении	Время, которое необходимо модулю, чтобы выполнить переход из состояния ”ВЫКЛЮЧЕНО” к состоянию ”ВКЛЮЧЕНО”.
Задержка при выключении	Время, которое необходимо модулю, чтобы выполнить переход из состояния ”ВКЛЮЧЕНО” к состоянию ”ВЫКЛЮЧЕНО”.
Индикация	Светодиоды (LED), которые указывают состояние ”ВКЛЮЧЕНО”/ ”ВЫКЛЮЧЕНО” выходной точки. Эти светодиоды электрически расположены на логической стороне (процессорной) цепей ввода/вывода.
Тип клеммника	Указывает, является ли клеммник съемным или несъемным.
Вес	Указывает вес модуля. См. Приложение E с перечнем весов для различных компонентов DL405.
Предохранители	Защитное устройство в выходных цепях, которое прерывает ток, когда его значение превышает номинал предохранителя. Они могут быть заменяемыми или незаменяемыми, с внешним или внутренним расположением.

Спецификации и работа процессора

3

В этой главе...

— Обзор

— Общие спецификации процессора

— Электрические спецификации процессоров

— Технические характеристики аппаратных средств процессора

— Использование резервного батарейного питания

— Выбор среды для хранения программ

— Настройка процессора

— Работа процессора

— Время отклика ввода/вывода

— Анализ времени сканирования процессора

— Системы счисления в ПЛК

— Карта распределения памяти

— Карта битового отображения Входа X/Выхода Y

— Карта битового отображения Управляющих Реле

— Карты битового отображения Состояний Таймера и Счетчика

— Карты битового отображения удаленного ввода/вывода

— Карта битового отображения Управления/Состояния стадий

Обзор

Общие технические характеристик и процессора

Процессор является сердцем системы управления. Почти все системные операции контролируются процессором, поэтому важно, чтобы он был правильно установлен и отлажен. В данной главе приводится информация, необходимая для понимания:

- различий между разными моделями процессора;
- требуемых шагов по установке и наладке процессора.

DL430, DL440 и DL450 являются модульными процессорами, которые могут устанавливаться в каркасы на 4, 6 или 8 слотов. Все модули ввода/вывода в семействе DL405 могут работать с любым процессором. Все процессоры DL405 предлагают широкий диапазон вычислительных мощностей и программных команд. Все они предлагают команды на языке Релейной Логики (RLL) и команды Стадийного программирования. (См. Главу 5). Они также обеспечивают обширную внутреннюю диагностику, которая может контролироваться из прикладной программы или с помощью интерфейса оператора.

Эти три стандартные типы процессоров работают от источников питания 110 или 220 В переменного тока. DL440/DL450 включают дополнительные версии процессора с питанием от постоянного тока: DL440DC-1 с питанием 24 В и DL440/DL450 DC-2 с питанием 125 В.

Технические характеристик и процессора DL430

DL430 имеет 6.5K программной памяти, включающей 3.5K программной памяти пользователя и 3.0K V-памяти (регистры данных). Он имеет 113 различных команд, доступных для программирования, и поддерживает максимум 640 точек ввода/вывода в локальном каркасе и в блоке расширения, а также 512 точек удаленного ввода/вывода.

Программы хранятся в Элетронно — Перепрограммируемом ПЗУ (ЭППЗУ), которое встроено в процессор. Кроме ЭППЗУ -памяти в процессоре имеется также Оперативная Память (RAM), в которой могут храниться параметры системы, V-память и другие данные, не относящиеся к прикладной программе.

DL430 имеет два встроенных коммуникационных порта. Первый из них имеет интерфейс RS232C, второй — RS232C/RS422. Это позволяет иметь для первого порта двухточечные (прямые) соединения, а для второго порта — варианты либо многоточечных сетевых соединений (например, DirectNET), либо прямых соединений.

Технические характеристик и процессора DL440

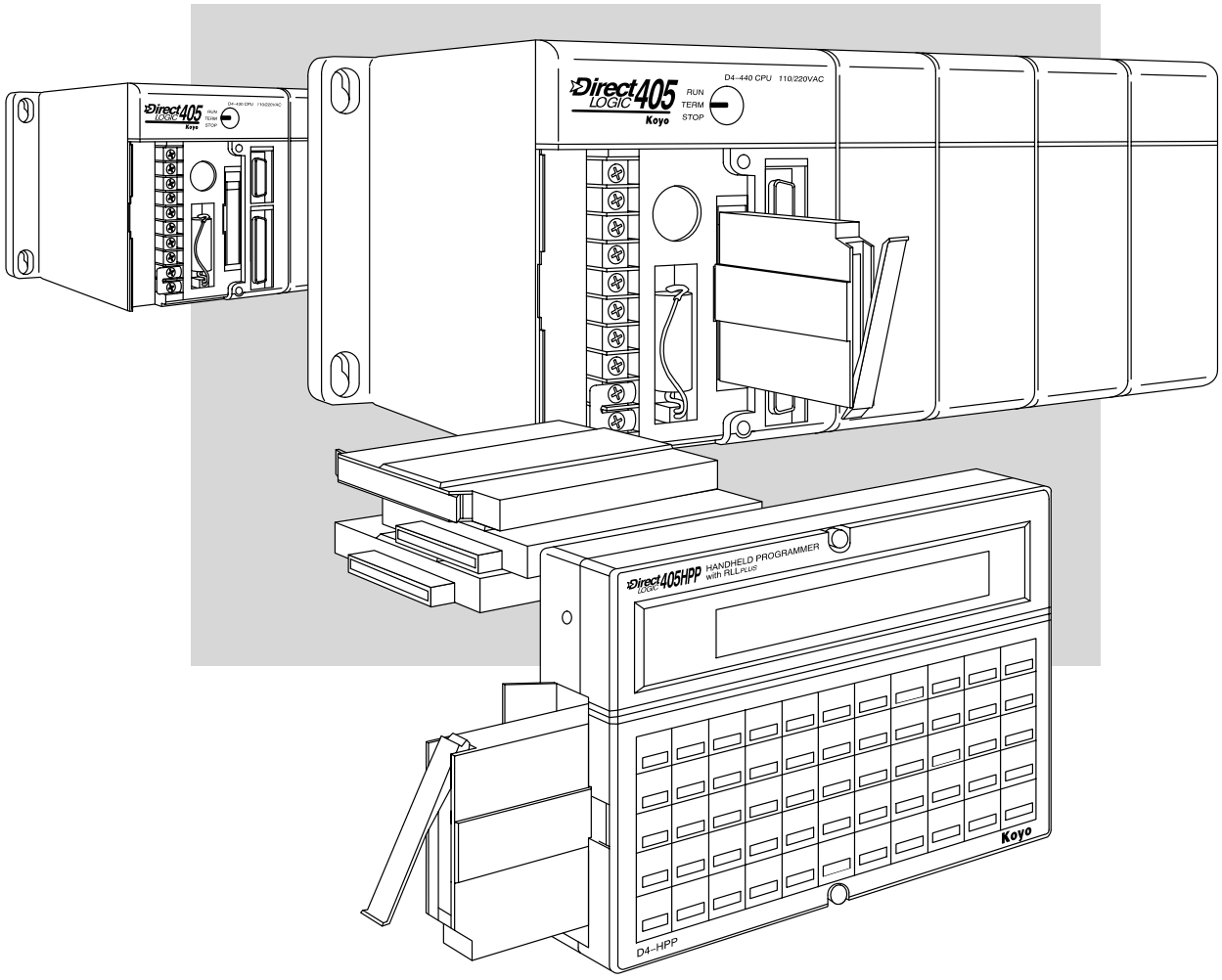
DL440 включает все технические характеристики DL430 плюс большее число точек ввода/вывода, большее число команд для программирования и больший объем памяти в съемном картридже памяти. Процессор имеет максимум 22.5K программной памяти, включающей 3.5K программной памяти пользователя и 3.0K V-памяти (регистры данных). Он поддерживает максимум 640 точек ввода/вывода в локальном каркасе и в блоке расширения, а также 1024 точек удаленного ввода/вывода. Процессор имеет два коммуникационных порта, функционирующих также как и порты DL430.

DL440 имеет 170 команд. Дополнительные 57 команд (по сравнению с набором команд DL430) позволяют разрабатывать более сложные программы с использованием подпрограмм и команд, поддерживающих работу с двойными словами, расширенные операции со стеками, обработку диагностических сообщений и ASCII/шестнадцатеричное форматирование данных.

Технические характеристик и процессора DL450

Новый DL450 предлагает все функции DL440 плюс большее число точек ввода/вывода, большее число команд и два дополнительных коммуникационных порта (всего 4). Процессор имеет максимум 30.8K программной памяти, включающей 15.5K памяти пользовательских программ и 15.3K V-памяти (регистры данных). Он поддерживает максимум 2048 точек ввода/вывода в локальном каркасе и в блоке расширения, а также 1536 точек удаленного ввода/вывода. Он имеет дополнительный RISC-микропроцессор для повышения вычислительной мощности. DL450 имеет 210 команд. Дополнительные 40 команд (по сравнению с набором команд DL440) включают таймеры командоаппарата, функцию печати, математику с плавающей точкой, тригонометрические функции и управление 16 контурами ПИД-регулирования.

DL450 имеет всего четыре коммуникационных порта. Первые два порта идентичны соответствующим портам DL430 и DL440. Третий порт имеет интерфейс RS232C и может быть сконфигурирован либо на протокол N-sequence, либо на протокол K-sequence. В нем используется модульный разъем для прямого подключения к таким устройствам, как панель оператора к Данным DV-1000. Четвертый порт имеет интерфейс RS-485, использующий протокол MODBUS (ведущее/ведомое устройство), протоколы N-sequence или K-sequence ведущее устройство Remote I/O. Эти четыре порта используют три разъема (нижний разъем в DL450 рассчитан на два порта).



Общие спецификации процессора

Техническая характеристика	DL430	DL440	DL450
Общая программная память (слов)	6.5K	14.5K / 22.5K*	22.8K / 30.8K*
Память для пользовательских программ (слов)	3.5K	7.5K / 15.5K*	7.5K / 15.5K*
V-память (слов)	3.0K	7.0K	15.3K
Выполнение 1K булевых операций	8 - 10 мс	2 - 3 мс	4 - 5 мс
Редактирование в рабочем режиме	Нет	Да	Да
Программирование в RLL и RLL^{plus}	Да	Да	Да
Ручной Программатор с интерфейсом на магнитном носителе	Да	Да	Да
Программирование на DirectSOFT для WindowsTM	Да	Да	Да
Встроенные коммуникационные порты	2 порта	2 порта	4 порта
Оперативная память на базе КМОП- технологии (CMOS RAM)	Нет	С картриджем памяти	С картриджем памяти
Программируемое ПЗУ с ультрафиолетовым стиранием информации (UVPROM)	Нет	С картриджем памяти	С картриджем памяти
ЭППЗУ	Стандартная для процессора	С картриджем памяти	С картриджем памяти
Флэш- память	Нет	Нет	Стандартная для процессора
Совместимость с:			
Модулями CoProcessor^{TM}	Да	Да	Да
Сетевыми модулями	Да	Да	Да
Модулем передачи данных RS232C/RS422	Да	Да	Да
Общее число точек ввода/вывода	1152	1664	3584
Общее число точек ввода/вывода, доступных как:			
Локальный ввод/вывод/Расширение локального ввода/вывода	640	640	2048
Удаленный ввод/вывод	512 max.	1024 max.	1536 max.
Каналы удаленного ввода/вывода	2	2	3
Локальные дискретные точки ввода, максимум	320	320	1024
Локальные дискретные точки вывода, максимум	320	320	1024
Каналы локального аналогового ввода, максимум	80 **	80 **	80 **
Каналы локального аналогового вывода, максимум	40**	40**	40**
Максимальное число каналов / ведущих устройств (удаленных или секционированных) на локальный каркас процессора	2	2	2
Расстояние удаленного ввода/вывода	3300 ft. (1000m)	3300 ft. (1000m)	3300 ft. (1000m)
Число дискретных точек ввода/вывода в модуле	8/16/32/64	8/16/32/64	8/16/32/64
Число слотов в каркасе	4/6/8	4/6/8	4/6/8

* Первое значение относится к процессорам, использующих картриджи памяти 7.5K, второе — к процессорам, использующим картриджи памяти 15.5K.

** При использовании удаленного ввода/вывода могут поддерживаться дополнительные дискретные и аналоговые точки ввода/вывода (при наличии резерва потребляемой мощности).

Техническая характеристика	DL430	DL440	DL450
Число доступных команд (см. главу 5 с описанием команд)	113	170	210
Управляющие реле	480	1024	2048
Специальные реле (определяемые в системе)	288	352	512
Стадии в RLLplus	384	1024	1024
V-память	3072 слов	7168 слов	15360 слов
Таймеры	128	256	256
Счетчики	128	128	256
Прямой ввод/вывод	Да	Да	Да
Ввод с прерыванием	8 точек	16 точек	16 точек
Подпрограммы	Нет	Да	Да
Циклы FOR/NEXT	Нет	Да	Да
Таймеры барабанных командоаппаратов	Нет	Нет	Да
Математика	Целая	Целая	Целая и с плавающей точкой
Управление контурами ПИД-регулятора, встроенное	Нет	Нет	16 контуров
Часы истинного времени/календарь	Нет	Да	Да
Внутрисистемная диагностика	Да	Да	Да
Защита паролем	Нет	Да	Да, многоуровневая
Регистрация системных и пользовательских ошибок	Нет	Да	Да
Резервное батарейное питание	Да	Да	Да

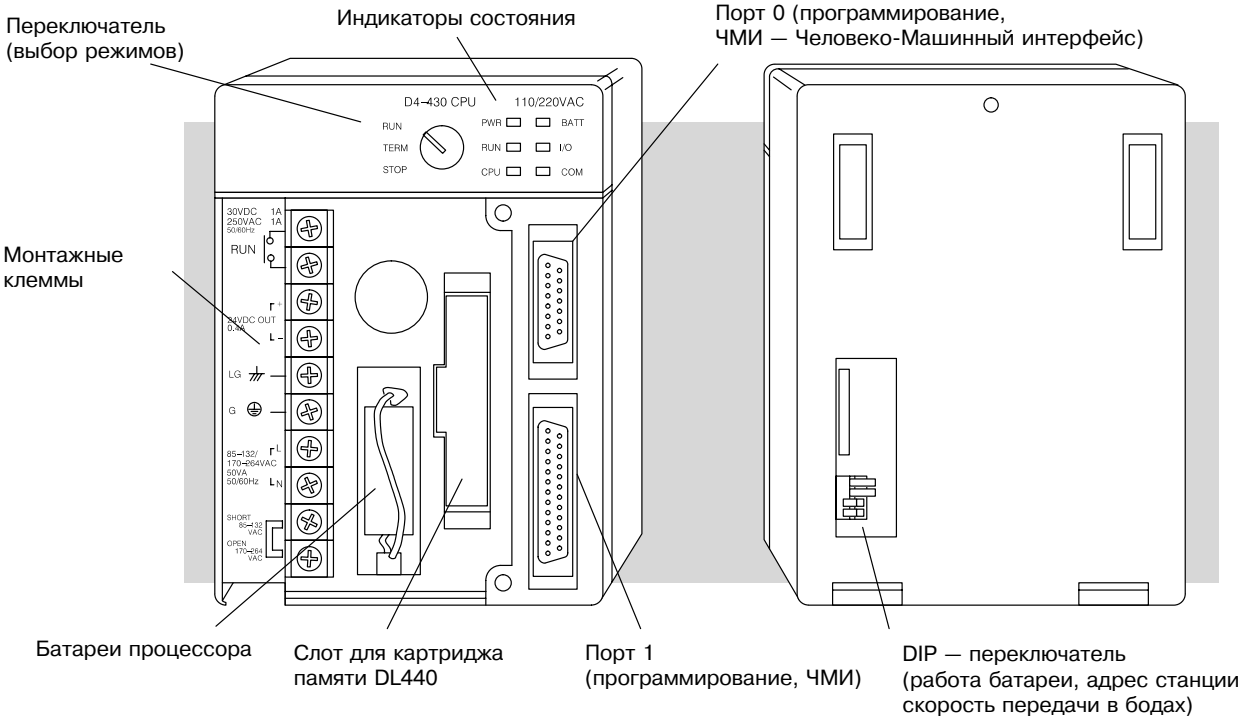
Электрические спецификации процессоров

Параметр	DL430/DL440/DL450	DL440/DL450 DC-1	DL440/Dl450 DC-2
Входное напряжение, номинал	120 В переменного тока	24 В постоянного тока	125 В постоянного тока
Диапазон входного напряжения	85 - 132В переменного тока	20 - 29 В постоянного тока	90 - 146 В постоянного тока
Пульсация входного напряжения	Нет данных	менее 10%	менее 10%
Максимальный пусковой ток	20 A	10 A	20 A
Потребляемая мощность, максимум	50 V A	38W	36 W
Стойкость к напряжению (электрическая прочность диэлектрика)	1 минута при1500 В переменного тока между первичным, вторичным, эксплуатационным заземлением и пусковым реле		
Сопротивление изоляции	> 10 MΩ при 500 В постоянного тока		
Выходное напряжение, вспомогательный источник питания	20-28 В постоянного тока (номинал 24 В), пульсация менее 1 В р-р (Нет у DL440/DL450 DC-1 и DL440/DL450 DC-2)		
Выходной ток, вспомогательный источник питания	400 мА максимум		

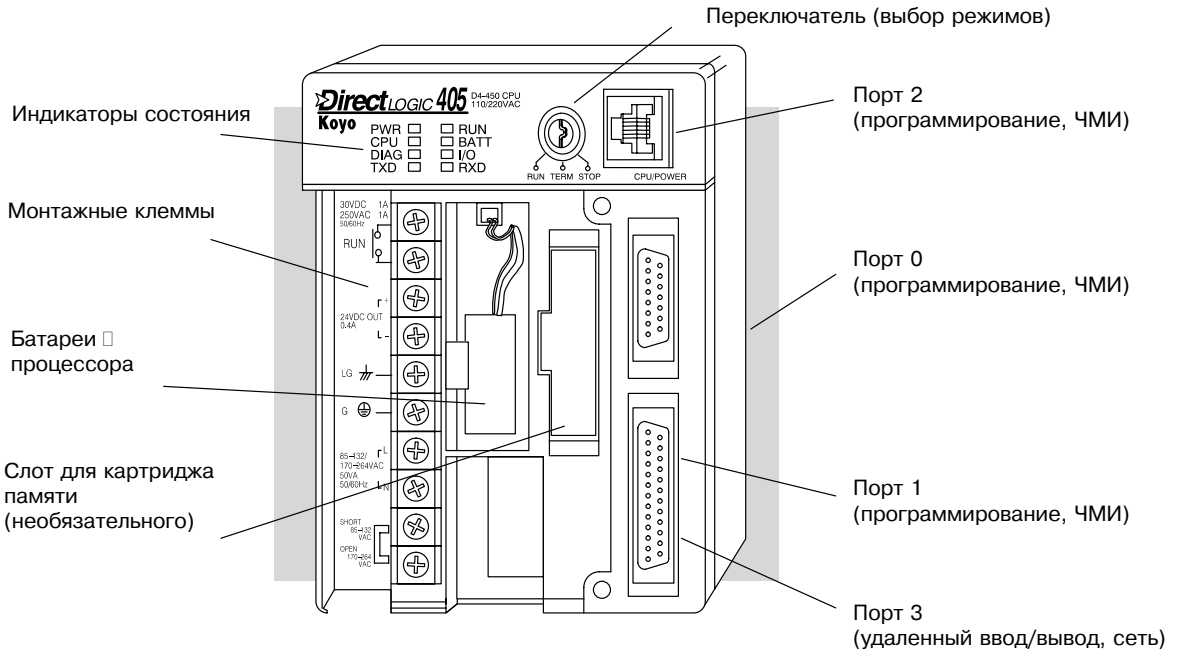
Технические характеристики аппаратных средств процессора

Процессоры DL430/DL440

На следующих рисунках показаны внешние аппаратные средства процессоров DL405.



Процессор DL450



Функции переключателя режимов

Переключатель режимов в процессорах DL405 имеет положения для разрешения и блокировки изменений программ в процессоре. Если переключатель режимов не находится в положении TERM, то изменения режима RUN и STOP не разрешаются никакому интерфейсному устройству (ручному программатору, программному пакету DirectSOFT или интерфейсу оператора). Программы могут просматриваться или контролироваться, но никакие изменения не могут вноситься. Если переключатель находится в положении TERM и программы не защищены паролем, то доступны все рабочие режимы, также как программы через подсоединенное устройство для программирования или контроля.

Положение переключателя режимов	Действие процессора
RUN (Запуск программы)	Процессор переводится в режим RUN, если не возникает никаких ошибок. Никакие изменения с помощью подсоединенных устройств программирования/ контроля не разрешены.
TERM (Терминал)	Допустимы режимы RUN, PROGRAM и TEST. Изменения режимов и программ с помощью подсоединенных устройств программирования/ контроля разрешены.
STOP (Для DL250 только остановка программы)	Процессор переводится в режим STOP. Никакие изменения с помощью подсоединенных устройств программирования/контроля не разрешены.

Существует два способа изменения режима процессора

1. Использовать переключатель режимов процессора для выбора рабочего режима.
2. Установить переключатель режимов процессора в положение TERM и использовать устройство программирования для изменения рабочих режимов. В этом положении переключателя вы можете менять режимы между Рабочим (RUN) и Программным (PROGRAM).

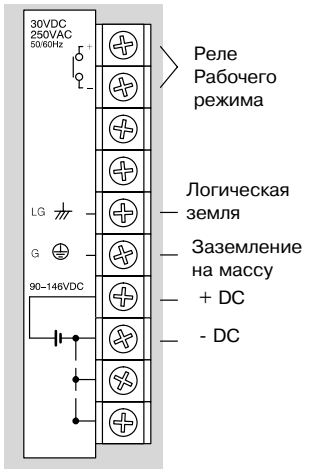
Индикаторы состояния

Светодиоды (LED) индикаторов состояния на передних панелях процессора имеют специфичные функции, которые могут помочь при программировании и при поиске неисправностей.

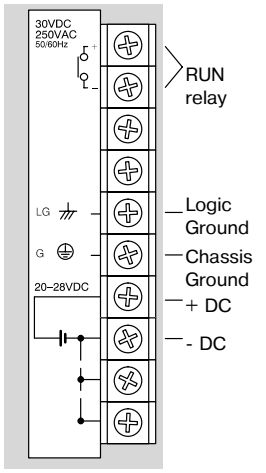
Индикатор	Состояние	Значение
PWR	ON	Питание исправно
	OFF	Отсутствие питания
RUN	ON	Процессор находится в Рабочем режиме
	OFF	Процессор находится в режиме Stop или в программном режиме
CPU	ON	Процессор диагностировал ошибку
	OFF	Процессор диагностировал исправную работу
BATT	ON	Напряжение батареи процессора низкое
	OFF	Напряжение батареи достаточное или индикация отключена
DIAG (DL450)	ON	Процессор диагностировал ошибку или ошибка в локальной шине
	OFF	Ошибки при диагностировании и в общей шине отсутствуют
I/O	ON	Диагностирована ошибка ввода/вывода
	OFF	Ошибки при диагностировании ввода/вывода отсутствуют
COM (DL430/DL440)	ON	Обнаружена ошибка в коммуникациях
	OFF	Ошибки в коммуникациях отсутствуют
TXD (DL450)	ON	Данные передаются процессором
	OFF	Данные не передаются процессором
RXD (DL450)	ON	Данные принимаются процессором
	OFF	Данные не принимаются процессором

Монтажные клеммы

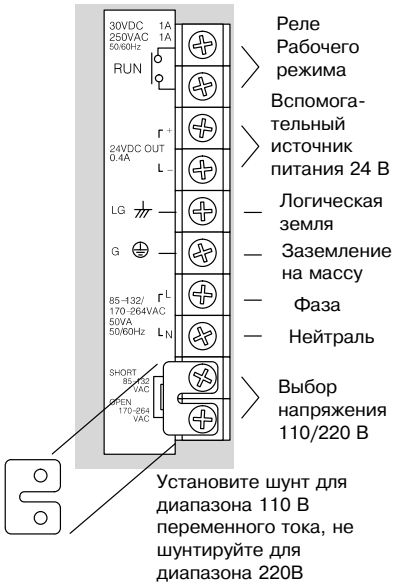
Клеммник, питание 125 В постоянного тока



Клеммник, питание 24 В постоянного тока



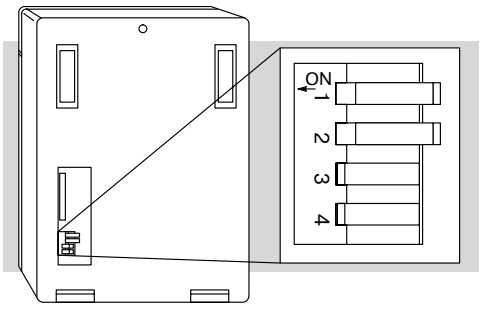
Клеммник, питание 110/220 В переменного тока



Установка DIP-переключателей процессора

√	√	X
430	440	450

На задней панели процессоров DL430 и DL440 расположен блок из четырех переключателей конфигурации. С помощью этих переключателей можно устанавливать определение низкого напряжения на батарее, заменять адрес станции, устанавливать скорость передачи в бодах порта 1 (с 25-контактным разъемом типа D). Чтобы задать такой выбор для процессора DL450 используйте вспомогательные AUX — функции на устройстве программирования.



Переключатель 1

- ON = Индикатор низкого напряжения на батарее заблокирован
- OFF = Индикатор низкого напряжения на батарее включен

Переключатель 2

- ON = Ручная корректировка адреса станции разрешена (адрес 1)
- OFF = Адрес станции устанавливается с помощью AUX — функции с устройства программирования.

ПРИМЕЧАНИЕ. Установка переключателя 2 в положение ON переводит адрес станции в значение ”1”. При этом адрес, установленный с помощью устройства программирования, не изменяется. Если снова перевести переключатель 2 в положение OFF, то адрес, установленный с помощью AUX-функции и хранимый в памяти, будет возвращен.

Скорость передачи в бодах порта 1	Переключатель 3	Переключатель 4
300	Off	Off
1200	Off	On
9600	On	Off
19200	On	On

ПРИМЕЧАНИЕ. Четность, режим и адрес станции для порта 1 устанавливаются с помощью AUX — функции с устройства программирования.

Порты коммуникаций

Процессоры DL405 имеют до четырех коммуникационных портов. Процессоры DL430 и DL440 имеют по два порта, а процессор DL450 имеет всего четыре порта.

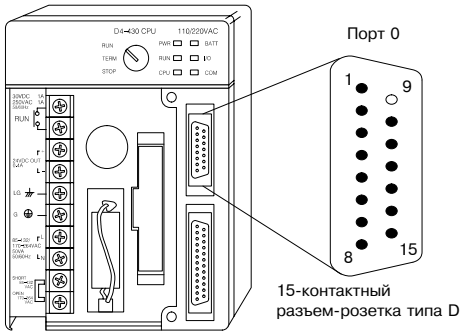
Порт 0, Спецификации

√	√	√
430	440	450

Первый порт (для всех процессоров) имеет 15 — контактный разъем типа D. Он предназначен для подсоединения основных устройств для программирования, например, DirectSOFT или интерфейса оператора. Ручной программатор D4-HPP может подсоединяться к процессору только

Через этот порт. Установлены следующие рабочие параметры Порта 1:

- 15 — контактный разъем-розетка типа D
- протокол: K-sequence
- RS232C, неизолированный, расстояние 15 м
- 9600 бод, 8-битовые данные, 1 стартовый, один стоповый бит, проверка на нечетность ODD
- Асинхронный, полудуплексный, оконечное оборудование данных (DTE)



Порт 0 Описание контактов (все процессоры)		
1	YOP	Сигнал соединения HPP с процессором
2	TXD	Передача данных (RS232C)
3	RXD	Прием данных (RS232C)
4	ONLINE	Запрос связи (TTL)
5	ABNO	Ошибка процессора (TTL)
6	PRDY	Готовность процессора к связи (TTL)
7	CTS	Готовность к пересылке (RS232C)
8	YOM	Сигнал соединения HPP с процессором
9	-	Не используется
10	LCBL	Сигнал подключения кабеля (TTL)
11	5V2	Питание 5 В постоянного тока схем HPP
12	5V2	Питание 5 В постоянного тока подсветки жидкокристаллических индикаторов HPP
13	0V	Логическое заземление
14	0V	Логическое заземление
15	0V	Логическое заземление

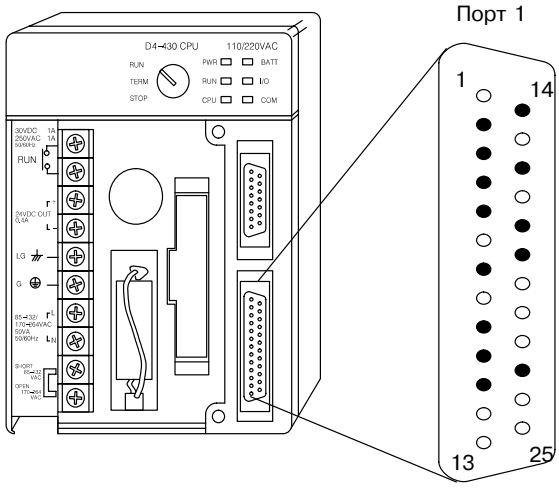
Порт 1, Спецификации

√	√	√
430	440	450

Порт 1 (во всех процессорах) имеет 25 - контактный разъем, в процессорах DL430/DL440 называется ”вторичный последовательный (comm.) порт”. Адрес вторичного последовательного порта хранится на картридже памяти вместе с конфигурацией ввода/вывода. Он предназначен для подсоединения основных устройств для программирования, например, DirectSOFT, интерфейса оператора или сети, но к нему нельзя подсоединить к ручному программатору. Порт 1 имеет дополнительные возможности, такие как программируемая скорость передачи, контроль по четности, ASCII/шестнадцатеричный режим и сетевой адрес. Его сигналы RS422 поддерживают многоточечные сетевые и программные приложения.

Корректировка скорости передачи и адреса станции производится с помощью DIP - переключателей, расположенных на задней панели процессоров DL430/DL440. Для установки указанных параметров в DL450 должны использоваться AUX-функции (он не имеет DIP - переключателей). Выбор между RS232C и RS422 задается при подключении кабеля к соответствующим контактам сигналов разъема. Контроль по четности, ASCII/шестнадцатеричный режим и адрес станции задаются с помощью вспомогательных AUX - функций с устройств для программирования.

- 25 - контактный разъем - розетка типа D.
- Протоколы: K-sequence, DirectNET. DL450 дополнительно поддерживает протоколы Non-sequence и MODBUS.
- RS232C/RS422, выбираемые адреса 1 - 90 (с использованием AUX-функции).
- 300/600/1200/2400/4800/9600/19200 бод (38400 только для DL450).
- Режимы Шестнадцатеричный / ASCII (с использованием AUX - функции для конфигурирования).
- 8-битовые данные, 1 стартовый бит, один стоповый бит, проверка на нечетность / отсутствие проверки.
- Асинхронный, полудуплексный (с использованием AUX - функции для конфигурирования), оконечное оборудование данных (DTE)



25-контактный
разъем-розетка типа D

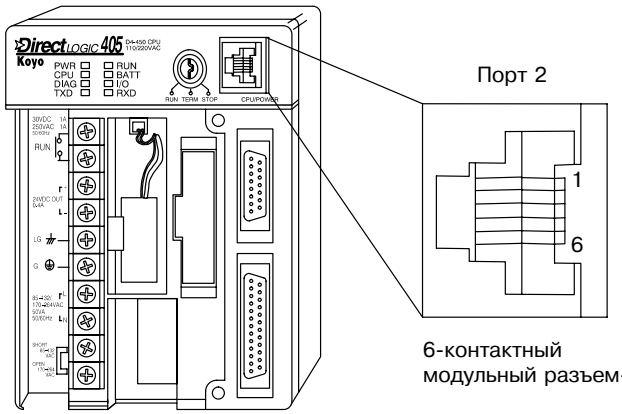
Порт 1. Описание контактов (все процессоры)		
1	-	Не используется
2	TXD	Передача данных (RS232C)
3	RXD	Прием данных (RS232C)
4	RTS	Запрос на пересылку (RS232C)
5	CTS	Готовность к передаче (RS232C)
6	-	Не используется
7	SG	Заземление сигнала (RS232C/RS422)
8	-	(Порт 3 в DL450)
9	RXD+	Получение данных "+" (RS422)
10	RXD-	Получение данных "-" (RS422)
11	CTS+	Готовность к передаче "+" (RS422C
12	-	(Порт 3 в DL450)
13	-	(Порт 3 в DL450)
14	TXD+	Передача данных "+" (RS422)
15	-	Не используется
16	TXD-	Передача данных "-" (RS422)
17	-	Не используется
18	RTS-	Запрос на пересылку "-" (RS422)
19	RTS+	Запрос на пересылку "+" (RS422)
20	-	Не используется
21	-	Не используется
22	-	Не используется
23	CTS-	Готовность к пересылке "-" (RS422)
24	-	(Порт 3 в DL450)
25	-	(Порт 3 в DL450)

Порт 2, Спецификации

X	X	√
430	440	450

Рабочие параметры порта 2 в процессоре DL450 конфигурируются с использованием вспомогательных функций с программирующего устройства.

- Разъем типа 6-контактной модульной розетки (телефонное гнездо RJ12)
- Протоколы: DirectNET (только как ведомое устройство), K-sequence, Непроцедурный.
- RS232C, 300/600/1200/2400/4800/9600/19200/38400 бод.
- 8-битовые данные, 1 стартовый бит, один стоповый бит; проверка на нечетность, проверка на четность, отсутствие проверки.
- Адреса узлов от 1 до 90.



6-контактный
модульный разъем-розетка

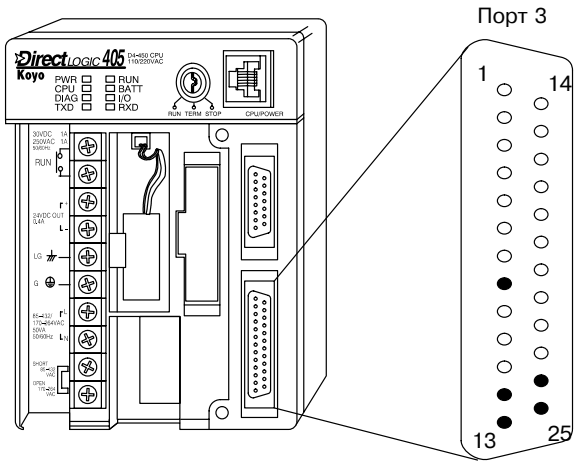
Порт 2 Описание контактов (только DL450)		
1	0V	Соединение с питанием (-) ("Земля")
2	5V	Соединение с питанием (+)
3	RXD	Прием данных (RS232C)
4	TXD	Передача данные (RS232C)
5	5V	Соединение с питанием (+)
6	0V	Соединение с питанием (-) ("Земля")

Порт 3, Спецификации

X	X	√
430	440	450

Рабочие параметры порта 3 в процессоре DL450 конфигурируются с использованием вспомогательных функций с программирующего устройства.

- 25 - контактный разъем - розетка типа D.
- Протоколы: DirectNET, K-sequence, Удаленный ввод/вывод, ведущий или ведомый MODBUS.
- RS422C, неизолированный, расстояние 1000 м.
- 300/600/1200/2400/4800/9600/19200/38400 бод (DirectNET,. K-sequence, MODBUS).
- 8-битовые данные, 1 стартовый бит, один стоповый бит; проверка на нечетность, проверка на четность, отсутствие проверки.
- Режимы Шестнадцатеричный / ASCII (с использованием AUX - функции для конфигурирования).
- Выбираемые адреса от 1 до 90 (с использованием AUX - функции для конфигурирования).

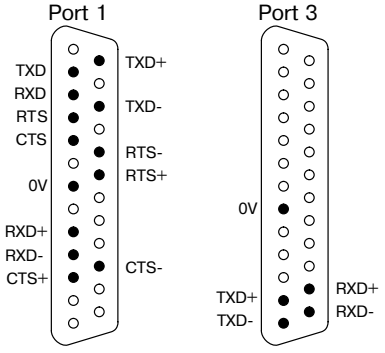


25-контактный
разъем-розетка типа D

Порт 3. Описание контактов (только DL450)		
1	-	Не используется
2		Порт 1
3		Порт 1
4		Порт 1
5		Порт 1
6	-	Не используется
7		Заземление сигнала
8	SG	Не используется
9		Порт 1
10		Порт 1
11		Порт 1
12	TXD+	Передача данных ”+” (RS422)
13	TXD-	Передача данных ”-” (RS422)
14		Порт 1
15	-	Не используется
16		Порт 1
17	-	Не используется
18		Порт 1
19		Порт 1
20	-	Не используется
21	-	Не используется
22	-	Не используется
23		Порт 1
24	RXD+	Получение данных ”+” (RS422)
25	RXD-	Получение данных ”-” (RS422)

Справа показано итоговое размещение контактов на 25 - контактном разъеме и функции портов 1 и 3. Два логических порта совместно используют контакты заземления, но имеют отдельные контакты по обмену данными. Вам, вероятно, придется заказать специальный разъем, который разделит сигналы для двух отдельных кабелей.

Два логических канала на одном 25-контактном разъеме



Выбор среды для хранения программ

Процессоры DL405 снабжены литиевой батареей, которая позволяет поддерживать в сохранности оперативную память (ОЗУ) системы, когда система остается без внешнего питания. Типовой срок службы батареи процессора — 5 лет. Он включает рабочее время ПЛК и обычные периоды отключения системы. Однако целесообразно установить новую батарею, если ваша батарея не заменялась в последнее время, а система отключалась на период более десяти дней.

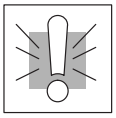
Индикаторы батарей будут мерцать, когда батарея требует замены.

- Мерцание с частотой 0.5 Гц означает, что батарея процессора требует замены.
- Мерцание с частотой 0.25 Гц означает, что батарея картриджа ОЗУ требует замены.

ПРИМЕЧАНИЕ. Перед установкой или заменой батареи процессора скопируйте V-память и системные параметры. Вы можете сохранить V-память и системные параметры либо на картридже памяти, на магнитной ленте или на персональном компьютере (с использованием DirectSOFT). Батарею процессора можно менять при включенном питании системы, чтобы предотвратить потерю памяти. Если же питание процессора было выключено, то нужно включить его минимум за 5 секунд до замены батареи. Это гарантирует, что емкость, используемая для поддержания необходимого для сохранения памяти уровня напряжения, будет полностью заряжена.

Для удаления отслужившей батареи процессора необходимо:

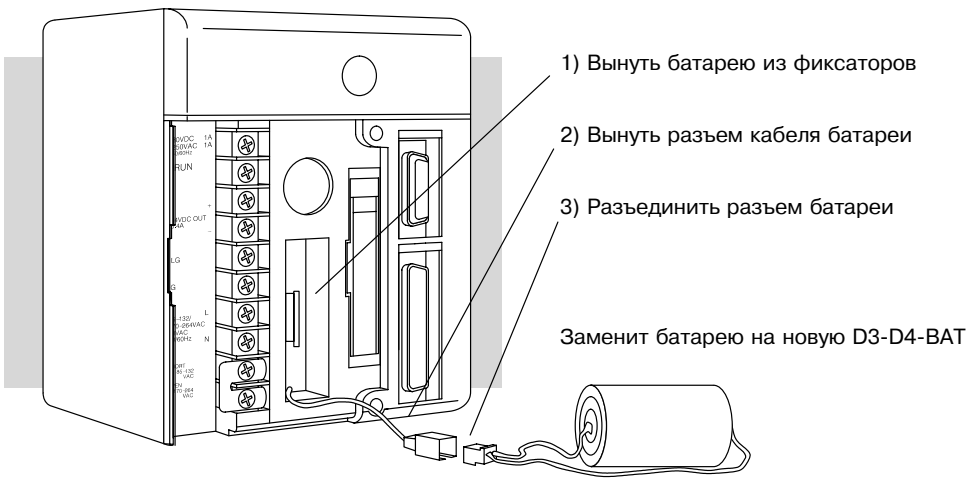
1. вынуть батарею из фиксаторов батареи,
2. освободить зажим двухпроводного разъема,
3. отсоединить разъем батареи.



ПРЕДУПРЕЖДЕНИЕ. Не пытайтесь перезарядить батарею или положить старую батарею вблизи огня. Батарея может взорваться или выделять опасные материалы.

Для установки батареи необходимо:

1. соединить разъем батареи (маркированный) так, чтобы совпали красные провода;
2. мягко вставить разъем батареи до щелчка зажима;
3. вставить батарею в фиксаторы (заподлицо с углублением под батарею);
4. отметить дату замены батареи.

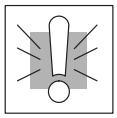


Выбор среды для хранения программ

В процессоре DL430 предусматривается встроенная память ЭППЗУ (Электронно — перепрограммируемое ПЗУ) как стандартное средство, поэтому он не может использовать другие способы хранения программ. Процессор DL440 требует для работы съемного картриджа памяти. В процессоре DL450 предусматривается встроенная ФЛЭШ - память как стандартное средство запоминания Он может иметь еще картридж памяти для хранения программ. Однако установка картриджа памяти блокирует внутреннюю ФЛЭШ - память. Следующие разделы помогут вам правильно выбрать запоминающее устройство для вашего типа процессора и приложения.

Встроенная память ЭППЗУ

Этот тип памяти относится к долговременной, он не зависит от батарейного питания для сохранения программ. ЭППЗУ может электрически перепрограммироваться без изъятия из Процессора. Вы можете установить Перемычку 3 (Jumper 3), которая защитит ЭППЗУ от записи. Перемычка устанавливается в заводских условиях, чтобы разрешить внесение изменений в ЭППЗУ. Если вы выбираете защиту от записи, изменяя положение перемычки, то вы не сможете делать изменения в программе.



ПРЕДУПРЕЖДЕНИЕ. НЕ изменяйте Перемычку 2. Она предназначена для операций при заводских испытаниях. Если вы измените Перемычку 2, то Процессор не будет правильно работать.

Энергозависимая и энергонезависимая память

Существует два типа памяти: энергозависимая и энергонезависимая. Энергозависимая память сохраняет данные только тогда, когда в запоминающей среде поддерживается необходимое напряжение. Энергонезависимая память не требует подачи питания для сохранения данных. Процессоры DL450 поддерживают необходимое напряжение либо от источника питания, либо при использовании батареи резервного питания.

Типы запоминающих устройств

Картриджи памяти применимы только в процессорах DL440 и DL450. Картридж памяти необходим для DL440, но он не обязателен для DL450. Съемный картридж памяти доступен либо как КМОП ОЗУ, ЭППЗУ или УФПЗУ. В типах устройств ОЗУ и ЭППЗУ внутри картриджа предусмотрена перемычка выбора защиты от записи. При открытом картридже перемычку можно поставить в положение защиты, чтобы предотвратить случайное стирание или изменение программ.

- КМОП ОЗУ (Комплементарный металло -оксидный полупроводник / оперативное запоминающее устройство) требует батареи для сохранения памяти, она размещена внутри картриджа. Память может легко изменяться или заменяться с ручного программатора или с помощью программ персонального компьютера. Срок службы батареи обычно 3 года. По вопросу замены батареи обратитесь к главе 9 ”Обслуживание и поиск неисправностей”.
- УФПЗУ (Программируемое ПЗУ с ультрафиолетовым стиранием информации) не требует батареи для сохранения памяти, поэтому оно относится к ”энергонезависимым” устройствам памяти. Однако стирание (очистка памяти) требует облучения интегральных схем памяти источником ультрафиолетового излучения. Процессор может перепрограммировать картридж УФПЗУ после стирания, но для этого нужен ручной программатор.

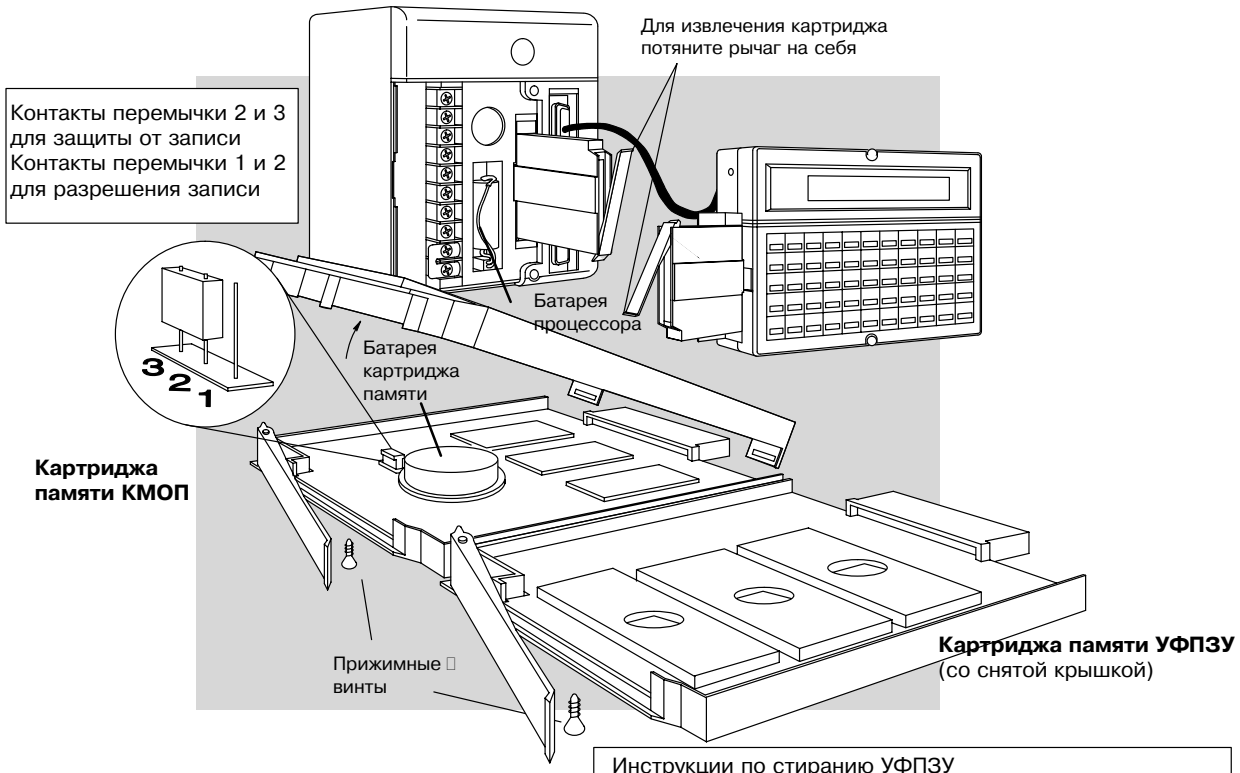
ПРИМЕЧАНИЕ. Если Вы приобрели картридж памяти УФПЗУ, рекомендуем вам также иметь либо ОЗУ, либо ЭППЗУ для разработки программ. После завершения разработки Вы должны далее использовать ручной программатор (D4-HPP), чтобы скопировать вашу прикладную программу в УФПЗУ. Мы рекомендуем вариант с картриджем УФПЗУ для приложений, имеющих массовый характер и не требующих частых изменений.

- ЭППЗУ (Электрически перепрограммируемое ПЗУ) не требует батареи для сохранения памяти, поэтому оно относится к ”энергонезависимым” устройствам памяти. Как стирание программ, так и программирование выполняются электрическим способом, не требующим источника ультрафиолетового излучения. Поэтому картридж памяти ЭППЗУ можно электрически перепрограммировать (если он не защищен от записи) без изъятия его из процессора. Этот тип памяти встроен в процессор DL440.

Картриджи памяти

На рисунке ниже изображен картридж памяти для DL440 или DL450. На нем показано, как картридж вставляется в процессор и в ручной программатор. На нем показано также, как открыть картридж для установки защиты от записи (для КМОП ОЗУ) и для стирания УФПЗУ. Подробная информация по подсоединению ручного программатора к процессору приведена в Руководстве по ручному программатору.

ПРЕДУПРЕЖДЕНИЕ. Не вставляйте и не снимайте картридж памяти при включенном питании. В этом случае можно повредить программы и пароль. Поврежденные программы могут привести к непредсказуемой работе, при которой возникает риск травмирования персонала или повреждения оборудования.



Инструкции по стиранию УФПЗУ
1) Вынуть картридж из процессора или HPP
2) Отвернуть прижимные винты картриджа
3) Снять крышку
4) Поместить картридж под ультрафиолетовую лампу для стирания, (обычно лампа с мощностью ~0,012 Вт/см^2 с расстояния 2,5 см стирает в течение 15µ 20 минут)
5) Верните крышку на место

Таблица мощности картриджей памяти

	D4-RAM-1	D4-RAM-2	D4-UV-1	D4-UV-2	D4-EE-2
Объем программной памяти	7.5K	15.5K	7.5 K	15.5K	15.5K
Тип батареи картриджа	Литиевая	Литиевая	Нет	Нет	Нет
Количество циклов записи			1000	1000	>10,000
Запрещение записи	Внутренняя перемычка	Внутренняя перемычка	Нет	Нет	Внутренняя перемычка
Метод очистки памяти	Электрический	Электрический	Ультрафиолет. излучение	Ультрафиолет. излучение	Электрический

Настройка процессора

Установка часов и календаря

X	√	√
430	440	450

Процессоры DL440 и DL450 имеют также Часы/Календарь, которые можно использовать для многих целей. Если у вас есть необходимость использовать эту функцию, то вы можете воспользоваться AUX функцией, позволяющей установить дату и время. Например, вы пожелаете использовать AUX 52 "Показать / Изменить Календарь" для установки даты и времени с ручного программатора. В DirectSOFT вы можете воспользоваться опциями меню настройки ПЛК. Имеются также две команды, позволяющие изменять или подправлять дату и время из прикладной программы. В главе 5 приводится информация по командам DATE и TIME, используемых для установки часов и календаря.

Процессор использует следующий формат для отображения даты и времени.

- Дата: Год, Месяц, Дата, День недели (0 - 6, с Воскресенья по Субботу)
- Время: 24-часовой формат, Часы, Минуты Секунды.

Отображение на ручном программаторе

94/01/02 23:08:17

Вы можете использовать AUX функцию для изменения любого элемента даты и времени. Однако процессор не будет автоматически корректировать расхождение между датой и днем недели. Например, если вы изменили дату на 15-ое число месяца, а 15-ое число является четвергом, то вы должны изменить день недели (если только процессор уже не показывает дату как четверг).

Переменное/ Фиксированное время сканирования

X	X	√
430	440	450

Процессор DL450 предусматривает три типа установки времени сканирования:

- Переменное — это стандартная установка времени сканирования, когда ПЛК сканирует столько времени, сколько занимает выполнение пользовательской программы.
- Фиксированное — время сканирования может устанавливаться с помощью константы, от 10 мс до 9999 мс. Операционная система вставляет задержку после выполнения каждой программной цепочки, чтобы реализовать фиксированное время сканирования.
- Ограниченное — ПЛК функционирует также как и при переменном времени сканирования, но генерирует ошибку превышения лимита времени, если время сканирования превышает заданное значение. Вы можете, например, при ошибке прервать выполнение программы.

При выборе желательного варианта времени сканирования для DL450 используйте DirectSOFT, войдите в диалоговый режим. Вызовите меню ПЛК, затем опцию Диагностика (Diagnostics), далее опцию Время Сканирования (Scantime), потом Настройка (Setup). Появятся три варианта: Переменное (Variable), Фиксированное (Fixed) и Ограниченное (Limit).

Защита с помощью пароля

Процессоры DL405 имеют защиту с помощью пароля, чтобы предотвратить несанкционированный доступ к программам или данным. Используйте AUX - функции 81, 82 и 83 соответственно для изменения пароля, разблокировки и блокировки процессора. Паролем может быть 8-значный цифровой (0 - 9) код. После ввода правильного пароля вы можете изменять его, заменить его всеми нулями, то есть снять защиту паролем. Процессор поставляется с заводским паролем 00000000.

Многоуровневый пароль (только в DL450). Процессор DL450 имеет промежуточный уровень защиты, который задается, если в пароле первым символом является символ "A". Остальные 7 символов должны быть цифровыми (0 - 9). Промежуточный уровень защиты паролем отличается от стандартного тем, что он позволяет с устройства интерфейса оператора изменять данные V-памяти, такие как заданные значения. Однако он не позволяет редактировать пользовательские программы.

Более подробная информация по паролям приведена в Приложении по вспомогательным функциям.

Вспомогательные функции

Многие задачи настройки процессора предполагают использование вспомогательных (AUX) функций. AUX функции выполняют много различных операций, начиная с выбора рабочего режима и кончая копированием программ в картриджи памяти. Они разделены на категории, которые предназначены для различных системных параметров. В приложении A приводится подробное описание этих AUX функций.

Вы можете получить доступ к AUX функциям из DirectSOFTTM или с ручного программатора с помощью специального меню. 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 DirectSOFT. В таблице ниже приводится перечень вспомогательных (AUX) функций для различных процессоров и ручного программатора.

AUX Функции и их описание		430	440	450	HPP
AUX 1* — Рабочие режимы					
11	Перейти в Рабочий Режим (RUN)	✔	✔	✔	-
12	Перейти в Режим Отладки (TEST)	✔	✔	✔	-
13	Перейти в Режим Программирования (PGM)	✔	✔	✔	-
14	Редактировать в Рабочем Режиме (Run Time Edit)	×	✔	✔	-
AUX 2* — Операции RLL					
21	Проверка Программы	✔	✔	✔	-
22	Изменение ссылки	×	✔	✔	-
23	Очистка памяти в диапазоне цепей	✔	✔	✔	-
24	Очистка памяти во всех цепях	✔	✔	✔	-
25	Выбрать Картридж памяти или Флэш-память	×	×	✔	-
26	Копировать содержимое Картриджа памяти во Флэш -память	×	×	✔	-
27	Копировать содержимое Флэш-памяти в Картридж памяти	×	×	✔	-
28	Сверить содержимое Флэш-памяти и Картридж памяти	×	×	✔	-
AUX 3* — Операции с V-Памятью					
31	Очистка V-памяти	✔	✔	✔	-
32	Стереть область V-памяти	✔	✔	✔	-
33	Найти значение в V-памяти	×	✔	✔	-
AUX 4* — Конфигурация ввода/вывода					
41	Показать конфигурацию ввода/вывода	✔	✔	✔	-
42	Диагностика ввода/вывода	✔	✔	✔	-
44	Проверка конфигурации ввода/вывода при включении питания	✔	✔	✔	-
45	Выбрать конфигурацию	✔	✔	✔	-
46	Конфигурировать ввод/вывод	×	✔	✔	-
47	Интеллектуальный ввод/вывод	✔	✔	✔	-

✔ — поддерживается

× — не поддерживается

– — не применяется

AUX Функции и их описание		430	440	450	HPP
AUX 5* — Конфигурация Процессора					
51	Изменить Имя программы	✔	✔	✔	-
52	Показать / Изменить Календарь	×	✔	✔	-
53	Показать Время сканирования	✔	✔	✔	-
54	Инициализировать Электронный блокнот	✔	✔	✔	-
55	Установить сторожевой таймер	✔	✔	✔	-
56	Конфигурировать последовательные порты	✔	✔	✔	-
57	Установить Области Сохранения	✔	✔	✔	-
58	Операции Тестирования	✔	✔	✔	-
5C	Показать архив ошибок	×	✔	✔	-
5D	Выбрать режим сканирования ПЛК	×	×	✔	-
AUX 6*— Конфигурация Ручного Программатора					
61	Показать номер версии	✔	✔	✔	
62	Включить/Отключить звуковой сигнал	×	×	×	✔
63	Отключить/включить подсветку индикации	×	×	×	✔
64	Выбрать режим работы с процессором /автономный режим	×	×	×	✔
65	Запустить самодиагностику	×	×	×	✔
AUX 7* — Операции с картриджем памяти					
71	Копировать память процессора в Картридж	✔	✔	✔	✔
72	Копировать память Картриджа в процессор	✔	✔	✔	✔
73	Сравнить память Картриджа с памятью процессора	✔	✔	✔	✔
74	Проверить очистку памяти на Картридже	×	×	×	✔
75	Очистить память Картриджа	×	×	×	✔
76	Показать тип Картриджа	✔	✔	✔	✔
77	Копировать магнитную ленту в Картридж	×	×	×	✔
78	Копировать Картриджа на магнитную ленту	×	×	×	✔
79	Сравнить магнитную ленту с Картриджем	×	×	×	✔
AUX 8* — Операции с паролем					
81	Изменить пароль	×	✔	✔	-
82	Разблокировать процессор	×	✔	✔	-
83	Заблокировать процессор	×	✔	✔	-

Стирание существующих программ

Перед вводом новой программы вы всегда должны чистить память релейных программ. Вы можете использовать AUX функцию 24 для полного удаления программы.

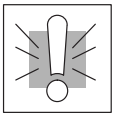
Вы можете использовать другие AUX функции для очистки других областей памяти.

- AUX 23 — Очистка памяти в диапазоне цепей
- AUX 24 — Очистка памяти во всех цепях
- AUX 31 — Очистка V-памяти

Инициализация системной памяти «scratchpad»

Процессоры DL405 поддерживает системные параметры в области памяти, которую часто называют ”электронный блокнот”. В некоторых случаях вы можете вносить изменения в настройку системы, которая хранится в системной памяти. Например, если вы определяете область Управляющих Реле (CR) как сохраняемую, то эти изменения запоминаются.

AUX 54 возвращает системную память к значениям по умолчанию.



ПРЕДУПРЕЖДЕНИЕ. У вас нет необходимости применять эту функцию до тех пор, пока вы не захотите удалить какую-либо информацию по настройке, которая хранится в системной памяти. 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.

Напоминаем, что данная AUX функция восстанавливает всю системную память. Если вы установили специальные параметры, например, сохраняемые области, то вам необходимо повторно ввести данные.

Установка сетевого адреса Процессора

Поскольку процессоры DL440 и DL450 имеют встроенные порты DirectNET, Вы можете использовать ручной программатор для установки сетевого адреса порта и параметров связи порта. Параметрами настройки по умолчанию являются:

- Адрес станции ”1”
- Шестнадцатеричный режим (Hex mode)
- Контроль по нечетности (Odd parity)

В Руководстве по DirectNET приводится дополнительная информация по настройкам линий связи при работе в сети. Если Вы используете нижний порт для устройства программирования, то можете применять только настройки по умолчанию. По дополнительным двум портам DL450 обратитесь к главе 4 ”Проектирование и конфигурирование системы”.

Используйте AUX функцию 56 для установки сетевого адреса и параметров связи для вторичного порта(ов).

Установка областей сохранения памяти

Процессоры DL405 обеспечивает по умолчанию определенные области сохранения памяти. Эти области сохранения по умолчанию пригодны для многих приложений, но вы можете изменить их, если ваше приложение требуют дополнительных областей сохранения или вообще не требует каких-либо областей сохранения. Областями по умолчанию являются:

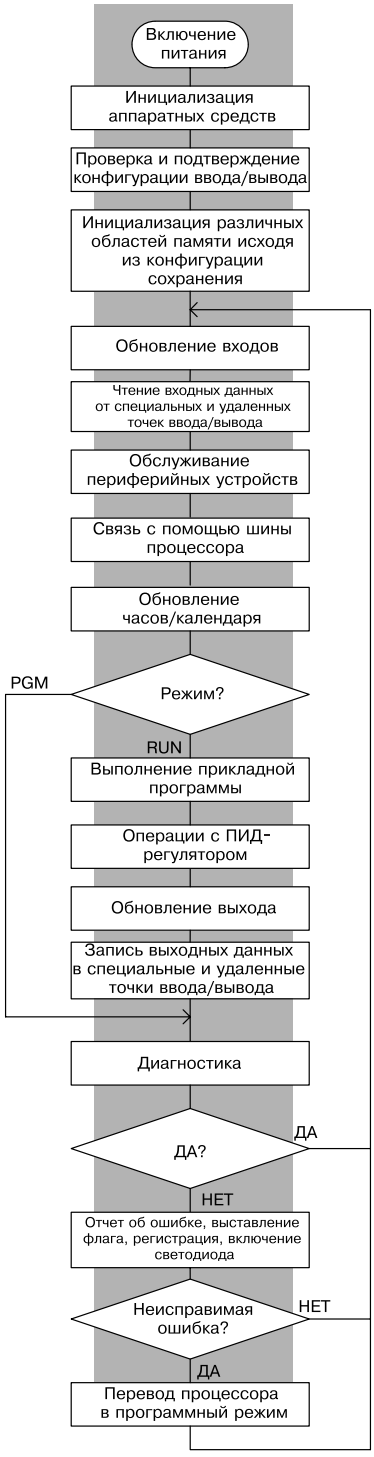
- Управляющие Реле — C600 - C737
- V-память — V2000 - V7377
- Таймеры — Не устанавливаются по умолчанию (но Вы можете сделать их сохраняемыми)
- Счетчики — CT0 - CT177
- Стадии — Не устанавливаются по умолчанию (но Вы можете сделать их сохраняемыми)

Используйте AUX функцию 57 для установки областей сохранения.

Работа процессора

Для того чтобы добиться надлежащего управления вашей аппаратурой или процессом, необходимо хорошо понимать, как процессоры DL405 управляют всеми аспектами работы системы. На приведенной ниже блок-схеме показаны основные задачи операционной системы процессора. В данном разделе изучаются следующие четыре аспекта работы процессора:

- **Операционная система процессора.** Процессор управляет всеми аспектами работы системы.
- **Режимы работы процессора.** Тремя основными режимами работы являются: Программный режим (Program), Рабочий режим (Run) и режим Тестирования (Test).
- **Временные характеристики процессора.** Обсуждаются две важных временных характеристики: время отклика ввода/вывода и время сканирования процессора.
- **Карта распределения памяти процессора.** Карта распределения памяти процессора отображает адреса различных системных средств, например, таймеров, счетчиков, входов и выходов.



Операционная система процессора

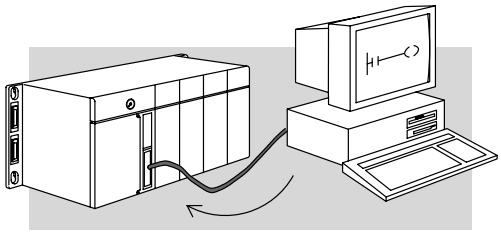
При включении питания процессор инициализирует внутренние аппаратные средства. Инициализация памяти начинается с проверки установок сохраняемой памяти. В общем случае, содержимое сохраняемой памяти поддерживается, а не сохраняемая память очищается (если не определено иное). После однократного просмотра задач при включении питания процессора начинает циклические операции сканирования. На блок-схеме справа показано, как различается обработка задач в зависимости от режима процессора и наличия ошибок. "Время сканирования" определяется как среднее время обхода всех задач. Следует отметить, что процессор всегда считывает входы, даже в программном режиме. Это позволяет программным средствам отслеживать состояние входов в любой момент времени. Выходы обновляются только в рабочем режиме. В программном режиме они отключены.

В рабочем режиме процессор выполняет программу пользователя. Сразу после этого выполняется каждый сконфигурированный контур ПИД - регулирования (только для DL450). Затем процессор записывает выходные результаты этих двух задач в соответствующие выходные точки.

Обнаруженные ошибки имеют два уровня. По исправимым ошибкам формируется отчет, а процессор остается в текущем режиме. Если обнаруживаются неисправимые ошибки (Fatalerror), то процессор переводится в Программный режим, а выходы сбрасываются.

Работа в программном режиме

В программном режиме процессор не выполняет прикладные программы и не обновляет выходные модули. Основным назначением программного режима является ввод или изменение прикладных программ. Вы можете использовать программный режим для установки параметров процессора, таких как сетевой адрес, сохраняемые области памяти и др.



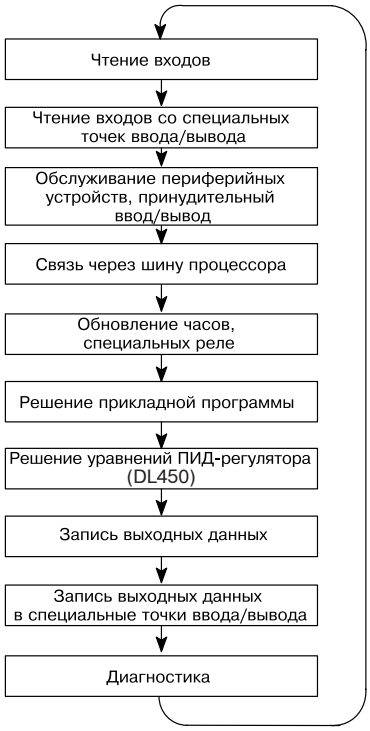
Загрузка программ

Вы можете использовать переключатель режимов в процессоре DL450 для выбора работы в программном режиме. Или при переключателе в положении TERM вы можете использовать программирующее устройство, например ручной программатор, для перевода процессора в программный режим.

Работа в рабочем режиме

В рабочем режиме процессор выполняет прикладную программу, делает вычисления для конфигурирования контура ПИД -регулятора и обновляет систему ввода/вывода. В рабочем режиме вы можете выполнять многие операции. Некоторые из них:

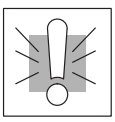
- Контроль и изменение состояния точек ввода/вывода.
- Обновление параметров настройки таймеров/счетчиков.
- Обновление ячеек памяти переменных.



Работу в рабочем режиме можно разделить на несколько ключевых зон. Важно понять, как эти зоны влияют на результаты решений вашей прикладной программы.

Вы можете использовать переключатель режимов для выбора работы в рабочем режиме(DL440 и DL450). Или при переключателе, установленном в положении TERM, вы можете использовать устройство для программирования, например ручной программатор, для перевода процессора в рабочий режим.

В рабочем режиме вы можете также редактировать программы. Редакционные изменения в рабочем режиме не являются ”безударными”. Пока принимается информация по новой программе, процессор поддерживает последнее состояние выходных сигналов. Но если в новой программе обнаружена ошибка, то процессор отключает все выходы и переходит в программный режим.



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 травмирования персонала или повреждения оборудования.

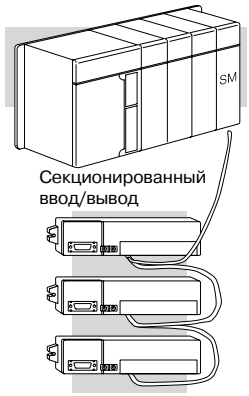
Чтение входов

Процессор считывает состояние всех входов и записывает их в регистр отображения. Ячейки регистра отображения входов обозначаются X и далее следует номер ячейки памяти. Данные регистра отображения используются процессором при решении прикладной программы.

Конечно, вход может измениться после того, как процессор считает входы. В общем случае время сканирования процессора измеряется миллисекундами. Если ваше приложение не может ждать следующего обновления входов/выходов, то вы можете использовать команды немедленного действия. Эти команды при решении прикладной программы не пользуются состоянием регистра с входными данными. Команды немедленного действия прямо считывают состояние входов прямо с модулей ввода/вывода. Однако применение команд немедленного действия удлиняет сканирование, поскольку процессор должен повторно считывать состояние точек ввода/вывода. Полный список команд немедленного действия приведен в главе 5.

Чтение входов специальных модулей и удаленного ввода/вывода

После того, как процессор считает входы из входных модулей, он считывает входные данные всех установленных специальных модулей, например, с модулей интерфейсов счетчиков и др. Одновременно производится сканирование и чтение состояния входов из устройств удаленного ввода/вывода. Тип данных GX используется как для удаленных входов, так и для удаленных выходов. (В Руководстве по удаленному и секционированному вводу/выводу DL405 дается более подробная информация о том, как устанавливать связь с удаленным вводом/выводом).



ПРИМЕЧАНИЕ. Может показаться, что состояние точек удаленного ввода/вывода обновляется при каждом сканировании. Но это не совсем верно. Процессор получает информацию от ведущего модуля удаленного ввода/вывода при каждом сканировании, но ведущее устройство удаленного ввода/вывода может не получить обновленную информацию со всех удаленных ведомых устройств. Напоминаем, что связь с удаленным вводом/выводом управляется ведущим устройством удаленного ввода/вывода, а не процессором.

Обслуживание периферийных устройств и форсирование ввода/вывода

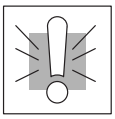
После считывания входов из входных модулей процессор опрашивает все подсоединенные периферийные устройства. Это в основном обслуживание коммуникаций подсоединенных устройств. Например, может запрашиваться устройство для программирования с целью определения того, нуждается ли оно в изменении состояния входов, выходов или состояния другого типа памяти.

Принудительный ввод/вывод — временно изменяет состояние бита дискретного сигнала. Например, вы желаете включить вход, хотя реально его нет. Это дает вам возможность изменять состояние точки, которое хранится в регистре отображения. Это значение может 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.

Принудительные входы — Процессор считывает состояние входов X в части цикла сканирования "Считывание входов". Когда процессор опрашивает программирующее устройство, он регистрирует любой запрос на принудительную установку входа X. Если вход используется в прикладной программе, то контакт X программы считается закрытым (on). Поскольку вход X — это реальный физический вход, то процессор изменит его состояние в очередном цикле сканирования.

Принудительные выходы — Выходы, которые не используются в программе, могут быть принудительно включены или отключены при поиске неисправностей и при обслуживании. Вы можете временно разрешить принудительную установку любого выхода, вставив команду END в начале пользовательской программы. Затем, пользуясь DirectSOFT или ручным программатором, принудительно включать или отключать выходы.

Процессоры DL405 сохраняют принудительное значение только в течение одного цикла сканирования, при условии, что точки ввода/вывода соответствуют реальным точкам модуля в системе. Однако, если адрес очки больше любого адреса реального ввода/вывода или он не используется в прикладной программе, то такая точка будет сохранять принудительное состояние.



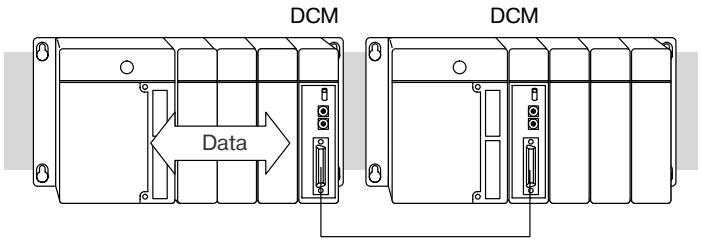
ПРЕДУПРЕЖДЕНИЕ. Только квалифицированные лица, знающие в полном объеме все аспекты приложения, могут вносить изменения в программу. Убедитесь, что Вы рассмотрели все последствия любых изменений и минимизировали риск травмирования персонала или повреждения оборудования.

Обновление специальных реле и специальных регистров

Существуют определенные ячейки V-памяти, которые содержат информацию регистров. Данная часть цикла выполнения гарантирует, что эти ячейки будут обновляться при каждом цикле сканирования. Аналогично, имеется несколько разных специальных реле, таких как диагностические реле и др., которые также обновляются в данном сегменте сканирования.

Связь через шину процессора

Многие специальные модули, например, модуль коммуникации данных, модули сопроцессора FACTS, могут передавать данные в процессор и из процессора по системной шине на системной плате. Состав этих данных шире, чем стандартное состояние точек ввода/вывода. Этот тип связи может быть только в рамках (локального) каркаса процессора. Это — только часть исполнительного цикла, используемого для взаимодействия с этими модулями. Процессор выполняет как чтение, так и запись запросов в данном сегменте.



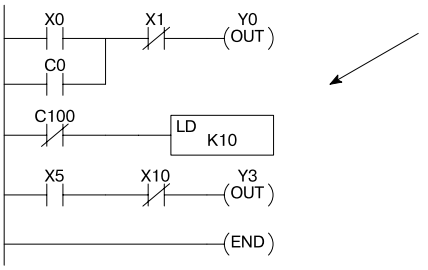
Обновление часов, специальных реле и специальных регистров

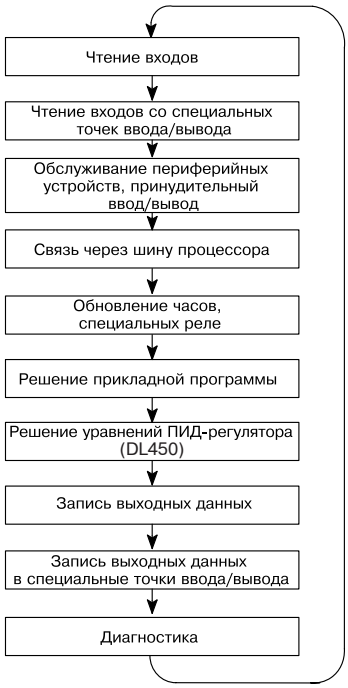
Процессоры DL440 и DL450 имеют внутренние часы реального времени и календарь, которые доступны прикладным программам. Специальные ячейки V-памяти поддерживают эту информацию. Данная часть исполнительного цикла обеспечивает обновление этих ячеек при каждом сканировании. Кроме того, имеется несколько различных специальных реле, например, диагностические реле и др., которые также обновляются в данном сегменте.

Решение прикладных программ

В данном сегменте цикла сканирования процессор проводит вычисления по каждой команде прикладной программы. Команды определяют отношение между состояниями входов и выходами системы.

Процессор начинает с первой цепочки программы релейной логики, выполняет вычисления слева направо и сверху вниз. Процесс продолжается по ступеням, пока не встретится команда END. В этом месте формируется новое состояние выходных данных.





Внутренние управляющие реле (C), стадии (S) и память переменных (V) также обновляются в данном сегменте.

Следует напомнить, что процессор может получать и хранить информацию о принудительных значениях при обслуживании периферийных устройств. Если на какие-то точки ввода/вывода или на данные памяти было оказано принудительное воздействие, регистр отображения выходов будет содержать эту информацию.

ПРИМЕЧАНИЕ. Если какая-либо выходная точка использовалась в прикладной программе, то результаты программного решения затрут информацию любого принудительного воздействия, которая сохранялась до этого. Например, если Y0 с помощью программирующего устройства было принудительно установлено в состояние "включено", а ступень, содержащая Y0, была рассчитана так, что Y0 должно быть отключено, то регистр отображения выходов покажет Y0 в состоянии "отключено". Конечно, вы можете принудительно воздействовать на выходную точку, которая не используется в прикладной программе. В этом случае точка сохранит принудительное значение, так как отсутствует решение прикладной программы.

Решение уравнений контура ПИД-регулятора

X	X	√
430	440	450

Процессор DL450 может обрабатывать до 16 встроеных контуров ПИД-регулятора. Расчет контуров запускается как отдельная задача сразу же после выполнения программы релейной логики. Рассчитываются только контуры, которые уже сконфигурированы, сами расчеты выполняются только в соответствии со встроенной в контур программой — планировщиком. Период дискретизации (интервал расчета) каждого контура программируется независимо. Обратитесь к главе 8 "Организация контуров ПИД- регулятора" за более подробной информацией о влиянии расчетов контуров ПИД-регулятора на общее время сканирования процессора.

Запись выходных данных

После того, как прикладная программа выполнила логическую схему команд и сформировала регистр отображения выходных данных, процессор записывает содержание регистра отображения выходных данных в соответствующие выходные точки в локальном каркасе процессора или в локальных каркасах расширения. Следует напомнить, что процессор также обеспечивает операции принудительного воздействия, изменения по которым хранятся в регистре отображения выходных данных, точки с принудительным воздействием обновляются в соответствии с состоянием, определенным выше.

Запись выходных данных в специальные и удаленные точки ввода/вывода

После обновления выходов в локальном каркасе и каркасах расширения процессор посылает в выходные точки информацию, требуемую всеми установленными специальными модулями. В частности, при этом сканировании состояние выходов переписывается из регистра отображения в устройства удаленного ввода/вывода .

ПРИМЕЧАНИЕ. Может показаться, что состояние точек удаленного ввода/вывода обновляется при каждом сканировании. Это не совсем верно. Процессор посылает информацию ведущему модулю удаленного ввода/вывода при каждом сканировании, но ведущий модуль будет обновлять реальныеудаленные модули в течение последующих циклов связи между ведущим и ведомыми модулями. Напоминаем, что передачей данных по каналам связи с удаленного ввода/вывода управляет ведущий модуль удаленного ввода/вывода, а не процессор.

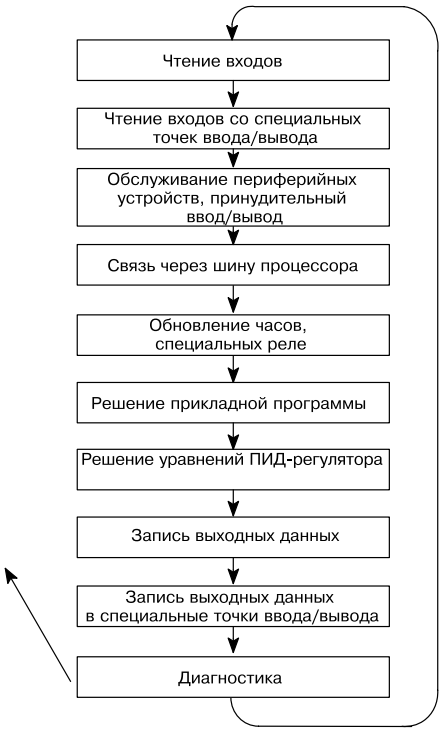
Диагностика

В данной части цикла сканирования процессор выполняет всю системную диагностику и решает другие задачи, такие как:

- вычисление времени сканирования,
- обновление специальных реле,
- сброс сторожевого таймера.

Процессоры DL405 автоматически обнаруживает и регистрирует многие ошибочные состояния. В Приложении B приводится список кодов различных ошибок, имеющих место в системе DL405.

Одной их наиболее важных задач диагностики является расчет времени сканирования и управление сторожевым таймером. Процессоры DL405 имеют таймер — ”сторож”, в котором хранится максимально допустимое время, в течение которого процессор должен закончить прикладной сегмент цикла сканирования. Его значение по умолчанию, установленное в заводских условиях, равно 200 миллисекунд. Если это время будет превышено, то процессор войдет в программный режим, сбросит все выходы и выдаст сообщение об ошибке. Например, если время сканирование будет превышено, то на ручном программаторе отобразится сообщение ”E003 S/W TIMEOUT”.



Вы можете использовать функцию AUX 53 для просмотра минимального, максимального и текущего времени сканирования. Можете использовать функцию AUX 55 для того, чтобы уменьшить или увеличить значение сторожевого таймера. Имеется также команда RSTWT, с помощью которой можно из прикладной программы сбросить сторожевой таймер во время сканирования процессора.

Время отклика ввода/вывода

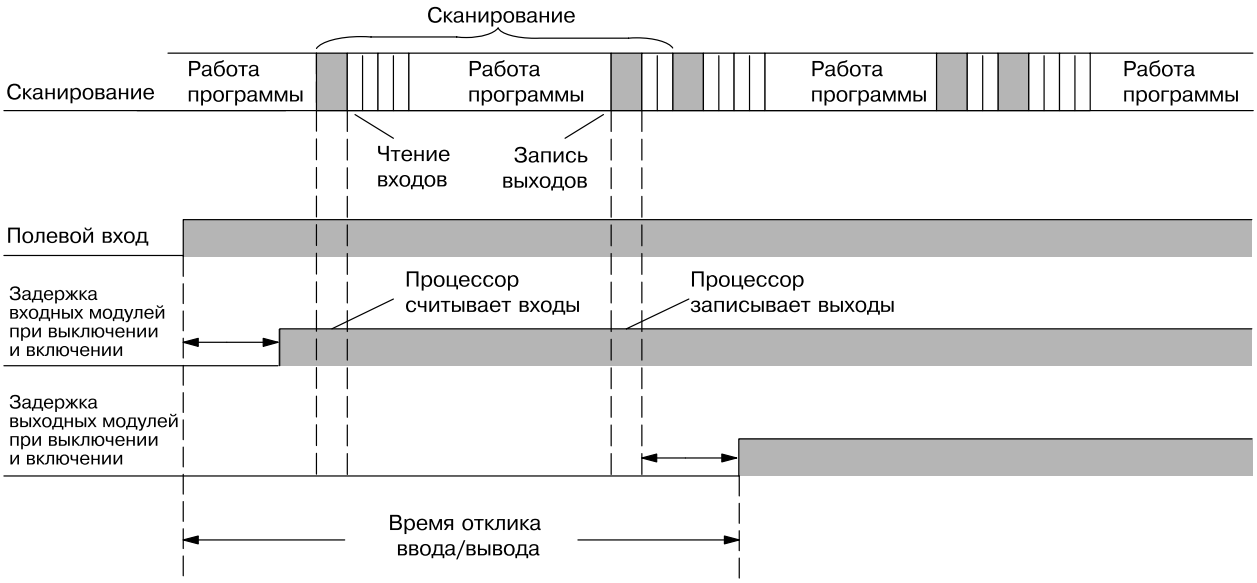
Важно ли быстродействие для вашего приложения ?

Временем отклика системы ввода/вывода является интервал времени, необходимый системе управления, чтобы считать изменения во входных точках и обновить соответствующие выходные точки. В большинстве приложений Процессор выполняет задачи практически мгновенно. Однако некоторые приложения требуют чрезвычайно быстрого обновления. Существует четыре момента, которые могут повлиять на время отклика:

- Момент в периоде сканирования, когда полевые входы меняют свое состояние.
- Время задержки входных модулей на выключение и включение.
- Время сканирования процессора.
- Время задержки выходных модулей на выключение и включение.

Нормальное минимальное время отклика ввода/вывода

Время отклика ввода/вывода будет наименьшим, когда модуль считывает изменения входов до начала Опроса Входов исполнительного цикла. В этом случае считываются входные состояния, решается прикладная программа и обновляются выходные точки. На следующей диаграмме показан пример распределения времени для этого случая.



В этом случае вы можете вычислить время отклика простым суммированием следующих элементов.

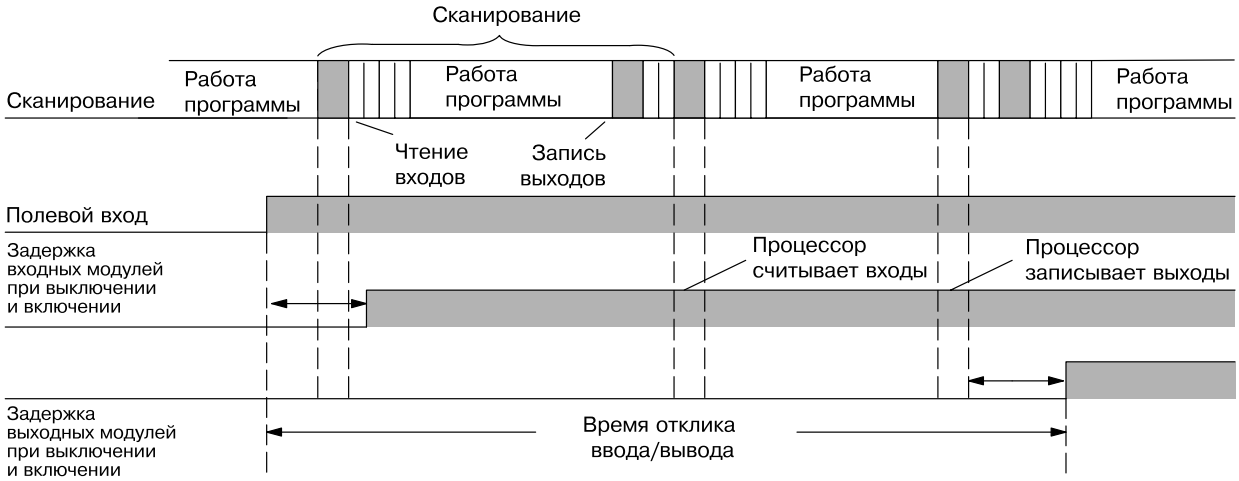
Задержка входа + Время сканирования + Задержка выхода = = Время отклика

Нормальное максимальное время отклика ввода/вывода

Время отклика ввода/вывода будет наибольшим, когда модуль считывает изменения входов после Опроса Входов исполнительного цикла. В этом случае новое состояние входа не будет считываться до следующего сканирования. На следующей диаграмме показан пример распределения времени для этого случая.

В этом случае вы можете вычислить время отклика простым суммированием следующих элементов.

Задержка входа + (2 x Время сканирования) + Задержка выхода = = Время отклика

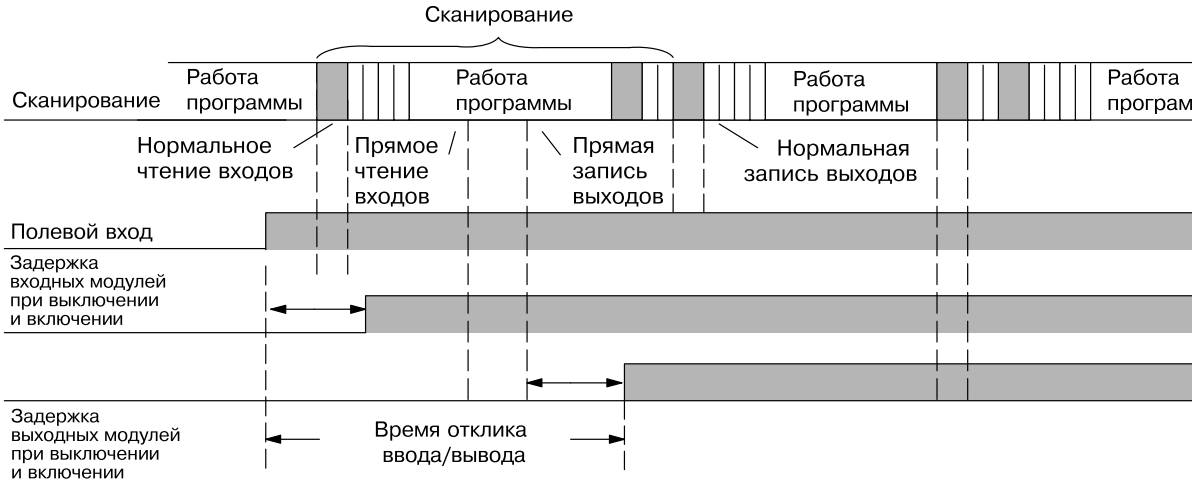


Улучшенное время отклика

Существует несколько способов, позволяющих повысить пропускную способность.

- Выбрать команды с более коротким временем выполнения.
- Использовать команды ввода/вывода немедленного действия (которые обновляют точки ввода/вывода в исполнительном сегменте программы релейной логики).
- Выбрать модули, которые имеют более короткое время отклика.

Команды ввода/вывода немедленного действия, вероятно, являются наиболее полезным способом. В следующем примере показаны команды ввода и вывода немедленного действия и их результат.



В этом случае вы можете вычислить время отклика простым суммированием следующих элементов.

Задержка входа + Время выполнения команд + Задержка выхода = Время отклика

Время выполнения команд вычисляется суммированием времен для команды немедленного ввода, команды немедленного вывода и всех команд между ними.

ПРИМЕЧАНИЕ. Когда команда немедленного действия считывает текущее состояние с модуля, она использует полученный результат для решения задач, выполняя это в рамках одной команды без обновления значений в регистре отображения. Поэтому любые обычные команды, которые последуют, будут по-прежнему использовать значения регистра отображения. Любые последующие команды немедленного действия будут снова обращаться к модулю, чтобы обновить состояние.

Анализ времени сканирования процессора

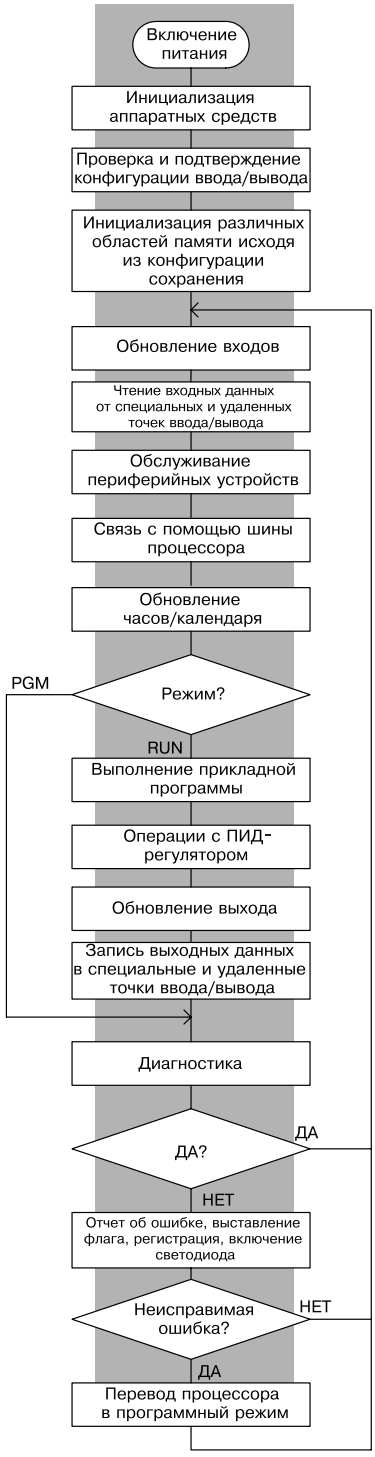
Время сканирования включает все циклические задачи, которые выполняются операционной системой. Можно использовать DirectSOFT или ручной программатор для отображения минимального, максимального и текущего времен сканирования, которые имели место после последнего перехода из программного режима в рабочий режим. Эта информация может быть весьма полезной для оценки эксплуатационных характеристик системы.

Как показано выше существует несколько сегментов, которые составляют цикл сканирования. Каждый из этих сегментов требует определенного времени на выполнение. Из всех сегментов только на длительность одного вы можете реально влиять — на время выполнения прикладной программы. Это связано с тем, что различные команды имеют разное время выполнения. Поэтому, если вы хотите иметь быстрое сканирование, то вы должны попытаться выбрать более быстрые команды.

Выбор модулей ввода/вывода и конфигурация системы, например, расширение или удаленный ввод/вывод, также влияют на время сканирования. Однако этот выбор, как правило, диктуется приложением.

Например, если вам необходимо считать импульсы при высокой скорости, вы, вероятно, должны использовать модуль высокоскоростного счетчика. Аналогично, если вы имеете пункты ввода/вывода, которые необходимо разместить в нескольких сотнях метров от процессора, то вам нужен удаленный ввод/вывод, так как гораздо быстрее и дешевле установить один кабель удаленного ввода/вывода , чем прокладывать все сигнальные провода для каждого конкретного пункта ввода/вывода.

В следующих параграфах приводится общая информация о том, сколько времени требуют отдельные сегменты.



Процесс инициализации

Процессор выполняет задачу инициализации сразу после включения питания. Требуемое для этого время зависит от загрузки системы, например, от числа установленных модулей ввода/вывода, типа используемого картриджа памяти и др. Задача инициализации выполняется сразу же после включения питания, она не влияет на время сканирования для прикладной программы.

Инициализация	DL430	DL440	DL450
Минимальное время	1.2 сек	1.0 сек	1.9 сек
Максимальное время	3.2 сек	2.5 сек	3.3 сек

Чтение входов

Время, необходимое для считывания состояния входных модулей локального каркаса и каркаса расширения зависит, от того, какой используется процессор, и от числа входных точек в этих каркасах. В следующей таблице показаны типовые времена обновления, требуемые процессором.

Временные факторы	DL430	DL440	DL450
Служебное время	20.0 мкс	14.5 мкс	20.0 мкс
На входной модуль	48.0 мкс	22.6 мкс	13.0 мкс
На входную точку	4.0 мкс	2.5 мкс	6.3 мкс

Например, время, необходимое DL430 для чтения двух входных модулей по 16 точек, можно рассчитать следующим образом. Пусть NM -число модулей, а NI — общее число входных точек.

Формула:

Время = 20.0 мкс + (48 мкс x NM) + (4.0 мкс x NI)

Пример:

Время = 20.0 мкс + (48 мкс x 2) + (4.0 мкс x 16)

Время = 180 мкс

ПРИМЕЧАНИЕ. Данная информация относится к времени, которое процессор тратит на считывание входных состояний из модулей. Не надо путать его с временем отклика ввода/вывода, которое было рассмотрено выше.

Чтение входов со специальных точек ввода/вывода

В данной части цикла процессор считывает все входные точки, относящиеся:

- к удаленному вводу/выводу
- к специальным модулям (например, к высокоскоростному счетчику и др.)

Время, необходимое для считывания входных состояний из этих модулей зависит от того, какой используется процессор, от числа модулей и числа входных точек.

Специальные модули	DL430	DL440	DL450
Служебное время	32.0 мкс	20.0 мкс	20.0 мкс
На модуль (входной)	100.0 мкс	67.0 мкс	13.0 мкс
На входную точку	80.0 мкс	54.0 мкс	13.8 мкс
Удаленные модули	**DL430**	**DL440**	**DL450**
Служебное время	32.0 мкс	22.0 мкс	19.0 мкс
На модуль (входной)	150.0 мкс	100.0 мкс	62.0 мкс
На входную точку	25.0 мкс	17.0 мкс	11.2 мкс

Например, время, необходимое DL430 для чтения двух входных модулей по 32 точки (размещенных в удаленном каркасе) и входных точек, относящихся к одному модулю высокоскоростного счетчика, можно рассчитать следующим образом. Пусть NM — число модулей, а NI — общее число входных точек.

Удаленный ввод/вывод	**Удаленный ввод/вывод**
Формула:	**Формула:**
Время = 32 мкс + (150 мкс x NM) + (25 мкс x NI)	Время = 32 мкс + (100 мкс x NM) + (80 мкс x NI)
Пример:	**Пример:**
Время = 32 мкс + (150 мкс x 2) + (25 мкс x 32)	Время = 32 мкс + (100 мкс x 1) + (25 мкс x 16)
Время = 1832 мкс	Время = 3244 мкс

Обслуживание Периферийных устройств

Общее время = 3244 мкс

Запросы на коммуникации могут появиться в любой момент при сканировании, однако процессор только "регистрирует" эти запросы до их обслуживания в части "Обслуживание Периферийных Устройств" цикла сканирования. (Процессор не тратит время на обслуживание, если отсутствует

Регистрация запроса (в любой момент)		DL430	DL440	DL450
Отсутствие соединения	Мин. и Макс.	0 мкс	0 мкс	0 мкс
Порт 0	Отправление мин./макс	52 / 62 мкс	40 / 48 мкс	38 / 38 мкс
	Получение мин./макс	60 / 78 мкс	52 / 63 мкс	45 / 45 мкс
Порт 1	Отправление мин./макс	60 / 78 мкс	46 / 50 мкс	41 / 48 мкс
	Получение мин./макс	60 / 86 мкс	66 / 70 мкс	47 / 59 мкс
Порт 2	Отправление мин./макс	Нет данных	Нет данных	41 / 48 мкс
	Получение мин./макс	Нет данных	Нет данных	47 / 59 мкс
Порт 3	Отправление мин./макс	Нет данных	Нет данных	38 / 38 мкс
	Получение мин./макс	Нет данных	Нет данных	45 / 45 мкс

В части "Обслуживание Периферийных Устройств" цикла сканирования процессор анализирует запросы на коммуникации и отвечает соответствующим образом. Время, необходимое для обслуживания периферийных устройств, зависит от содержания запроса.

Обслуживание запроса	DL430	DL440	DL450
Минимальное	170 мкс	120 мкс	96 мкс
Максимальное в рабочем режиме	18 мкс	26 мкс	160 мкс
Максимальное в программном режиме	3 сек	15 сек	11.2 сек.

Связь через шину процессора

Некоторые специальные модули могут взаимодействовать также непосредственно с процессором через шину процессора. В данной части цикла процессор полностью отрабатывает все коммуникации через шину. Реально необходимое время зависит от типа установленного модуля и типа обрабатываемого запроса.

ПРИМЕЧАНИЕ. Некоторые специальные модули могут иметь значительное влияние на продолжительность сканирования процессора. Если время критично для вашего приложения, обратитесь к документации по модулю для получения информации о том, как он влияет на время сканирования.

Обновление часов/календаря, специальных реле, специальных регистров

На этом этапе обновляются часы, календарь и специальные реле, результаты загружаются в специальные ячейки V-памяти. Такое обновление выполняется как в рабочем, так и в программном режимах.

Режимы		DL430	DL440	DL450
Программный режим	Минимум	8.0 мкс фиксированное	35 мкс	12.0 мкс
	Максимум	8.0 мкс фиксированное	48.0 мкс	12.0 мкс
Рабочий режим	Минимум	20.0 мкс	60 мкс	22.0 мкс
	Максимум	26.0 мкс	85 мкс	29.0 мкс

Запись выходных данных

Время, необходимое для записи состояний выходов для локальных модулей и модулей расширения ввода/вывода зависит от того, какой используется процессор, и от числа выходных точек в каркасе. В следующей таблице показаны типовые времена обновления, требуемые процессором.

Временные факторы	DL430	DL440	DL450
Служебное время	20.0 мкс	12.6 мкс	15.0 мкс
На выходной модуль	45.0 мкс	21.0 мкс	13.0 мкс
На выходную точку	4.0 мкс	2.5 мкс	14.1 мкс

Например, время, необходимое DL430 для записи данных для двух выходных модулей по 32 точки, можно рассчитать следующим образом (где NM — число модулей, NO — общее число выходных точек).

Формула:

Время = 20 мкс + (45.0 мкс x NM) + (4 мкс x NO)

Пример:

Время = 20 мкс + (45.0 мкс x 2) + (4 мкс x 32)

Время = 238 мкс

Запись выходных данных в специальные точки ввода/вывода

В данной части цикла процессор записывает во все выходные точки, относящиеся:

- к удаленному вводу/выводу,
- к специальным модулям (например, высокоскоростному счетчику и др.)

Время, необходимое для записи всех данных регистра отображения выходов в эти модули, зависит от того, какой используется процессор, от числа модулей и числа выходных точек.

Специальные модули	DL430	DL440	DL450
Служебное время	32.0 мкс	20.0 мкс	18.0 мкс
На модуль (с выходами)	100.0 мкс	67.0 мкс	13.0 мкс
На выходную точку	80.0 мкс	54.0 мкс	14.1 мкс
Удаленные модули	**DL430**	**DL440**	**DL450**
Служебное время	32.0 мкс	22.0 мкс	15.0 мкс
На модуль (с выходами)	150.0 мкс	100.0 мкс	54.0 мкс
На входную точку	25.0 мкс	17.0 мкс	13.9 мкс

Например, время, необходимое DL430 для записи в два выходных модуля по 32 точки (размещенные в удаленном каркасе) и выходных точек, относящихся к одному модулю высокоскоростного счетчика, можно рассчитать следующим образом. Пусть NM — число модулей, а NO — общее число выходных точек.

Удаленный ввод/вывод

Формула:

Время = 32 мкс + (150 мкс x NM) + (25 мкс x NO)

Пример:

Время = 32 мкс + (150 мкс x 2) + (25 мкс x 32)

Время = 1832 мкс

Высокоскоростной счетчик

Формула:

Время = 32 мкс + (100 мкс x NM) + (80 мкс ’ NO)

Пример:

Время = 32 мкс + (100 мкс x 1) + (80 мкс x 4)

Время = 452 мкс

Общее время = 2284 мкс

ПРИМЕЧАНИЕ: Это общее время является реальным временем, необходимым процессору для обновления выходов. Оно не включает какое-либо дополнительное время, которое необходимо процессору для реального обслуживания конкретных специальных модулей.

Диагностика

Процессоры DL450 имеют много типов систем диагностики. Требуемое для них время зависит от многих факторов, например от числа установленных модулей ввода/вывода и др. В следующей таблице показаны ожидаемое минимальное и максимальное время.

Время диагностирования	DL430	DL440	DL450
Минимальное	680.0 мкс	540.0 мкс	282.0 мкс
Максимальное	880.0 мкс	920.0 мкс	398.0 мкс

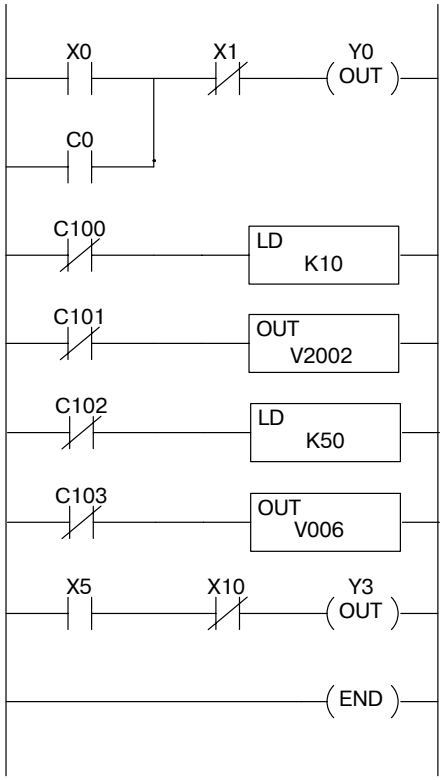
Выполнение прикладной программы

Процессор обрабатывает программу сверху (с адреса 0) до команды END. Процессор выполняет программу слева направо и сверху вниз. На каждой ступени вычисляется соответствующий регистр отображения или обновляется ячейка памяти.

Время, необходимое для выполнения прикладной программы, зависит от типа и числа используемых команд и от служебных затрат времени.

Вы можете сложить времена выполнения всех команд в вашей программе, чтобы найти общее время выполнения программы.

Например, время выполнения на DL430 программы, представленной на рисунке, рассчитывается следующим образом.



Команда		Время
STR X0		4.7 мкс
OR C0		3.1 мкс
ANDN X1		3.4 мкс
OUT Y0		8.3 мкс
STRN C100	5.7 мкс	
LD K10		112.0 мкс
STRN C101	8.3 мкс	
OUT V2002	26.0 мкс	
STRN C102	8.3 мкс	
LD K50		112.0 мкс
STRN C103	8.3 мкс	
OUT V0006	26.0 мкс	
STR X5		4.7 мкс
ANDN X10		3.4 мкс
OUT Y3		8.3 мкс
END		15.5 мкс
СУММА		358 мкс

В приложении C приведен полный перечень времен выполнения команд для процессоров DL405.

Команды Управления Программой: Процессоры DL440 и DL450 предлагают дополнительные команды, которые могут изменить ход выполнения программы. В эти команды входят циклы FOR/NEXT, подпрограммы, программы прерывания. Эти команды могут прерывать нормальный процесс выполнения программы и влиять на время выполнения программы. В главе 5 приводится детальная информация о том, как работают команды этого типа.

Системы счисления в ПЛК

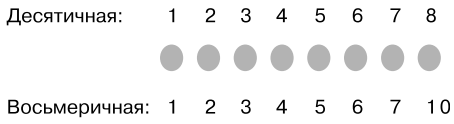
Если вы — новый пользователь ПЛК или вы впервые применяете PLCDirect в ваших ПЛК, выделите время на изучение того, как ПЛК работает с числами. Вы найдете, что каждый изготовитель ПЛК имеет собственное соглашение об использовании чисел в ПЛК. Выделите время, чтобы познакомиться с тем, как используются числа в ПЛК PLCDirect. Эта информация относится ко всем нашим ПЛК!

Как и каждый хороший компьютер, ПЛК хранит числа и манипулирует ими в двоичном виде: ноль и один. Так почему же мы имеем дело со столь многими различными представлениями чисел? Для конкретных целей некоторые представления чисел более удобны, чем другие. Иногда мы используем числа для представления размера или количества чего-либо. Другие числа относятся к ячейкам или адресам, или ко времени. При описании технических устройств числа имеют конкретный смысл.

Ресурсы ПЛК

В зависимости от модели и конфигурации ПЛК предлагают фиксированное количество ресурсов. Под словом ”ресурс” понимается память переменных (V-память), точки ввода/вывода, таймеры, счетчики и др. Большинство модульных ПЛК позволяют вам добавлять точки ввода/вывода группами из 8 точек. В действительности все ресурсы наших ПЛК считаются в восьмеричной форме. Для компьютера легче считать группы из восьми штук, чем из десяти, так как восемь есть степень 2.

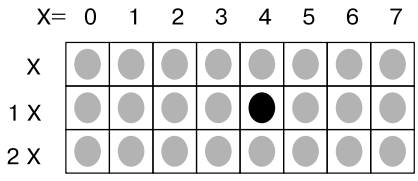
Восьмеричная система означает просто счет по группам из восьми предметов. На рисунке справа восемь кружков. Их число в десятичной системе — ”8”, а в восьмеричной — ”10” (8 и 9 отсутствуют в восьмеричной системе). В восьмеричной системе это означает 1 группу из 8 плюс 0 (нет единичных).



На рисунке ниже приведены две группы по восемь кружков. В восьмеричной системе мы имеем ”20” предметов, что означает 2 группы по восемь плюс 0 единичных. Нельзя говорить ”двадцать”, надо говорить ”два - ноль восьмеричных”. В этом четкое различие между системами счисления.

После определения того, как считаются ресурсы, следует перейти к тому, как осуществляется доступ ПЛК к ресурсам (это не одно и то же). Команды процессора при доступе к ресурсам ПЛК используют восьмеричные адреса. Восьмеричные адреса тоже имеют восьмеричное значение, за исключением того, что счет начинается с ноля. Число ноль для компьютера важно, поэтому его нельзя пропустить.

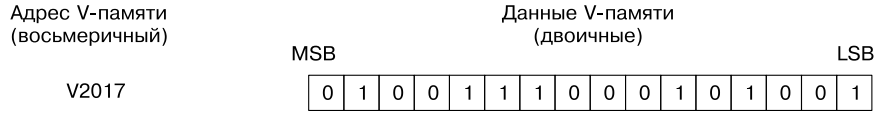
Указанные кружочки можно размещать в квадратах решетки, показанной справа. Для доступа к ресурсу команда вашего ПЛК может обращаться к ячейке, используя показанную восьмеричную ссылку. Если ресурсом являются счетчики, то ”CT14” дает доступ к ячейке, показанной в виде черного кружочка.



V-память

В памяти переменных (называемой "V-памятью") хранятся данные для программы релейной логики и для параметров настройки конфигурации. Ячейки V-памяти и адреса V-памяти — это одно и то же, они нумеруются в восьмеричной системе. Например, V2073 правильно указанная ячейка, V1983 не правильно (цифр 8 и 9 нет в восьмеричной системе).

Каждая ячейка V-памяти является одним словом, содержащим 16 бит. Для регистров конфигурации в нашем руководстве указывается каждый бит слова V-памяти. Наименее значимый бит (LSB) находится справа, наиболее значимый бит (MSB) находится слева. Слово "значимость" относится к относительным двоичным весам битов.

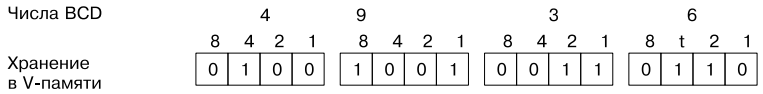


Данные в V-памяти — это 16-битовое двоичное число, но мы редко программируем регистры данных по битам. Мы используем команды и средства визуального отображения, которые позволяют нам работать с двоичными, десятичными, восьмеричными и шестнадцатеричными числами. Все они преобразуются и хранятся в двоичном виде.

Не имеет значения, как называть двоичные, восьмеричные и др. числа. Важно, чтобы источник или механизм, который записывает данные в ячейки V-памяти, и механизм, который затем их считывает, использовали один и тот же тип данных (то есть восьмеричный, двоичный, шестнадцатеричный или любой другой). Ячейка V-памяти есть просто место для хранения. Она сама ни преобразует, ни переносит данные.

Двоично-десятичные числа

Поскольку люди считают в десятичной системе, ввод и представление данных ПЛК также делает в десятичной форме (через интерфейсы операторов). Но компьютеры более эффективны при использовании строго двоичных чисел. Компромиссом между этими двумя системами является Двоично-Десятичное (BCD) представление чисел. Цифры BCD изменяются от 0 до 9 и хранятся как четыре двоичных бита (полубайт). Это позволяет хранить в каждой ячейке V памяти четыре цифры BCD с диапазоном десятичных чисел от 0000 до 9999.



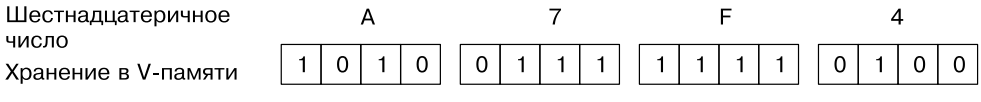
В чисто двоичном виде 16-битовое слово представляет числа от 0 до 65535. При хранении чисел в BCD диапазон сокращается от 0 до 9999. Многие математические команды используют данные в BCD, а DirectSOFT и ручной программатор позволяют вводить и просматривать данные в BCD. Специальные команды RLL преобразуют числа из BCD в двоичные и наоборот.

Шестнадцатеричные числа

Шестнадцатеричные числа аналогичны BCD числам, за исключением того, что они используют все возможные двоичные значения каждой 4-битовой цифры. Это — 16-ричное счисление, поэтому требуется 16 цифр. Для дополнения десятичных цифр от 0 до 9 используются буквы от A до F, как показано ниже.

Десятичные	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Шестнадцатеричные	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Шестнадцатеричное число из 4 цифр может представить все 65536 значений в слове V-памяти. Диапазон шестнадцатеричных чисел: от 0000 до FFFF. В ПЛК часто требуется этот полный диапазон для показаний чувствительных элементов и др. Шестнадцатеричная система является удобным для человека способом просмотреть двоичные данные.

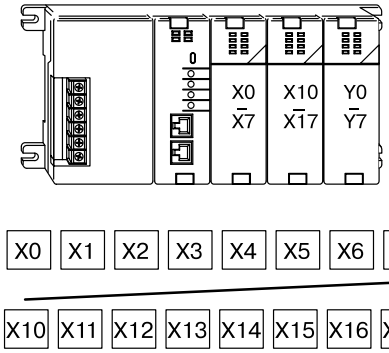


Карта распределения памяти

В системах ПЛК, как правило, обрабатываются многие типы информации. Сюда входят: входное и выходное состояния устройства, различные временные элементы, номера элементов и др. Важно понимать, как система представляет и хранит различные типы данных. Вам может понадобиться информация о том, как система идентифицирует входные и выходные точки, слова данных и др. В последующих параграфах рассматриваются различные типы памяти, используемые в процессорах DL405. После описания памяти следует краткое описание карт распределения памяти в процессорах DL430, DL440 и DL450.

Восьмеричная система нумерации

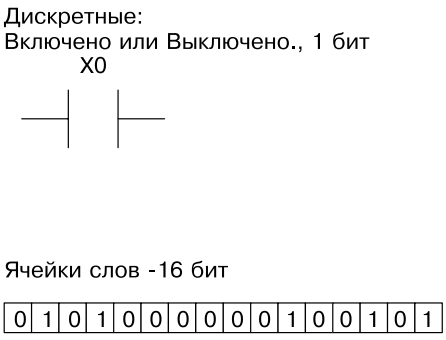
Все ячейки или области памяти нумеруются восьмеричными числами (с основанием 8). Например, на схеме справа показано, как действует восьмеричная система нумерации в случае дискретных входных точек. Заметим, что восьмеричная система не содержит номеров с цифрами 8 или 9.



Обратитесь к предыдущему разделу по восьмеричной системе нумерации.

Ячейки дискретных значений и ячейки слов

При изучении различных типов памяти вы обратили внимание на два типа памяти в DL405 — память дискретных значений и память слов. Память дискретных значений — это один бит, который может принимать значение 1 или 0. Пословная память рассматривается как V-память (переменных), представляет собой 16-битовую ячейку, обычно используемую для манипулирования с данными/числами и др и для их хранения.



Некоторая информация автоматически сохраняется в V-памяти. Например, текущие значения таймера запоминается в V-памяти.

Ячейки V-памяти для областей дискретной памяти

Область памяти дискретных значений предназначена для входов, выходов, управляющих реле, специальных реле, стадий, битов состояния таймеров и битов состояния счетчиков. Однако вы также имеете доступ к битовым данным в слове V-памяти. Каждая ячейка V-памяти состоит из 16 последовательных ячеек дискретных значений. Например, на следующей схеме показано, как входные точки X отображаются в ячейки V-памяти.

Эти области дискретной памяти и соответствующие ячейки V-памяти указаны в в данной главе в таблицах Распределения памяти для процессоров DL430, DL440 и DL450.

Входные точки (Тип данных X)

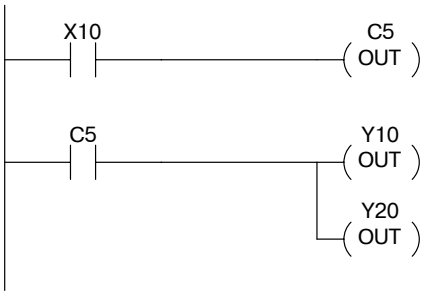
Дискретные входные точки помечаются как тип данных X. По числу дискретных входных точек обратитесь к картам памяти по конкретному типу вашего процессора. В данном примере выходная точка Y0 будет включена при включении входа X0.

Выходные точки (Тип данных Y)

Дискретные выходные точки помечаются как тип данных Y. По числу дискретных выходных точек обратитесь к картам памяти по конкретному типу вашего процессора. В данном примере выходная точка Y1 будет включена при включении входа X1.

Управляющие реле (Тип данных C)

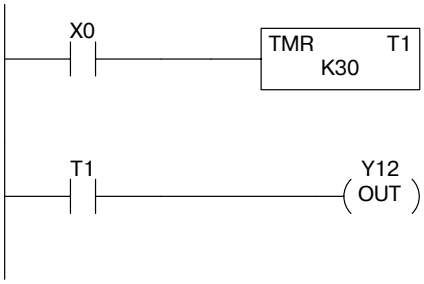
Управляющие реле представляются дискретными битами, обычно используемыми для управления пользовательской программой. Управляющие реле не представляют собой реальное известное устройство, то есть они не могут быть физически связаны с переключателями, выходными обмотками и др. Они являются внутренними для процессора. Управляющие реле могут программироваться как дискретные входы или дискретные выходы. Их ячейки используются при программировании как ячейки памяти дискретных значений (C) или как соответствующая им ячейка слова, которая имеет 16 последовательных ячеек дискретных значений.



В данном примере ячейка памяти C5 будет включена при включении входа X10. Вторая ступенька рисунка дает простой пример того, как можно использовать управляющее реле в качестве входа.

Биты таймеров и состояния таймеров (Тип данных T)

Число таймеров зависит от модели используемого вами процессора. В таблице в конце данного раздела приводится число таймеров в DL430, DL440 и DL450. Сколько бы вы не имели таймеров, вы имеете доступ к битам их состояния, которые отражают отношение между текущим и установленным значением определенного таймера. Бит состояния будет "включен", когда текущее значение станет равным или большим установленного значения соответствующего таймера.



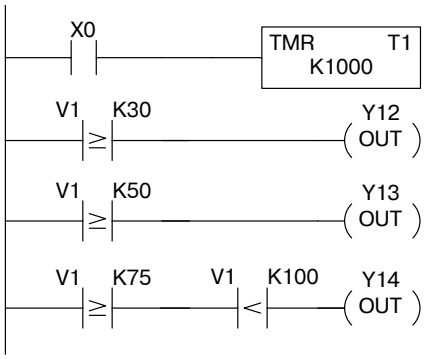
Когда вход X0 включается, запускается таймер T1. Когда таймер достигнет установленного значения 3 сек. (K из 30), контакт состояния таймера T1 включится. Когда T1 включается, выход Y12 также включается.

ПРИМЕЧАНИЕ. Некоторые таймеры используют один регистр V-памяти, другие типы таймеров требуют двух регистров V-памяти. См описание команд в главе 5.

Текущие значения таймера (Тип данных V)

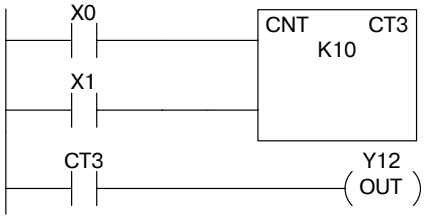
Некоторая информация автоматически запоминается в V-памяти. Это относится также к текущим значениям таймеров. Например, V0 поддерживает текущее значение таймера 0, V1 поддерживает текущее значение таймера 1 и т. д. Формат — 4 цифры BCD.

Основная причина, почему так делается, — гибкость программирования. Приведенный пример показывает, как вы можете использовать связанные контакты для отслеживания нескольких временных интервалов при помощи одного таймера.



Биты счетчиков и состояния счетчиков (Тип данных CT)

Вы имеете доступ к битам состояния счетчиков, которые отражают отношение между текущим и установленным значением определенного счетчика. Бит состояния счетчика будет "включен", когда текущее его значение станет равным или большим установленного значения соответствующего счетчика.

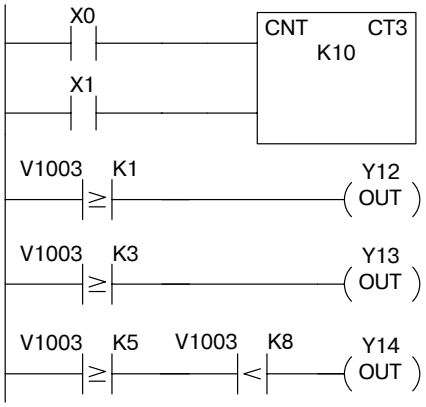


Всякий раз, когда контакт X0 переходит из состояния "выключен" в состояния "включен", счетчик увеличивается на единицу. (Если X1 включается, то счетчик возвращается в состояние "ноль"). Когда счетчик достигает заданных 10 подсчетов (K из 10), то контакт состояния счетчика CT3 "включается". Когда "включается" CT3, включается и выход Y12.

Текущие значения счетчика (Тип данных V)

Текущие значения счетчиков, так же как и таймеров, автоматически запоминается в V-памяти. Например, V1000 поддерживает текущее значение счетчика CT0, V1001 поддерживает текущее значение счетчика CT1 и т. д. Текущие значения представляют собой четырехзначные двоично-десятичные числа.

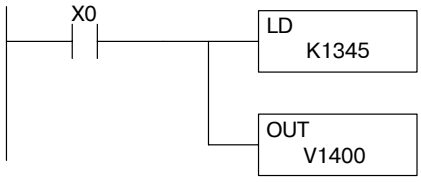
Основная причина, почему так делается, — гибкость программирования. Приведенный пример показывает, как вы можете использовать связанные контакты для отслеживания значений счетчика.



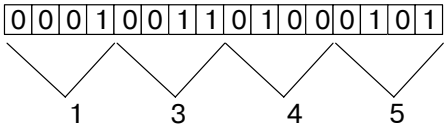
Память слов (Тип данных V)

Память слов относится к V-памяти (переменных), представляет собой 16-битовую ячейку, обычно используемую для манипулирования данными/числами и др. и для их хранения Некоторая информация автоматически сохраняется в V-памяти. Например, текущие значения таймера запоминается в V-памяти.

На приведенном примере показано, как четырехзначная BCD константа загружается в аккумулятор и затем запоминается в ячейке V-памяти.

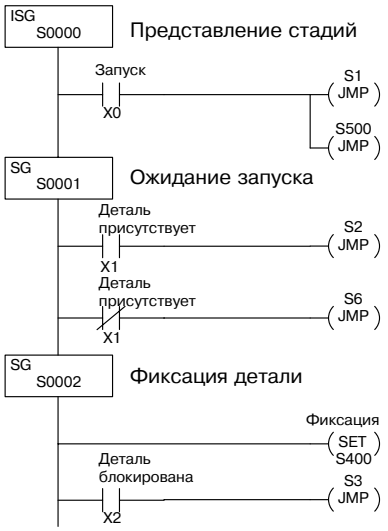


Ячейки слов — 16 битов



Стадии (Тип данных S)

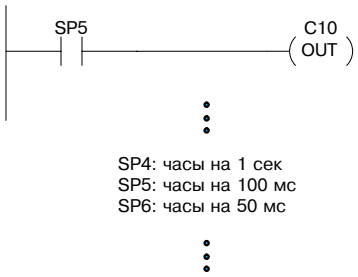
Стадии используются в программах RLLplus для создания структурированных программ, подобным блок-схемам. Каждая стадия программа обозначает программный сегмент. Когда программный сегмент, или стадия, активен, то выполняется логика данного сегмента. Если стадия "отключена", или не активна, логика сегмента не выполняется, а процессор переходит к следующей активной стадии. (В главе 7 дается более детальное описание программирование на RLLplus).



Каждый сегмент стадии также имеет дискретный бит состояния, который можно использовать как вход, указывающий, активна ли стадия или нет. Если стадия активна, то бит ее состояния "включен". Если стадия не активна, то бит ее состояния "выключен". Этот бит состояния может "включаться" и "выключаться" также другими командами, такими как команды SET или RESET. Это дает вам возможность легко управлять стадиями программы.

Специальные реле (Тип данных SP)

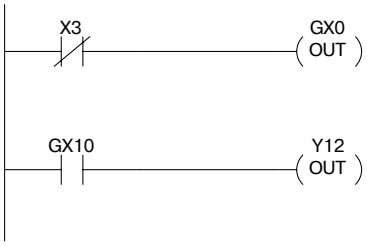
Специальные реле представляют собой ячейки дискретной памяти с заранее определенным набором функций. Существует много различных типов специальных реле. Например, некоторые из них оказывают помощь при разработке программ. Другие содержат информацию о рабочем состоянии системы и т. д. В приложении D приводится полный перечень специальных реле.



В примере на рисунке управляющее реле C10 будет включаться на 50 мсек. и отключаться на 50 мсек., так как SP5 является реле с предварительно заданным значением, оно находится в состоянии "включено" в течение 50 мсек. и в состоянии "выключено" также в течение 50 мсек.

Точки удаленного ввода/вывода (Тип данных GX)

Точки удаленного ввода/вывода представляются с помощью общих реле. Они в основном используются только для управления удаленным вводом/выводом, но они могут также использоваться и как обычные управляющие реле, если в системе имеется удаленный ввод/вывод. Обратите внимание, что один и тот же тип данных,GX, используется и для удаленного ввода и для удаленного вывода. Поэтому в вашей прикладной программе должна быть подпрограмма, которая определяет, какие ячейки являются входными, а какие — выходными. (В Руководстве по удаленному и секционированному вводу/выводу приведена более подробная информация по этому вопросу).



В данном примере ячейка памяти GX0 представляет выходную точку, а ячейка памяти GX10 — выходную точку.

Системные параметры (тип данных V)

Многие системные параметры, например, коды ошибок, автоматически сохраняются в заданных ячейках V-памяти. В этих ячейках запоминается информация часов/календаря, коды ошибок и другие данные по настройке системы.

X	√	√
430	440	450

V-память системы	Описание содержимого
V700, V701	Содержит копию содержимого аккумулятора. V700 - младшее 16-битовое слово, V701 - старшее 16-битовое слово.
V702, V703	Содержит копию содержимого 1 ячейки стека данных. V702 - младшее 16-битовое слово, V703 - старшее 16-битовое слово.
V704, V705	Содержит копию содержимого 2 ячейки стека данных. V704 - младшее 16-битовое слово, V705 - старшее 16-битовое слово.
V706, V707	Содержит копию содержимого 3 ячейки стека данных. V706 - младшее 16-битовое слово, V707 - старшее 16-битовое слово.
V710, V71 1	Содержит копию содержимого 4 ячейки стека данных. V710 - младшее 16-битовое слово, V711 - старшее 16-битовое слово.
V712, V713	Содержит копию содержимого 5 ячейки стека данных. V712 - младшее 16-битовое слово, V713 - старшее 16-битовое слово.
V714, V715	Содержит копию содержимого 6 ячейки стека данных. V714 - младшее 16-битовое слово, V715 - старшее 16-битовое слово.
V716, V717	Содержит копию содержимого 7 ячейки стека данных. V716 - младшее 16-битовое слово, V717 - старшее 16-битовое слово.
V720, V721	Содержит копию содержимого 8 ячейки стека данных. V720 - младшее 16-битовое слово, V721 - старшее 16-битовое слово.
V737	Содержит двоично-десятичное значение (от 0 до 1000) для параметра с прерыванием по времени.
V7747	Содержит таймер календаря с отсчетом 10 мс, используемый в часах/календаре.
V7766	Содержит число секунд часов (от 00 до 59).
V7767	Содержит число минут часов (от 00 до 59).
V7770	Содержит число часов (от 00 до23).
V7771	Содержит день недели (0 - воскресенье, 1 - понедельник и т. д.).
V7772	Содержит день месяца (1, 2 и т. д.).
V7773	Содержит месяц (от 01 до12).
V7774	Содержит год (от 00 до 99).

√	√	√
430	440	450

V-память системы	Описание содержимого
V7746	430/440: Ошибка коммуникаций. Процессор автоматически сохраняет номера каркаса и слота модуля с ошибкой связи. 450: Напряжение батареи с точностью до десятых долей вольта.
V7747	430/440: Код ошибки коммуникаций. 450: Резервная ячейка.
V7751	Код ошибки ошибочного сообщения. Хранится четырехзначный двоично-десятичный код, используемый командой FAULT при ее выполнении. Если используются сообщения с кодом ASCII (только в DL440/DL450), то номер ссылки на метку данных (DLBL) для этого сообщения также хранится в этой ячейке.

√	√	√
430	440	450

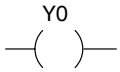
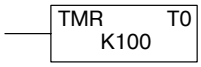
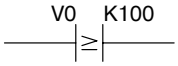
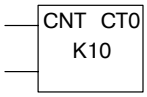
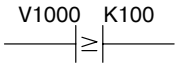
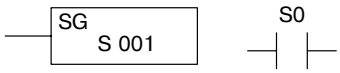
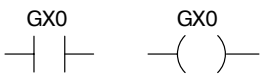
V-память системы	**Описание содержимого**
V7752	Ошибка в конфигурации ввода/вывода. Хранится код идентификатора модуля, который не согласован с действующей конфигурацией.
V7753	Ошибка в конфигурации ввода/вывода. Хранится правильный код идентификатора модуля.
V7754	Ошибка в конфигурации ввода/вывода. Задается номер каркаса и номер слота.
V7755	Код ошибки. Хранится код неисправимой ошибки.
V7756	Код ошибки. Хранится код старшей ошибки.
V7757	430/440: Код ошибки. Хранится код младшей ошибки. 450: Код ошибки коммуникаций
V7760	Ошибка модуля. Идентифицируются номера каркаса и слота.
V7762	Ошибка модуля. Идентифицируется тип ошибки.
V7763	Грамматическая ошибка в программе. Идентифицируется ячейка с синтаксической ошибкой в программе.
V7764	Грамматическая ошибка в программе. Идентифицируется тип ошибки.
V7765	Сканирование. Хранится общее число циклов после последнего перехода из программного режима в рабочий.
V7775	Сканирование. Хранится текущее время сканирования.
V7776	Сканирование. Хранится минимальное время сканирования, которое имело место после последнего перехода из программного режима в рабочий.
V7777	Сканирование. Хранится максимальное время сканирования, которое имело место после последнего перехода из программного режима в рабочий.

Следующие системные управляющие реле применимы только для настройки удаленного ввода/вывода процессора DL450 с коммуникационного порта 3.

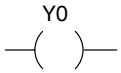
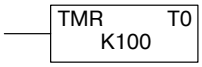
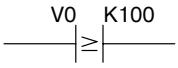
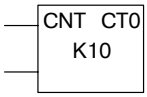
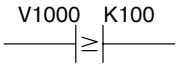
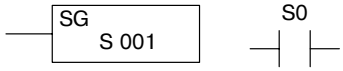
X	X	√
430	440	450

Системные управляющие реле	**Описание содержимого**
C740	Завершение настройки. Релейная логика должна замкнуть это реле, когда закончена запись в таблицу настроек удаленного ввода/вывода.
C741	Стирание полученных данных. При установлении этого флага будут стерты при ошибке связи полученные данные.
C743	Повторный запуск. Включение этого реле возобновит работу после ”зависания” при ошибке связи.
с C750 до C757	Ошибка настройки. Соответствующее реле будет ”включено”, если таблица настроек содержит ошибку (C750 = ведущее устройство, C751 = ведомое 1,... ..C757 = ведомое 7)
с C760 до C767	Готовность коммуникаций. Соответствующее реле будет ”включено”, если данные таблицы настроек правильны (C760 = ведущее устройство, C761 = ведомое 1,... ..C767 = ведомое 7)

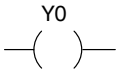
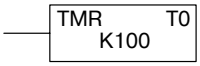
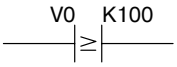
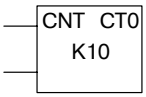
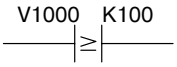
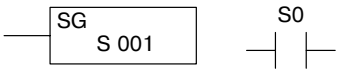
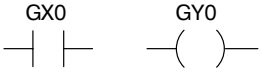
Карта распределения памяти DL430

Тип памяти	Указатель (восьмеричный) дискретной памяти	Указатель (восьмеричный) пословной памяти	Кол-во десяти-чное	Обозначение
Входные точки	X0 - X477	V40400 - V40423	320	
Выходные точки	Y0 - Y477	V40500 - V40523	320	
Управляющие реле	C0 - C737	V40600 - V40635	512	
Специальные реле	SP0 - SP137 SP320 - SP617	V41200 - V41205 V41215 - V41230	288	
Таймеры	T0 - T177	Нет	128	
Текущие значения таймера	Нет	V00000 - V00177	128	
Биты состояния таймера	T0 - T177	V41100 - V41107	128	
Счетчики	CT0 - CT177	Нет	128	
Текущие значения счетчика	Нет	V01000 - V01177	128	
Биты состояния счетчика	CT0 - CT177	V41140 - V41147	128	
Слова данных пользователя	Нет	V1400 - V7377	3072	Не специфицируются, используются о многими командами
Стадии	S0 - S577	V41000 - V41027	384	
Удаленный ввод/вывод	GX0 - GX777	V40000 - V40037	512	
Системные параметры	Нет	V7400 - V7777	256	Не специфицируются, используются о многими командами

Карта распределения памяти DL440

Тип памяти	Указатель (восьмеричный) дискретной памяти	Указатель (восьмеричный) пословной памяти	Кол-во десяти-чное	Обозначение
Входные точки	X0 - X477	V40400 - V40423	320	
Выходные точки	Y0 - Y477	V40500-V40523	320	
Управляющие реле	C0 - C1777	V40600-V40677	1024	
Специальные реле	SP0 - SP137 SP320 - SP717	V41200-V41205 V41215-V41234	352	
Таймеры	T0 - T377	Нет	256	
Текущие значения таймеров	Нет	V00000 - V00377	256	
Биты состояния таймеров	T0 - T377	V41100 - V41117	256	
Счетчики	CT0 - CT177	Нет	128	
Текущие значения счетчиков	Нет	V01000 - V01177	128	
Биты состояния счетчиков	CT0 - CT177	V41140 - V41147	128	
Слова данных пользователя	Нет	V1400 - V7377 V10000 - V17777	3072 4096	Не специфицируются, используются со многими командами
Стадии	S0 - S1777	V41000 - V41077	1024	
Удаленный ввод/вывод	GX0 - GX1777	V40000 - V40077	1024	
Системные параметры	Нет	V700 - V737 V7400 - V7777	288	Не специфицируются, используются со многими командами

Карта распределения памяти DL450

Тип памяти	Указатель (восьмеричный) дискретной памяти	Указатель (восьмеричный) пословной памяти	Кол-во десяти-чное	Обозначение
Входные точки	X0 - X1777	V40400 - V40477	1024	
Выходные точки	Y0 - Y1777	V40500-V40577	1024	
Управляющие реле	C0 - C3777	V40600-V40777	2048	
Специальные реле	SP0 - SP137 SP320 - SP717	V41200-V41237	512	
Таймеры	T0 - T377	Нет	256	
Текущие значения таймеров	Нет	V00000 - V00377	256	
Биты состояния таймеров	T0 - T377	V41100 - V41117	256	
Счетчики	CT0 - CT377	Нет	256	
Текущие значения счетчиков	Нет	V01000 - V01377	256	
Биты состояния счетчиков	CT0 - CT377	V41140 - V41157	256	
Слова данных пользователя	Нет	V1400 - V7377 V10000 - V37777	3072 12288	Не специфицируются, используются со многими командами
Стадии	S0 - S1777	V41000 - V41077	1024	
Удаленный ввод/вывод	GX0 - GX2777 GY0 - GY2777	V40000 - V40137 V40200 - V40337	1536 combined GX + GY	
Системные параметры	Нет	V700 - V777 V7400 - V7777	320	Не специфицируются, используются со многими командами

Карта битового отображения Входа X / Выхода Y

В данной таблице приводится список отдельных входных и выходных точек, связанных с каждым битом адреса V-памяти для процессоров DL430, DL440 и DL450.

MSB								Точки входов (X) и выходов (Y) в DL430, DL440 и DL450							LSB	Адрес входа X	Адрес выхода Y
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0		
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000	V40400	V40500
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	V40401	V40501
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	V40402	V40502
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	V40403	V40503
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	V40404	V40504
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	V40405	V40505
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	V40406	V40506
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	V40407	V40507
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200	V40410	V40510
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220	V40411	V40511
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240	V40412	V40512
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260	V40413	V40513
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300	V40414	V40514
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320	V40415	V40515
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340	V40416	V40516
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360	V40417	V40517
417	416	415	414	413	412	411	410	407	406	405	404	403	402	401	400	V40420	V40520
437	436	435	434	433	432	431	430	427	426	425	424	423	422	421	420	V40421	V40521
457	456	455	454	453	452	451	450	447	446	445	444	443	442	441	440	V40422	V40522
477	476	475	474	473	472	471	470	467	466	465	464	463	462	461	460	V40423	V40523

MSB								Дополнительные точки входов (X) и выходов (Y) в DL450							LSB	Адрес входа X	Адрес выхода Y
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0		
517	516	515	514	513	512	511	510	507	506	505	504	503	502	501	500	V40424	V40524
537	536	535	534	533	532	531	530	527	526	525	524	523	522	521	520	V40425	V40525
557	556	555	554	553	552	551	550	547	546	545	544	543	542	541	540	V40426	V40526
577	576	575	574	573	572	571	570	567	566	565	564	563	562	561	560	V40427	V40527
617	616	615	614	613	612	611	610	607	606	605	604	603	602	601	600	V40430	V40530
637	636	635	634	633	632	631	630	627	626	625	624	623	622	621	620	V40431	V40531
657	656	655	654	653	652	651	650	647	646	645	644	643	642	641	640	V40432	V40532
677	676	675	674	673	672	671	670	667	666	665	664	663	662	661	660	V40433	V40533
717	716	715	714	713	712	711	710	707	706	705	704	703	702	701	700	V40434	V40534
737	736	735	734	733	732	731	730	727	726	725	724	723	722	721	720	V40435	V40535
757	756	755	754	753	752	751	750	747	746	745	744	743	742	741	740	V40436	V40536
777	776	775	774	773	772	771	770	767	766	765	764	763	762	761	760	V40437	V40537

MSB								**Дополнительные точки входов (X) и выходов (Y) в DL450**							**LSB**	**Адрес входа X**	**Адрес выхода Y**
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0		
1017	1016	1015	1014	1013	1012	1011	1010	1007	1006	1005	1004	1003	1002	1001	1000	V40440	V40540
1037	1036	1035	1034	1033	1032	1031	1030	1027	1026	1025	1024	1023	1022	1021	1020	V40441	V40541
1057	1056	1055	1054	1053	1052	1051	1050	1047	1046	1045	1044	1043	1042	1041	1040	V40442	V40542
1077	1076	1075	1074	1073	1072	1071	1070	1067	1066	1065	1064	1063	1062	1061	1060	V40443	V40543
1117	1116	1115	1114	1113	1112	1111	1110	1107	1106	1105	1104	1103	1102	1101	1100	V40444	V40544
1137	1136	1135	1134	1133	1132	1131	1130	1127	1126	1125	1124	1123	1122	1121	1120	V40445	V40545
1157	1156	1155	1154	1153	1152	1151	1150	1147	1146	1145	1144	1143	1142	1141	1140	V40446	V40546
1177	1176	1175	1174	1173	1172	1171	1170	1167	1166	1165	1164	1163	1162	1161	1160	V40447	V40547
1217	1216	1215	1214	1213	1212	1211	1210	1207	1206	1205	1204	1203	1202	1201	1200	V40450	V40550
1237	1236	1235	1234	1233	1232	1231	1230	1227	1226	1225	1224	1223	1222	1221	1220	V40451	V40551
1257	1256	1255	1254	1253	1252	1251	1250	1247	1246	1245	1244	1243	1242	1241	1240	V40452	V40552
1277	1276	1275	1274	1273	1272	1271	1270	1267	1266	1265	1264	1263	1262	1261	1260	V40453	V40553
1317	1316	1315	1314	1313	1312	1311	1310	1307	1306	1305	1304	1303	1302	1301	1300	V40454	V40554
1337	1336	1335	1334	1333	1332	1331	1330	1327	1326	1325	1324	1323	1322	1321	1320	V40455	V40555
1357	1356	1355	1354	1353	1352	1351	1350	1347	1346	1345	1344	1343	1342	1341	1340	V40456	V40556
1377	1376	1375	1374	1373	1372	1371	1370	1367	1366	1365	1364	1363	1362	1361	1360	V40457	V40557
1417	1416	1415	1414	1413	1412	1411	1410	1407	1406	1405	1404	1403	1402	1401	1400	V40460	V40560
1437	1436	1435	1434	1433	1432	1431	1430	1427	1426	1425	1424	1423	1422	1421	1420	V40461	V40561
1457	1456	1455	1454	1453	1452	1451	1450	1447	1446	1445	1444	1443	1442	1441	1440	V40462	V40562
1477	1476	1475	1474	1473	1472	1471	1470	1467	1466	1465	1464	1463	1462	1461	1460	V40463	V40563
1517	1516	1515	1514	1513	1512	1511	1510	1507	1506	1505	1504	1503	1502	1501	1500	V40464	V40564
1537	1536	1535	1534	1533	1532	1531	1530	1527	1526	1525	1524	1523	1522	1521	1520	V40465	V40565
1557	1556	1555	1554	1553	1552	1551	1550	1547	1546	1545	1544	1543	1542	1541	1540	V40466	V40566
1577	1576	1575	1574	1573	1572	1571	1570	1567	1566	1565	1564	1563	1562	1561	1560	V40467	V40567
1617	1616	1615	1614	1613	1612	1611	1610	1607	1606	1605	1604	1603	1602	1601	1600	V40470	V40570
1637	1636	1635	1634	1633	1632	1631	1630	1627	1626	1625	1624	1623	1622	1621	1620	V40471	V40571
1657	1656	1655	1654	1653	1652	1651	1650	1647	1646	1645	1644	1643	1642	1641	1640	V40472	V40572
1677	1676	1675	1674	1673	1672	1671	1670	1667	1666	1665	1664	1663	1662	1661	1660	V40473	V40573
1717	1716	1715	1714	1713	1712	1711	1710	1707	1706	1705	1704	1703	1702	1701	1700	V40474	V40574
1737	1736	1735	1734	1733	1732	1731	1730	1727	1726	1725	1724	1723	1722	1721	1720	V40475	V40575
1757	1756	1755	1754	1753	1752	1751	1750	1747	1746	1745	1744	1743	1742	1741	1740	V40476	V40576
1777	1776	1775	1774	1773	1772	1771	1770	1767	1766	1765	1764	1763	1762	1761	1760	V40477	V40577

Карта битового отображения управляющих реле

В данной таблице приводится список отдельных управляющих реле, связанных с каждым битом адреса V-памяти.

MSB								**Управляющие Реле (C) в DL430, DL440 и DL450**							**LSB**	**Адрес**
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0	
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000	V40600
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	V40601
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	V40602
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	V40603
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	V40604
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	V40605
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	V40606
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	V40607
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200	V40610
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220	V40611
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240	V40612
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260	V40613
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300	V40614
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320	V40615
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340	V40616
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360	V40617
417	416	415	414	413	412	411	410	407	406	405	404	403	402	401	400	V40620
437	436	435	434	433	432	431	430	427	426	425	424	423	422	421	420	V40621
457	456	455	454	453	452	451	450	447	446	445	444	443	442	441	440	V40622
477	476	475	474	473	472	471	470	467	466	465	464	463	462	461	460	V40623
517	516	515	514	513	512	511	510	507	506	505	504	503	502	501	500	V40624
537	536	535	534	533	532	531	530	527	526	525	524	523	522	521	520	V40625
557	556	555	554	553	552	551	550	547	546	545	544	543	542	541	540	V40626
577	576	575	574	573	572	571	570	567	566	565	564	563	562	561	560	V40627
617	616	615	614	613	612	611	610	607	606	605	604	603	602	601	600	V40630
637	636	635	634	633	632	631	630	627	626	625	624	623	622	621	620	V40631
657	656	655	654	653	652	651	650	647	646	645	644	643	642	641	640	V40632
677	676	675	674	673	672	671	670	667	666	665	664	663	662	661	660	V40633
717	716	715	714	713	712	711	710	707	706	705	704	703	702	701	700	V40634
737	736	735	734	733	732	731	730	727	726	725	724	723	722	721	720	V40635

MSB								**Дополнительные управляющие реле (C) в DL440 и DL450**							**LSB**	**Адрес**
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0	
757	756	755	754	753	752	751	750	747	746	745	744	743	742	741	740	V40636
777	776	775	774	773	772	771	770	767	766	765	764	763	762	761	760	V40637
1017	1016	1015	1014	1013	1012	1011	1010	1007	1006	1005	1004	1003	1002	1001	1000	V40640
1037	1036	1035	1034	1033	1032	1031	1030	1027	1026	1025	1024	1023	1022	1021	1020	V40641
1057	1056	1055	1054	1053	1052	1051	1050	1047	1046	1045	1044	1043	1042	1041	1040	V40642
1077	1076	1075	1074	1073	1072	1071	1070	1067	1066	1065	1064	1063	1062	1061	1060	V40643
1117	1116	1115	1114	1113	1112	1111	1110	1107	1106	1105	1104	1103	1102	1101	1100	V40644
1137	1136	1135	1134	1133	1132	1131	1130	1127	1126	1125	1124	1123	1122	1121	1120	V40645
1157	1156	1155	1154	1153	1152	1151	1150	1147	1146	1145	1144	1143	1142	1141	1140	V40646
1177	1176	1175	1174	1173	1172	1171	1170	1167	1166	1165	1164	1163	1162	1161	1160	V40647
1217	1216	1215	1214	1213	1212	1211	1210	1207	1206	1205	1204	1203	1202	1201	1200	V40650
1237	1236	1235	1234	1233	1232	1231	1230	1227	1226	1225	1224	1223	1222	1221	1220	V40651
1257	1256	1255	1254	1253	1252	1251	1250	1247	1246	1245	1244	1243	1242	1241	1240	V40652
1277	1276	1275	1274	1273	1272	1271	1270	1267	1266	1265	1264	1263	1262	1261	1260	V40653
1317	1316	1315	1314	1313	1312	1311	1310	1307	1306	1305	1304	1303	1302	1301	1300	V40654
1337	1336	1335	1334	1333	1332	1331	1330	1327	1326	1325	1324	1323	1322	1321	1320	V40655
1357	1356	1355	1354	1353	1352	1351	1350	1347	1346	1345	1344	1343	1342	1341	1340	V40656
1377	1376	1375	1374	1373	1372	1371	1370	1367	1366	1365	1364	1363	1362	1361	1360	V40657
1417	1416	1415	1414	1413	1412	1411	1410	1407	1406	1405	1404	1403	1402	1401	1400	V40660
1437	1436	1435	1434	1433	1432	1431	1430	1427	1426	1425	1424	1423	1422	1421	1420	V40661
1457	1456	1455	1454	1453	1452	1451	1450	1447	1446	1445	1444	1443	1442	1441	1440	V40662
1477	1476	1475	1474	1473	1472	1471	1470	1467	1466	1465	1464	1463	1462	1461	1460	V40663
1517	1516	1515	1514	1513	1512	1511	1510	1507	1506	1505	1504	1503	1502	1501	1500	V40664
1537	1536	1535	1534	1533	1532	1531	1530	1527	1526	1525	1524	1523	1522	1521	1520	V40665
1557	1556	1555	1554	1553	1552	1551	1550	1547	1546	1545	1544	1543	1542	1541	1540	V40666
1577	1576	1575	1574	1573	1572	1571	1570	1567	1566	1565	1564	1563	1562	1561	1560	V40667
1617	1616	1615	1614	1613	1612	1611	1610	1607	1606	1605	1604	1603	1602	1601	1600	V40670
1637	1636	1635	1634	1633	1632	1631	1630	1627	1626	1625	1624	1623	1622	1621	1620	V40671
1657	1656	1655	1654	1653	1652	1651	1650	1647	1646	1645	1644	1643	1642	1641	1640	V40672
1677	1676	1675	1674	1673	1672	1671	1670	1667	1666	1665	1664	1663	1662	1661	1660	V40673
1717	1716	1715	1714	1713	1712	1711	1710	1707	1706	1705	1704	1703	1702	1701	1700	V40674
1737	1736	1735	1734	1733	1732	1731	1730	1727	1726	1725	1724	1723	1722	1721	1720	V40675
1757	1756	1755	1754	1753	1752	1751	1750	1747	1746	1745	1744	1743	1742	1741	1740	V40676
1777	1776	1775	1774	1773	1772	1771	1770	1767	1766	1765	1764	1763	1762	1761	1760	V40677

MSB								**Дополнительные управляющие реле (C) в DL450**							**LSB**	**Адрес**
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0	
2017	2016	2015	2014	2013	2012	2011	2010	2007	2006	2005	2004	2003	2002	2001	2000	V40700
2037	2036	2035	2034	2033	2032	2031	2030	2027	2026	2025	2024	2023	2022	2021	2020	V40701
2057	2056	2055	2054	2053	2052	2051	2050	2047	2046	2045	2044	2043	2042	2041	2040	V40702
2077	2076	2075	2074	2073	2072	2071	2070	2067	2066	2065	2064	2063	2062	2061	2060	V40703
2117	2116	2115	2114	2113	2112	2111	2110	2107	2106	2105	2104	2103	2102	2101	2100	V40704
2137	2136	2135	2134	2133	2132	2131	2130	2127	2126	2125	2124	2123	2122	2121	2120	V40705
2157	2156	2155	2154	2153	2152	2151	2150	2147	2146	2145	2144	2143	2142	2141	2140	V40706
2177	2176	2175	2174	2173	2172	2171	2170	2167	2166	2165	2164	2163	2162	2161	2160	V40707
2217	2216	2215	2214	2213	2212	2211	2210	2207	2206	2205	2204	2203	2202	2201	2200	V40710
2237	2236	2235	2234	2233	2232	2231	2230	2227	2226	2225	2224	2223	2222	2221	2220	V40711
2257	2256	2255	2254	2253	2252	2251	2250	2247	2246	2245	2244	2243	2242	2241	2240	V40712
2277	2276	2275	2274	2273	2272	2271	2270	2267	2266	2265	2264	2263	2262	2261	2260	V40713
2317	2316	2315	2314	2313	2312	2311	2310	2307	2306	2305	2304	2303	2302	2301	2300	V40714
2337	2336	2335	2334	2333	2332	2331	2330	2327	2326	2325	2324	2323	2322	2321	2320	V40715
2357	2356	2355	2354	2353	2352	2351	2350	2347	2346	2345	2344	2343	2342	2341	2340	V40716
2377	2376	2375	2374	2373	2372	2371	2370	2367	2366	2365	2364	2363	2362	2361	2360	V40717
2417	2416	2415	2414	2413	2412	2411	2410	2407	2406	2405	2404	2403	2402	2401	2400	V40720
2437	2436	2435	2434	2433	2432	2431	2430	2427	2426	2425	2424	2423	2422	2421	2420	V40721
2457	2456	2455	2454	2453	2452	2451	2450	2447	2446	2445	2444	2443	2442	2441	2440	V40722
2477	2476	2475	2474	2473	2472	2471	2470	2467	2466	2465	2464	2463	2462	2461	2460	V40723
2517	2516	2515	2514	2513	2512	2511	2510	2507	2506	2505	2504	2503	2502	2501	2500	V40724
2537	2536	2535	2534	2533	2532	2531	2530	2527	2526	2525	2524	2523	2522	2521	2520	V40725
2557	2556	2555	2554	2553	2552	2551	2550	2547	2546	2545	2544	2543	2542	2541	2540	V40726
2577	2576	2575	2574	2573	2572	2571	2570	2567	2566	2565	2564	2563	2562	2561	2560	V40727
2617	2616	2615	2614	2613	2612	2611	2610	2607	2606	2605	2604	2603	2602	2601	2600	V40730
2637	2636	2635	2634	2633	2632	2631	2630	2627	2626	2625	2624	2623	2622	2621	2620	V40731
2657	2656	2655	2654	2653	2652	2651	2650	2647	2646	2645	2644	2643	2642	2641	2640	V40732
2677	2676	2675	2674	2673	2672	2671	2670	2667	2666	2665	2664	2663	2662	2661	2660	V40733
2717	2716	2715	2714	2713	2712	2711	2710	2707	2706	2705	2704	2703	2702	2701	2700	V40734
2737	2736	2735	2734	2733	2732	2731	2730	2727	2726	2725	2724	2723	2722	2721	2720	V40735
2757	2756	2755	2754	2753	2752	2751	2750	2747	2746	2745	2744	2743	2742	2741	2740	V40736
2777	2776	2775	2774	2773	2772	2771	2770	2767	2766	2765	2764	2763	2762	2761	2760	V40737

MSB								**Дополнительные управляющие реле (C) в DL450**							**LSB**	**Адрес**
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0	
3017	3016	3015	3014	3013	3012	3011	3010	3007	3006	3005	3004	3003	3002	3001	3000	V40740
3037	3036	3035	3034	3033	3032	3031	3030	3027	3026	3025	3024	3023	3022	3021	3020	V40741
3057	3056	3055	3054	3053	3052	3051	3050	3047	3046	3045	3044	3043	3042	3041	3040	V40742
3077	3076	3075	3074	3073	3072	3071	3070	3067	3066	3065	3064	3063	3062	3061	3060	V40743
3117	3116	3115	3114	3113	3112	3111	3110	3107	3106	3105	3104	3103	3102	3101	3100	V40744
3137	3136	3135	3134	3133	3132	3131	3130	3127	3126	3125	3124	3123	3122	3121	3120	V40745
3157	3156	3155	3154	3153	3152	3151	3150	3147	3146	3145	3144	3143	3142	3141	3140	V40746
3177	3176	3175	3174	3173	3172	3171	3170	3167	3166	3165	3164	3163	3162	3161	3160	V40747
3217	3216	3215	3214	3213	3212	3211	3210	3207	3206	3205	3204	3203	3202	3201	3200	V40750
3237	3236	3235	3234	3233	3232	3231	3230	3227	3226	3225	3224	3223	3222	3221	3220	V40751
3257	3256	3255	3254	3253	3252	3251	3250	3247	3246	3245	3244	3243	3242	3241	3240	V40752
3277	3276	3275	3274	3273	3272	3271	3270	3267	3266	3265	3264	3263	3262	3261	3260	V40753
3317	3316	3315	3314	3313	3312	3311	3310	3307	3306	3305	3304	3303	3302	3301	3300	V40754
3337	3336	3335	3334	3333	3332	3331	3330	3327	3326	3325	3324	3323	3322	3321	3320	V40755
3357	3356	3355	3354	3353	3352	3351	3350	3347	3346	3345	3344	3343	3342	3341	3340	V40756
3377	3376	3375	3374	3373	3372	3371	3370	3367	3366	3365	3364	3363	3362	3361	3360	V40757
3417	3416	3415	3414	3413	3412	3411	3410	3407	3406	3405	3404	3403	3402	3401	3400	V40750
3437	3436	3435	3434	3433	3432	3431	3430	3427	3426	3425	3424	3423	3422	3421	3420	V40751
3457	3456	3455	3454	3453	3452	3451	3450	3447	3446	3445	3444	3443	3442	3441	3440	V40752
3477	3476	3475	3474	3473	3472	3471	3470	3467	3466	3465	3464	3463	3462	3461	3460	V40753
3517	3516	3515	3514	3513	3512	3511	3510	3507	3506	3505	3504	3503	3502	3501	3500	V40754
3537	3536	3535	3534	3533	3532	3531	3530	3527	3526	3525	3524	3523	3522	3521	3520	V40755
3557	3556	3555	3554	3553	3552	3551	3550	3547	3546	3545	3544	3543	3542	3541	3540	V40756
3577	3576	3575	3574	3573	3572	3571	3570	3567	3566	3565	3564	3563	3562	3561	3560	V40757
3617	3616	3615	3614	3613	3612	3611	3610	3607	3606	3605	3604	3603	3602	3601	3600	V40750
3637	3636	3635	3634	3633	3632	3631	3630	3627	3626	3625	3624	3623	3622	3621	3620	V40751
3657	3656	3655	3654	3653	3652	3651	3650	3647	3646	3645	3644	3643	3642	3641	3640	V40752
3677	3676	3675	3674	3673	3672	3671	3670	3667	3666	3665	3664	3663	3662	3661	3660	V40753
3717	3716	3715	3714	3713	3712	3711	3710	3707	3706	3705	3704	3703	3702	3701	3700	V40754
3737	3736	3735	3734	3733	3732	3731	3730	3727	3726	3725	3724	3723	3722	3721	3720	V40755
3757	3756	3755	3754	3753	3752	3751	3750	3747	3746	3745	3744	3743	3742	3741	3740	V40756
3777	3776	3775	3774	3773	3772	3771	3770	3767	3766	3765	3764	3763	3762	3761	3760	V40757

Карты битового отображения состояний таймера и счетчика

В данной таблице приводится список отдельных контактов таймера и счетчика, связанных с каждым битом адреса V-памяти.

MSB							Контакты таймера (T) и счетчика (CT) в DL430, DL440, DL450 LSB							LSB	Адрес таймера	Адрес счетчика
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0	
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000	V41100	V41140
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	V41101	V41141
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	V41102	V41142
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	V41103	V41143
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	V41104	V41144
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	V41105	V41145
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	V41106	V41146
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	V41107	V41147

В данной части таблицы приводятся дополнительные контакты таймера, доступного в DL440 и DL450.

MSB								Дополнительные контакты таймера (T) в DL440, DL450							LSB	Адрес таймера
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0	
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200	V41110
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220	V41111
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240	V41112
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260	V41113
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300	V41114
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320	V41115
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340	V41116
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360	V41117

В данной части таблицы приводятся дополнительные контакты счетчика, доступного в DL450.

MSB								Дополнительные контакты счетчика (CT) в DL450							LSB	Адрес счетчика
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0	
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200	V41150
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220	V41151
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240	V41152
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260	V41153
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300	V41154
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320	V41155
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340	V41156
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360	V41157

Карта битового отображения удаленного ввода/вывода

В данной таблице приводится список отдельных точек удаленного ввода/вывода, связанных с каждым битом адреса V-памяти. Процессоры DL430 и DL440 используют точки типа GX как для удаленного ввода, так и для удаленного вывода. Процессор и DL450 имеет дополнительные точки типа GY, которые используются как указатели точек удаленного вывода.

MSB	Точки удаленного ввода/вывода (GX) и (GY) в DL430 в DL440 и DL450															LSB	Адрес GX	Адрес GY
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0			
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000		V40000	V40200
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020		V40001	V40201
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040		V40002	V40202
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060		V40003	V40203
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100		V40004	V40204
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120		V40005	V40205
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140		V40006	V40206
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160		V40007	V40207
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200		V40010	V40210
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220		V40011	V40211
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240		V40012	V40212
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260		V40013	V40213
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300		V40014	V40214
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320		V40015	V40215
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340		V40016	V40216
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360		V40017	V40217
417	416	415	414	413	412	411	410	407	406	405	404	403	402	401	400		V40020	V40220
437	436	435	434	433	432	431	430	427	426	425	424	423	422	421	420		V40021	V40221
457	456	455	454	453	452	451	450	447	446	445	444	443	442	441	440		V40022	V40222
477	476	475	474	473	472	471	470	467	466	465	464	463	462	461	460		V40023	V40223
517	516	515	514	513	512	511	510	507	506	505	504	503	502	501	500		V40024	V40224
537	536	535	534	533	532	531	530	527	526	525	524	523	522	521	520		V40025	V40225
557	556	555	554	553	552	551	550	547	546	545	544	543	542	541	540		V40026	V40226
577	576	575	574	573	572	571	570	567	566	565	564	563	562	561	560		V40027	V40227
617	616	615	614	613	612	611	610	607	606	605	604	603	602	601	600		V40030	V40230
637	636	635	634	633	632	631	630	627	626	625	624	623	622	621	620		V40031	V40231
657	656	655	654	653	652	651	650	647	646	645	644	643	642	641	640		V40032	V40232
677	676	675	674	673	672	671	670	667	666	665	664	663	662	661	660		V40033	V40233
717	716	715	714	713	712	711	710	707	706	705	704	703	702	701	700		V40034	V40234
737	736	735	734	733	732	731	730	727	726	725	724	723	722	721	720		V40035	V40235
757	756	755	754	753	752	751	750	747	746	745	744	743	742	741	740		V40036	V40236
777	776	775	774	773	772	771	770	767	766	765	764	763	762	761	760		V40037	V40237

В данной части таблицы приводятся дополнительные точки удаленного ввода/вывода типа GX для DL440 и DL450. Точки удаленного вывода типа GY доступны только для DL450 (точки типа GX в DL440 работают как точки ввода и вывода).

MSB **Дополнительные точки удаленного ввода/вывода (GX) в DL440 и DL450** **LSB**																**Адрес GX**	**Адрес GY** (DL450)
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0		
1017	1016	1015	1014	1013	1012	1011	1010	1007	1006	1005	1004	1003	1002	1001	1000	V40040	V40240
1037	1036	1035	1034	1033	1032	1031	1030	1027	1026	1025	1024	1023	1022	1021	1020	V40041	V40241
1057	1056	1055	1054	1053	1052	1051	1050	1047	1046	1045	1044	1043	1042	1041	1040	V40042	V40242
1077	1076	1075	1074	1073	1072	1071	1070	1067	1066	1065	1064	1063	1062	1061	1060	V40043	V40243
1117	1116	1115	1114	1113	1112	1111	1110	1107	1106	1105	1104	1103	1102	1101	1100	V40044	V40244
1137	1136	1135	1134	1133	1132	1131	1130	1127	1126	1125	1124	1123	1122	1121	1120	V40045	V40245
1157	1156	1155	1154	1153	1152	1151	1150	1147	1146	1145	1144	1143	1142	1141	1140	V40046	V40246
1177	1176	1175	1174	1173	1172	1171	1170	1167	1166	1165	1164	1163	1162	1161	1160	V40047	V40247
1217	1216	1215	1214	1213	1212	1211	1210	1207	1206	1205	1204	1203	1202	1201	1200	V40050	V40250
1237	1236	1235	1234	1233	1232	1231	1230	1227	1226	1225	1224	1223	1222	1221	1220	V40051	V40251
1257	1256	1255	1254	1253	1252	1251	1250	1247	1246	1245	1244	1243	1242	1241	1240	V40052	V40252
1277	1276	1275	1274	1273	1272	1271	1270	1267	1266	1265	1264	1263	1262	1261	1260	V40053	V40253
1317	1316	1315	1314	1313	1312	1311	1310	1307	1306	1305	1304	1303	1302	1301	1300	V40054	V40254
1337	1336	1335	1334	1333	1332	1331	1330	1327	1326	1325	1324	1323	1322	1321	1320	V40055	V40255
1357	1356	1355	1354	1353	1352	1351	1350	1347	1346	1345	1344	1343	1342	1341	1340	V40056	V40256
1377	1376	1375	1374	1373	1372	1371	1370	1367	1366	1365	1364	1363	1362	1361	1360	V40057	V40257
1417	1416	1415	1414	1413	1412	1411	1410	1407	1406	1405	1404	1403	1402	1401	1400	V40060	V40260
1437	1436	1435	1434	1433	1432	1431	1430	1427	1426	1425	1424	1423	1422	1421	1420	V40061	V40261
1457	1456	1455	1454	1453	1452	1451	1450	1447	1446	1445	1444	1443	1442	1441	1440	V40062	V40262
1477	1476	1475	1474	1473	1472	1471	1470	1467	1466	1465	1464	1463	1462	1461	1460	V40063	V40263
1517	1516	1515	1514	1513	1512	1511	1510	1507	1506	1505	1504	1503	1502	1501	1500	V40064	V40264
1537	1536	1535	1534	1533	1532	1531	1530	1527	1526	1525	1524	1523	1522	1521	1520	V40065	V40265
1557	1556	1555	1554	1553	1552	1551	1550	1547	1546	1545	1544	1543	1542	1541	1540	V40066	V40266
1577	1576	1575	1574	1573	1572	1571	1570	1567	1566	1565	1564	1563	1562	1561	1560	V40067	V40267
1617	1616	1615	1614	1613	1612	1611	1610	1607	1606	1605	1604	1603	1602	1601	1600	V40070	V40270
1637	1636	1635	1634	1633	1632	1631	1630	1627	1626	1625	1624	1623	1622	1621	1620	V40071	V40271
1657	1656	1655	1654	1653	1652	1651	1650	1647	1646	1645	1644	1643	1642	1641	1640	V40072	V40272
1677	1676	1675	1674	1673	1672	1671	1670	1667	1666	1665	1664	1663	1662	1661	1660	V40073	V40273
1717	1716	1715	1714	1713	1712	1711	1710	1707	1706	1705	1704	1703	1702	1701	1700	V40074	V40274
1737	1736	1735	1734	1733	1732	1731	1730	1727	1726	1725	1724	1723	1722	1721	1720	V40075	V40275
1757	1756	1755	1754	1753	1752	1751	1750	1747	1746	1745	1744	1743	1742	1741	1740	V40076	V40276
1777	1776	1775	1774	1773	1772	1771	1770	1767	1766	1765	1764	1763	1762	1761	1760	V40077	V40277

В данной части таблицы приводятся дополнительные точки удаленного ввода/вывода (GX и GY), доступные только для DL450.

MSB								**Дополнительные точки удаленного ввода/вывода (GX) и (GY) в DL450**							**LSB**	**Адрес GX**	**Адрес GY**
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0		
2017	2016	2015	2014	2013	2012	2011	2010	2007	2006	2005	2004	2003	2002	2001	2000	V40100	V40300
2037	2036	2035	2034	2033	2032	2031	2030	2027	2026	2025	2024	2023	2022	2021	2020	V40101	V40301
2057	2056	2055	2054	2053	2052	2051	2050	2047	2046	2045	2044	2043	2042	2041	2040	V40102	V40302
2077	2076	2075	2074	2073	2072	2071	2070	2067	2066	2065	2064	2063	2062	2061	2060	V40103	V40303
2117	2116	2115	2114	2113	2112	2111	2110	2107	2106	2105	2104	2103	2102	2101	2100	V40104	V40304
2137	2136	2135	2134	2133	2132	2131	2130	2127	2126	2125	2124	2123	2122	2121	2120	V40105	V40305
2157	2156	2155	2154	2153	2152	2151	2150	2147	2146	2145	2144	2143	2142	2141	2140	V40106	V40306
2177	2176	2175	2174	2173	2172	2171	2170	2167	2166	2165	2164	2163	2162	2161	2160	V40107	V40307
2217	2216	2215	2214	2213	2212	2211	2210	2207	2206	2205	2204	2203	2202	2201	2200	V40110	V40310
2237	2236	2235	2234	2233	2232	2231	2230	2227	2226	2225	2224	2223	2222	2221	2220	V40111	V40311
2257	2256	2255	2254	2253	2252	2251	2250	2247	2246	2245	2244	2243	2242	2241	2240	V40112	V40312
2277	2276	2275	2274	2273	2272	2271	2270	2267	2266	2265	2264	2263	2262	2261	2260	V40113	V40313
2317	2316	2315	2314	2313	2312	2311	2310	2307	2306	2305	2304	2303	2302	2301	2300	V40114	V40314
2337	2336	2335	2334	2333	2332	2331	2330	2327	2326	2325	2324	2323	2322	2321	2320	V40115	V40315
2357	2356	2355	2354	2353	2352	2351	2350	2347	2346	2345	2344	2343	2342	2341	2340	V40116	V40316
2377	2376	2375	2374	2373	2372	2371	2370	2367	2366	2365	2364	2363	2362	2361	2360	V40117	V40317
2417	2416	2415	2414	2413	2412	2411	2410	2407	2406	2405	2404	2403	2402	2401	2400	V40120	V40320
2437	2436	2435	2434	2433	2432	2431	2430	2427	2426	2425	2424	2423	2422	2421	2420	V40121	V40321
2457	2456	2455	2454	2453	2452	2451	2450	2447	2446	2445	2444	2443	2442	2441	2440	V40122	V40322
2477	2476	2475	2474	2473	2472	2471	2470	2467	2466	2465	2464	2463	2462	2461	2460	V40123	V40323
2517	2516	2515	2514	2513	2512	2511	2510	2507	2506	2505	2504	2503	2502	2501	2500	V40124	V40324
2537	2536	2535	2534	2533	2532	2531	2530	2527	2526	2525	2524	2523	2522	2521	2520	V40125	V40325
2557	2556	2555	2554	2553	2552	2551	2550	2547	2546	2545	2544	2543	2542	2541	2540	V40126	V40326
2577	2576	2575	2574	2573	2572	2571	2570	2567	2566	2565	2564	2563	2562	2561	2560	V40127	V40327
2617	2616	2615	2614	2613	2612	2611	2610	2607	2606	2605	2604	2603	2602	2601	2600	V40130	V40330
2637	2636	2635	2634	2633	2632	2631	2630	2627	2626	2625	2624	2623	2622	2621	2620	V40131	V40331
2657	2656	2655	2654	2653	2652	2651	2650	2647	2646	2645	2644	2643	2642	2641	2640	V40132	V40332
2677	2676	2675	2674	2673	2672	2671	2670	2667	2666	2665	2664	2663	2662	2661	2660	V40133	V40333
2717	2716	2715	2714	2713	2712	2711	2710	2707	2706	2705	2704	2703	2702	2701	2700	V40134	V40334
2737	2736	2735	2734	2733	2732	2731	2730	2727	2726	2725	2724	2723	2722	2721	2720	V40135	V40335
2757	2756	2755	2754	2753	2752	2751	2750	2747	2746	2745	2744	2743	2742	2741	2740	V40136	V40336
2777	2776	2775	2774	2773	2772	2771	2770	2767	2766	2765	2764	2763	2762	2761	2760	V40137	V40337

Карта битового отображения управления/состояния стадий

В данной таблице приводится список отдельных битов управления стадиями, связанных с каждым адресом V-памяти.

MSB							Биты управления стадиями (S) в DL430, DL440 и DL450								LSB	Адрес
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0	
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000	V41000
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	V41001
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	V41002
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	V41003
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	V41004
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	V41005
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	V41006
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	V41007
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200	V41010
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220	V41011
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240	V41012
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260	V41013
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300	V41014
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320	V41015
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340	V41016
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360	V41017
417	416	415	414	413	412	411	410	407	406	405	404	403	402	401	400	V41020
437	436	435	434	433	432	431	430	427	426	425	424	423	422	421	420	V41021
457	456	455	454	453	452	451	450	447	446	445	444	443	442	441	440	V41022
477	476	475	474	473	472	471	470	467	466	465	464	463	462	461	460	V41023
517	516	515	514	513	512	511	510	507	506	505	504	503	502	501	500	V41024
537	536	535	534	533	532	531	530	527	526	525	524	523	522	521	520	V41025
557	556	555	554	553	552	551	550	547	546	545	544	543	542	541	540	V41026
577	576	575	574	573	572	571	570	567	566	565	564	563	562	561	560	V41027

MSB							Дополнительные биты управления стадиями (S) в DL440 и DL450								LSB	Адрес
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0	
617	616	615	614	613	612	611	610	607	606	605	604	603	602	601	600	V41030
637	636	635	634	633	632	631	630	627	626	625	624	623	622	621	620	V41031
657	656	655	654	653	652	651	650	647	646	645	644	643	642	641	640	V41032
677	676	675	674	673	672	671	670	667	666	665	664	663	662	661	660	V41033
717	716	715	714	713	712	711	710	707	706	705	704	703	702	701	700	V41034
737	736	735	734	733	732	731	730	727	726	725	724	723	722	721	720	V41035
757	756	755	754	753	752	751	750	747	746	745	744	743	742	741	740	V41036
777	776	775	774	773	772	771	770	767	766	765	764	763	762	761	760	V41037

MSB	Дополнительные биты управления стадиями (S) в DL440 и DL450															LSB	Адрес
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0		
1017	1016	1015	1014	1013	1012	1011	1010	1007	1006	1005	1004	1003	1002	1001	1000		V41040
1037	1036	1035	1034	1033	1032	1031	1030	1027	1026	1025	1024	1023	1022	1021	1020		V41041
1057	1056	1055	1054	1053	1052	1051	1050	1047	1046	1045	1044	1043	1042	1041	1040		V41042
1077	1076	1075	1074	1073	1072	1071	1070	1067	1066	1065	1064	1063	1062	1061	1060		V41043
1117	1116	1115	1114	1113	1112	1111	1110	1107	1106	1105	1104	1103	1102	1101	1100		V41044
1137	1136	1135	1134	1133	1132	1131	1130	1127	1126	1125	1124	1123	1122	1121	1120		V41045
1157	1156	1155	1154	1153	1152	1151	1150	1147	1146	1145	1144	1143	1142	1141	1140		V41046
1177	1176	1175	1174	1173	1172	1171	1170	1167	1166	1165	1164	1163	1162	1161	1160		V41047
1217	1216	1215	1214	1213	1212	1211	1210	1207	1206	1205	1204	1203	1202	1201	1200		V41050
1237	1236	1235	1234	1233	1222	1231	1230	1227	1226	1225	1224	1223	1222	1221	1220		V41051
1257	1256	1255	1254	1253	1252	1251	1250	1247	1246	1245	1244	1243	1242	1241	1240		V41052
1277	1276	1275	1274	1273	1272	1271	1270	1267	1266	1265	1264	1263	1262	1261	1260		V41053
1317	1316	1315	1314	1313	1312	1311	1310	1307	1306	1305	1304	1303	1302	1301	1300		V41054
1337	1336	1335	1334	1333	1332	1331	1330	1327	1326	1325	1324	1323	1322	1321	1320		V41055
1357	1356	1355	1354	1353	1352	1351	1350	1347	1346	1345	1344	1343	1342	1341	1340		V41056
1377	1376	1375	1374	1373	1372	1371	1370	1367	1366	1365	1364	1363	1362	1361	1360		V41057
1417	1416	1415	1414	1413	1412	1411	1410	1407	1406	1405	1404	1403	1402	1401	1400		V41060
1437	1436	1435	1434	1433	1432	1431	1430	1427	1426	1425	1424	1423	1422	1421	1420		V41061
1457	1456	1455	1454	1453	1452	1451	1450	1447	1446	1445	1444	1443	1442	1441	1440		V41062
1477	1476	1475	1474	1473	1472	1471	1470	1467	1466	1465	1464	1463	1462	1461	1460		V41063
1517	1516	1515	1514	1513	1512	1511	1510	1507	1506	1505	1504	1503	1502	1501	1500		V41064
1537	1536	1535	1534	1533	1532	1531	1530	1527	1526	1525	1524	1523	1522	1521	1520		V41065
1557	1556	1555	1554	1553	1552	1551	1550	1547	1546	1545	1544	1543	1542	1541	1540		V41066
1577	1576	1575	1574	1573	1572	1571	1570	1567	1566	1565	1564	1563	1562	1561	1560		V41067
1617	1616	1615	1614	1613	1612	1611	1610	1607	1606	1605	1604	1603	1602	1601	1600		V41070
1637	1636	1635	1634	1633	1632	1631	1630	1627	1626	1625	1624	1623	1622	1621	1620		V41071
1657	1656	1655	1654	1653	1652	1651	1650	1647	1646	1645	1644	1643	1642	1641	1640		V41072
1677	1676	1675	1674	1673	1672	1671	1670	1667	1666	1665	1664	1663	1662	1661	1660		V41073
1717	1716	1715	1714	1713	1712	1711	1710	1707	1706	1705	1704	1703	1702	1701	1700		V41074
1737	1736	1735	1734	1733	1732	1731	1730	1727	1726	1725	1724	1723	1722	1721	1720		V41075
1757	1756	1755	1754	1753	1752	1751	1750	1747	1746	1745	1744	1743	1742	1741	1740		V41076
1777	1776	1775	1774	1773	1772	1771	1770	1767	1766	1765	1764	1763	1762	1761	1760		V41077

Проектирование и конфигурирование системы

4

В этой главе...

— Стратегии проектирования системы на базе DL405
— Размещение и конфигурирование модулей
— Расчет потребляемой мощности
— Расширение локального ввода/вывода
— Расширение удаленного ввода/вывода
— Расширение секционированного ввода/вывода
— Сетевые подключения к MODBUS и DirectNET
— Функционирование ведомого устройства в сети
— Функционирование ведущего устройства в сети

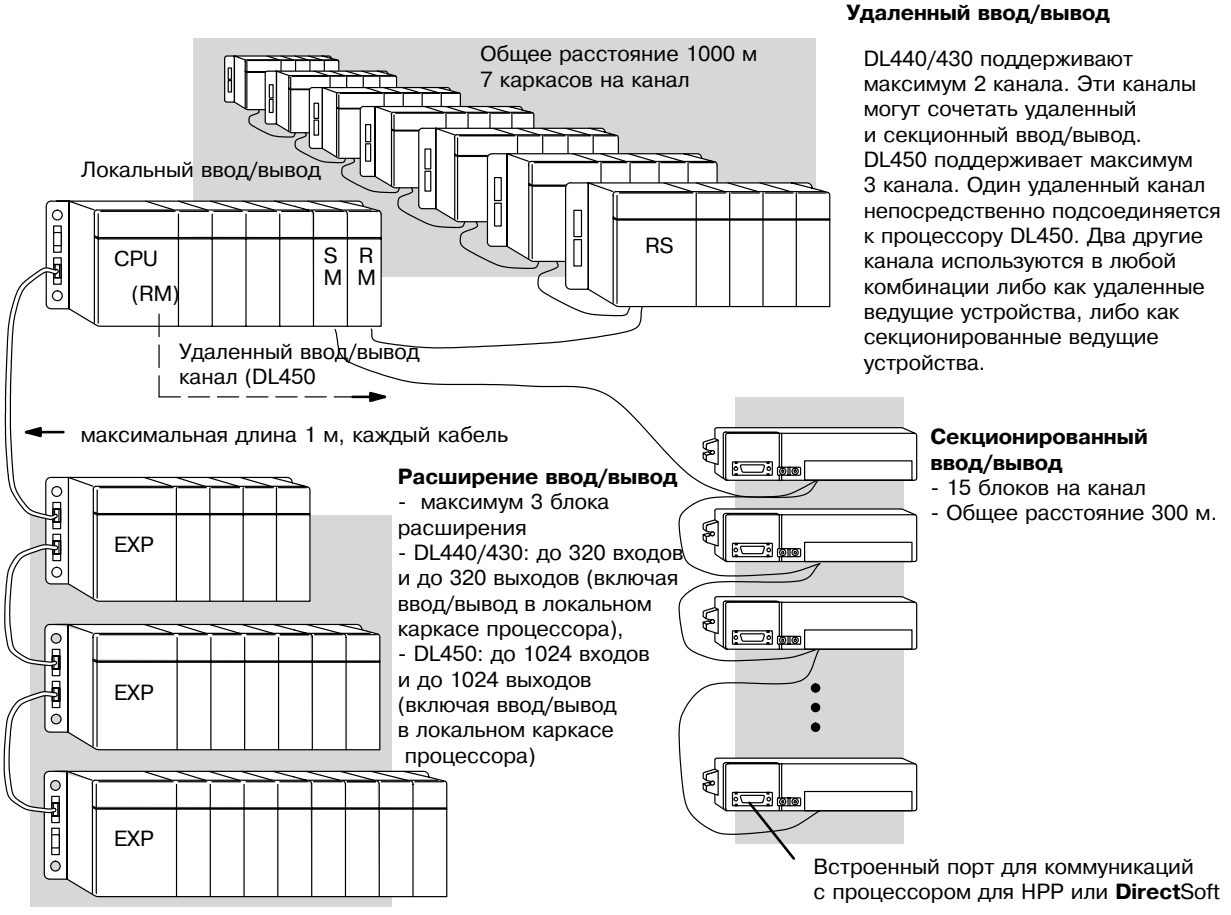
Стратегии проектирования системы на базе DL405

Конфигурации подсистемы ввода/вывода

Процессоры DL405 предусматривают различные способы подключения сетевых функций к системе:

- **Локальный ввод/вывод** - состоит из модулей ввода/вывода, расположенных в том же каркасе, что и процессор.
- **Расширение ввода/вывода** - состоит из модулей ввода/вывода, расположенных в каркасах расширения рядом с локальным каркасом. Кабели расширения подсоединяют их к шине процессора локального каркаса с помощью последовательного (гирляндного) подключения.
- **Удаленный ввод/вывод (Remote I/O)** - состоит из модулей ввода/вывода, которые подсоединяются к процессору локального каркаса через удаленный ведущий модуль, или непосредственно подключаются к порту 3 процессора DL450.
- **Секционированный ввод/вывод (Slice I/O)** - состоит из небольших фиксированных блоков ввода/вывода (не модулей), которые последовательно подсоединяются к процессору локального каркаса через секционированный ведущий модуль. Заметьте, что вы не можете подсоединить секционированный ввод/вывод непосредственно к процессору DL450.

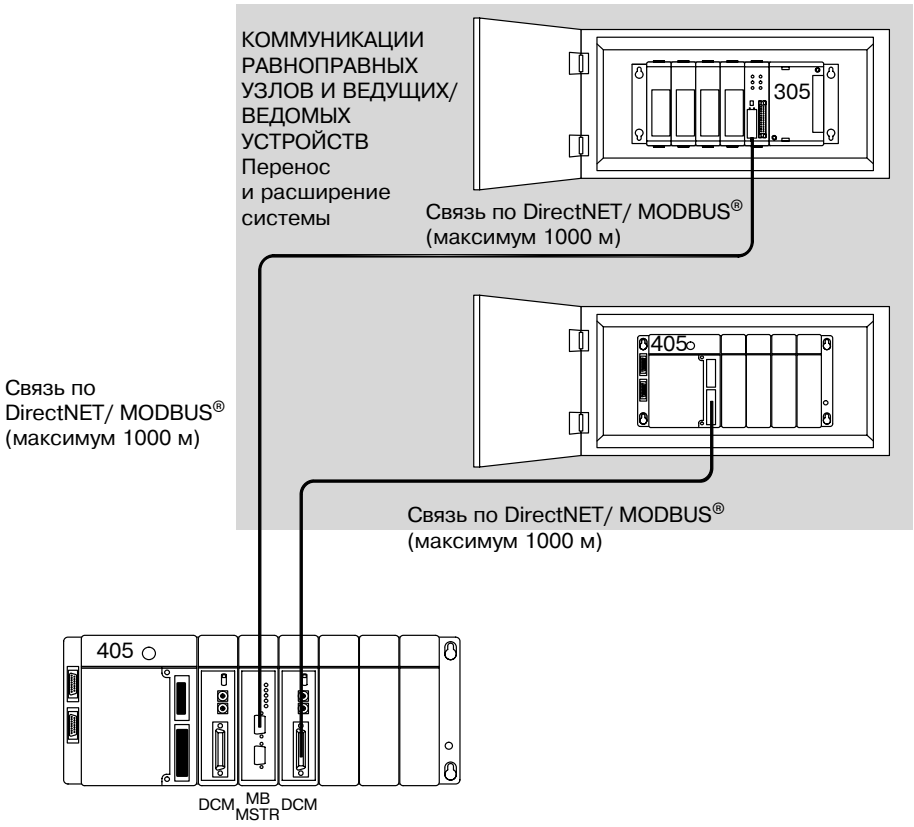
Система DL405 может разрабатываться при различных сочетаниях этих конфигураций. Все конфигурации подсистемы ввода/вывода, кроме секционных ведомых блоков, используют стандартные комплекты модулей и каркасов подсистемы ввода/вывода DL405. Секционные ведомые блоки имеют собственные каркасы и действуют как отдельные вводы/выводы. Ниже приводится краткое описание каждой из этих конфигураций. Позже в данной главе будут детально рассмотрены примеры каждой конфигурации.



Конфигурации подключения к сети

ПЛК DL405 предусматривают следующие четыре способа подключения модулей ввода/вывода к системе:

- **Через модуль коммуникации данных** - соединяет систему DL405 с устройствами, использующими протокол DirectNET, или подсоединяет как ведомое устройство к сети MODBUS.
- **Через коммуникационные порты** - процессор DL450 имеет два дополнительных (а всего четыре) встроенных коммуникационных порта. Он допускает два сетевых подключения непосредственно к процессору. См. главу 3 "Спецификации и работа процессора" по спецификациям конкретных портов и разделы в конце данной главы по сетевым соединениям.
- **Через ведущий модуль MODBUS** - Вы можете использовать ведущие модули MODBUS в любом слоте системы DL405 для подключения его как ведущего устройства к сети MODBUS с использованием протокола RTU.
- **Через ведомый модуль MODBUS** - Вы можете использовать ведомые модули MODBUS в любом слоте системы DL405 для подключения его как ведомого устройства к сети MODBUS с использованием протокола RTU.
- **Через модуль сетевого интерфейса TIWAY®** - Интерфейса с сетями Texas Instrument and Siemens TIWAY с использованием этого модуля как ведомого устройства.
- **Через сетевой модуль общих данных** - Сетевой модуль общих данных позволяет сделать равноправные соединения между системами ПЛК DL405.



Размещение и конфигурирование модулей

Правильное размещение модулей/ блоков

Большинство обычно используемых модулей ввода/вывода для системы DL405 (переменного, постоянного, переменного/постоянного тока, релейные, аналоговые) могут располагаться в вашей системе в любом каркасе. В приведенной ниже таблице указывается правильное размещение всех модулей/блоков в системе DL405. Напоминаем, что лимит мощности может ограничить число модулей в каркасе (эта проблема рассматривается ниже).

Модуль/блок	Локальный каркас процессора	Расширение локал. каркаса	Удаленный каркас
Процессоры	Только слот процессора		
Блоки расширения		Только слот процессора	
Входные модули постоянного тока на 8/16/32 точки	√	√	√
Входные модули постоянного тока на 64 точки	√ Примечание 1	√ Примечание 1, 2	
Входные модули переменного тока	√	√	√
Входные модули постоянного/ переменного тока	√	√	√
Выходные модули постоянного тока на 8/16/32 точки	√	√	√
Выходные модули постоянного тока на 64 точки	√ Примечание 1	√ Примечание 1, 2	
Выходные модули переменного тока	√	√	√
Релейные Выходные Модули	√	√	√
Аналоговые Модули	√	√	√
Удаленный ввод/вывод			
Удаленное ведущее уст-во	√	√ Примечание 2	
Удаленный ведомый блок			Только слот процессора
Секционированное ведущее уст-во	√	√ Примечание 2	
Коммуникационные и сетевые модули	√	√ Примечание 2	
Модули сопроцессора	√	√ Примечание 2	
Специальные модули			
Прерывания	DL430 -только слот 0 DL440 -только слот 0 и 1 DL450 -только слот 0		
Высокоскоростной счетчик	√	√	
Модуль ПИД - регулятора	√	√ Примечание 2	
Система распределения сигнала (SDS)	√	√ Примечание 2	
4-контурный контроллер темературы	√	√ Примечание 2	
Имитатор входа	√	√	√
Фильтр	√	√	√

Примечание 1. При использовании модулей на 64 точки Вы не можете применять любые специальные модули в слотах 5, 6, 7 в том же каркасе.

Примечание 2. Специальные модули допустимы в каркасах расширения только тогда, когда используется процессор DL450, а все каркасы в системе имеют тип D4-xxB-1.

Методы конфигурирования ввода/вывода

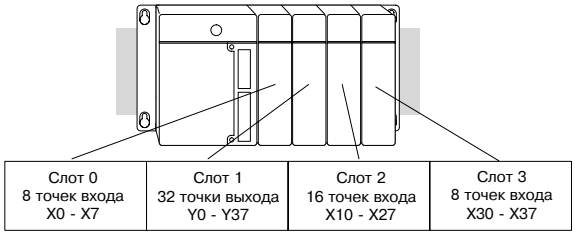
Существует два метода конфигурирования ввода/вывода для процессоров DL405:

- **Автоматическое конфигурирование:** Процессор автоматически конфигурирует ввод/вывод. Он присваивает наименьший номер модулю в слоте 0 (слот рядом с процессором), следующий номер последующему модулю в каркасе и т. д. Номера присваиваются только модулям, фактически находящимся в каркасе, пустые слоты не нумеруются. Это - режим процессора по умолчанию.
- **Ручное конфигурирование: (только для DL440/DL450).** Позволяет вам присваивать номера вводам/выводам. Номера могут назначаться пустым слотам или в любом порядке, если только они идут группами по 16 или по 32.

Автоматическое конфигурирование

Процессоры DL405 автоматически определяют при включении питания любой из установленных модулей ввода/вывода (включая специальные модули), а также устанавливает точную конфигурацию и адреса точек ввода/вывода. В большинстве приложений конфигурация никогда не меняется.

При адресации точек ввода/вывода используется восьмеричная система счисления, адреса начинаются с X0 и Y0 в следующем от процессора слоте. Адреса присваиваются группами по 8, 16, 32 или 64 в зависимости от числа точек в модуле ввода/вывода. Дискретные входные и выходные модули могут располагаться в любом порядке, но существуют ограничения на размещение некоторых специальных модулей. На приводимой ниже схеме иллюстрируется соглашение по нумерации точек ввода/вывода на примере отдельной системы.



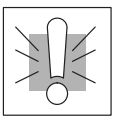
Как ручной программатор, так и DirectSOFT, имеют AUX функцию, с помощью которой вы можете автоматически сконфигурировать подсистему ввода/вывода. Например, на ручном программаторе функция AUX 46 выполняет автоматическое конфигурирование, которое позволяет процессору проверить установленные модули ввода/вывода, определить конфигурацию и адресацию ввода/вывода. В DirectSOFT может использоваться опция меню ПЛК "Конфигурировать ввод/вывод".

Возможно вам никогда не понадобится эта функция, но процессоры DL440 и DL450 позволяют вручную присвоить адреса модулей ввода/вывода для любого слота в локальном каркасе или каркасе расширения. Вы можете вручную изменять автоматически установленную конфигурацию, присваивать произвольную нумерацию модулей ввода/вывода. Например, два соседних входных модуля могут иметь начальные адреса X10 и X200.

Ручное конфигурирование

X	√	√
430	440	450

При автоматическом конфигурировании адреса присваиваются в рамках 8 точек. Однако при ручном конфигурировании предполагается, что все модули включают, по крайней мере, 16 точек, поэтому вы можете присваивать адреса, кратные 20 (восьмеричные). Например, X30 и X50 являются неправильными адресами. Вы можете использовать модули на 8 точек, однако присваивать 16 адресов, из которых 8 точек с большими адресами не будут использоваться.

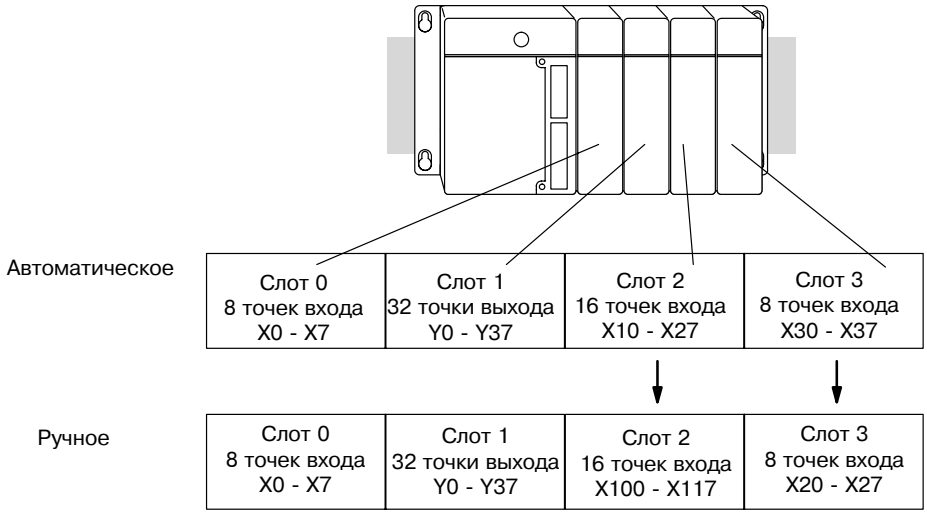


ПРЕДУПРЕЖДЕНИЕ. Если вы вручную сконфигурировали какой-то слот ввода/вывода, то адресация других модулей ввода/вывода может измениться. Это происходит потому, что процессор DL405 не допускает присвоения дублированных адресов для модулей ввода/вывода. Перед тем как вы переведете процессор в Рабочий режим, вы всегда должны исправить ошибки в конфигурации ввода/вывода. Неисправленные ошибки могут вызвать непредсказуемую работу аппаратуры, что может привести к риску травмирования персонала или к повреждению оборудования.

Удаление ручной конфигурации

После ручного конфигурирования система автоматически сохраняет новые адреса модулей ввода/вывода в данном цикле включения питания. Вы можете удалить (стереть) все ручные изменения в конфигурации, просто перейдя на автоматическое конфигурирование.

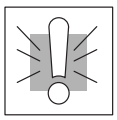
На следующем рисунке показано, как изменяются адреса ввода/вывода после ручного конфигурирования слотов.



Проверка конфигурации ввода/вывода при включении питания

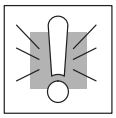
Проверка конфигурации ввода/вывода при включении питания Процессоры DL405 могут быть также установлены на автоматическую проверку конфигурации модулей ввода/вывода при включении питания. Выбрав эту функцию, вы можете обнаружить любое изменение, которое могло произойти за то время, пока питание было отключено. Например, если кто-то установил выходной модуль в слот, ранее поддерживаемый как входной модуль, то при проверке конфигурации будет обнаружено это изменение и на экран ручной программатора или на экран DirectSOFT будет выведено соответствующее сообщение (используйте функцию AUX 44 на HPP для проверки конфигурации).

Если система при включении питания и при проверке конфигурации ввода/вывода обнаруживает изменение, она генерирует код ошибки E522 НОВАЯ КОНФИГУРАЦИЯ ВВОДА/ВЫВОДА. Чтобы определить точный адрес каркаса и слота, где это изменение произошло, вы можете использовать функцию AUX 42.



ПРЕДУПРЕЖДЕНИЕ. Перед тем, как перевести процессор в Рабочий режим, вы должны исправить ошибки в конфигурации ввода/вывода. Неисправленные ошибки могут вызвать непредсказуемую работу аппаратуры, что может привести к риску травмирования персонала и к повреждению оборудования.

При обнаружении ошибки в конфигурации вы можете пожелать применить новую конфигурацию ввода/вывода. Например, вы можете специально изменить модуль ввода/вывода, чтобы поработать с изменением в программе. Вы можете использовать функцию AUX 45 для выбора новой конфигурации или оставить существующую конфигурацию, сохраненную в памяти.



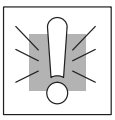
ПРЕДУПРЕЖДЕНИЕ. Проверенная конфигурация будет правильно работать со всеми программами процессора. Всегда исправляйте ошибки в конфигурации ввода/вывода перед переходом в Рабочий режим. Неисправленные ошибки могут вызвать непредсказуемую работу аппаратуры, что может привести к риску травмирования персонала и к повреждению оборудования.

Расчет потребляемой мощности

Управление вашими ресурсами мощности

Конфигурация ввода/вывода зависит от выбора модулей ввода/вывода, каркасов и размещения модулей. При определении типов и числа модулей ввода/вывода, которые вы собираетесь использовать в системе DL405, важно не забывать, что доступная мощность источника питания системы ограничена. В приводимой таблице показана потребляемая мощность для выбранного вами процессора, блока расширения или удаленного ведомого устройства. Следующая таблица поможет вам рассчитать мощность, необходимую для вашего варианта подсистемы ввода/вывода. В конце данного раздела дается пример расчета мощности и рабочие таблицы для ваших самостоятельных расчетов.

Если выбранная вами подсистема ввода/вывода требует большей мощности, чем максимальная доступная мощность источника питания, то вам следует перенести несколько модулей в каркас расширения, который имеет другой источник питания.



ПРЕДУПРЕЖДЕНИЕ. Чрезвычайно важно правильно провести расчет потребляемой мощности. Если вы превышаете доступную мощность, то система будет работать непредсказуемым образом, что может привести к нанесению вреда персоналу или к повреждению оборудования.

Характеристика мощности процессоров

В следующем разделе приводится значение доступного тока при двух напряжениях источников питания процессора DL405, блока расширения и удаленных ведомых устройств. Используйте эти значения тока при расчете мощности, потребляемой вашей системой. Указанный в таблице вспомогательный источник тока на 24 В выведен на клеммную колодку каркаса, что позволяет вам подсоединить к ней устройства или модули DL405, требующие напряжения 24 В постоянного тока.

Процессоры	Ток при напряжении 5 В, мА	Ток вспомогательного источника питания 24 В, мА		Процессоры	Ток при напряжении 5 В, мА	Ток вспомогательного источника питания 24 В, мА
D4-430	3700	400		D4-EX	4000	400
D4-440	3700	400		D4-EXDC	4000	Нет
D4-440DC-1	3700	Нет		D4-EXDC-2	3700	Нет
D4-440DC-2	3700	Нет		D4-RS	3700	400
D4-450	3100	400		D4-RSDC	3700	Нет

Требования к мощности модулей

В следующей таблице приведен максимальный ток, необходимый для каждого модуля DL405. Используйте эти токи для расчета потребляемой вашей системой мощности. Если необходим внешний источник питания на 24 В постоянного тока, то следует использовать соответствующий источник питания 24 В каркаса, пока не будет превышена его мощность.

Устройство	Требуемый ток при 5 В (мА)	Требуемый ток внешнего источника 24 В (мА)
Каркасы ввода/вывода		
D4-04B, D4-04BNX, D4-04B-1	80	Нет
D4-06B, D4-06BNX, D4-06B-1	80	Нет
D4-08B, D4-08BNX, D4-08B-1	80	Нет
Входные модули постоянного тока		
D4-08ND3S	100	Нет
D4-16ND2	150	Нет
D4-16ND2F	150	Нет
D4-32ND3-1	150	Нет
D4-32ND3-2	150	Нет
D4-64ND2	300 (max)	Нет
Входные модули переменного тока		
D4-08NA	100	Нет
D4-16NA	150	Нет
Входные модули перем./пост. тока		
D4-16NE3	150	Нет
F4-08NES	90	Нет
Выходные модули постоянного тока		
D4-08TD1	150	35
F4-08TD1S	295	Нет
D4-16TD1	200	125
D4-16TD2	350	Нет
D4-32TD1	250	140
D4-32TD1-1	250	140 (5-15VDC)
D4-32TD2	350	120/4A макс, включая нагрузку
D4-64TD1	800 (max)	Нет
Выходные модули переменного тока		
D4-08T A	250	Нет
D4-16T A	450	Нет
Релейные выходные модули		
D4-08TR	550	Нет
F4-08TRS-1	575	Нет
F4-08TRS-2	575	Нет
D4-16TR	1000	Нет
Программирование		
D4-HPP	320	Нет
DV-1000	150	Нет

Устройство	Требуемый ток при 5 В (мА)	Требуемый ток внешнего источника 24 В (мА)
Аналоговые модули		
D4-04AD	200	200
F4-04AD	85	100
F4-04ADS	270	120
F4-08AD	75	90
D4-02DA	250	300
F4-04DA	120	180
F4-04DA-1	70	75 + 20 на контурt
F4-04DA-2	70	75 + 20 на контурit
F4-08DA-1	70	100 + 20 на контур
F4-16DA-1	70	100 + 20 на контур
F4-08THM-n	120	50 + 20 на контур
Удаленный ввод/вывод		
D4-RM	300	Нет
D4-SM	300	Нет
D4-SS-88	Нет	100, (250 w/ HPP)
D4-SS-106	Нет	100, (250 w/ HPP)
D4-SS-16T	Нет	100, (250 w/ HPP)
D4-SS-16N	Нет	100, (250 w/ HPP)
Коммуникационные и сетевые модули		
D4-DCM	500	Нет
F4-MAS-MB	235	Нет
F4-MAS-MBR	350	Нет
F4-SL V-MB	235	Нет
F4-SL V-MBR	350	Нет
F4-SL V-TW	480	Нет
F4-SDN	235	Нет
FA-UNICON	Нет	65
Сопроцессоры ™		
F4-CP128	305	Нет
F4-CP512	235	Нет
F4-CP128-T	350	Нет
F4-CP128-R	350	Нет
Специальные модули		
D4-INT	100	Нет
D4-HSC	300	Нет
F4-16PID	160	Нет
F4-8MPI	225	170
D4-16SIM	150	Нет
F4-SDS	110	Нет
F4-4L TC	280	75

Пример расчета потребляемой мощности

На следующем примере показано, как надо рассчитывать потребляемую мощность для системы DL405.

№ каркаса 0	**Тип модуля**	**5 В пост. тока (мА)**	**Выход 24 В вспомогательного источника питания (мА)**
Используемые процессор/блок расширения/ удаленное ведомое устройство	D4-430	3700	400
Слот 0	D4-16ND2	+ 150	+ 0
Слот 1	D4-16ND2	+ 150	+ 0
Слот 2	D4-02DA	+ 250	+ 300
Слот 3	D4-08ND3S	+ 100	+ 0
Слот 4	D4-08ND3S	+ 100	+ 0
Слот 5	D4-16TD2	+ 400	+ 0
Слот6	D4-16TD2	+ 400	+ 0
Слот7	D4-16TR	+ 1000	+ 0
Другие			
Каркас	D4-08B	+ 80	+ 0
Ручной программатор	D4-HPP	+ 320	+ 0
Максимальная требуемая мощность		**2950**	**300**
Оставшаяся доступная мощность		3700-2950 = **750**	400 - 300 = **100**

1. Используйте таблицу расчета потребляемой мощности, заполните ее для процессора / блока расширения / удаленного ведомого устройства, модулей ввода/вывода и для любых других устройств, которые используют мощность системы, включая устройства, использующие выход 24 В постоянного тока. Обратите особое внимание на ток, потребляемый от процессора, блока расширения и удаленного ведомого устройства, поскольку это различное питание. Устройства, которые отнесены в категорию ”Другие” - это устройства типа каркаса и ручного программатора, они получают питание из системы, хотя непосредственно и не установлены в каркасе.
2. Заполните столбцы для потребляемого тока, начиная со слота 0, проставьте итог в строке ”Максимальная требуемая мощность”.
3. Вычтите строку ”Максимальная требуемая мощность” из строки ”Используемые процессор/блок расширения/удаленное ведомое устройство”. Запишите разность в строку ”Оставшаяся доступная мощность”.
4. Если ”Максимальная требуемая мощность” больше, чем ”Используемые процессор/блок расширения/удаленное ведомое устройство”, то баланс мощности превышен. Небезопасно использовать такую конфигурацию, вам следует изменить вашу конфигурацию подсистемы ввода/вывода.

Рабочая таблица для расчета потребляемой мощности

Вы можете скопировать и использовать следующую пустую таблицу для ваших расчетов потребляемой мощности.

№ каркаса _____	Тип модуля	5 В пост. тока (мА)	Выход 24 В вспомогательного источника питания (мА)
Используемые процессор/блок расширения/ удаленное ведомое устройство			
Слот 0			
Слот 1			
Слот 2			
Слот 3			
Слот 4			
Слот 5			
Слот6			
Слот7			
Другие			
Максимальная требуемая мощность			
Оставшаяся доступная мощность			

1. Используйте таблицу расчета потребляемой мощности, заполните ее для процессора / блока расширения / удаленного ведомого устройства, модулей ввода/вывода и для любых других устройств, которые используют мощность системы, включая устройства, использующие выход 24 В постоянного тока. Обратите особое внимание на ток, потребляемый от процессора, блока расширения и удаленного ведомого устройства, поскольку это различное питание. Устройства, которые отнесены в категорию ”Другие” - это устройства типа каркаса и ручного программатора, они получают питание из системы, хотя непосредственно и не установлены в каркасе.
2. Заполните столбцы для потребляемого тока, начиная со слота 0, проставьте итог в строке ”Максимальная требуемая мощность”.
3. Вычтите строку ”Максимальная требуемая мощность” из строки ”Используемые процессор/блок расширения/удаленное ведомое устройство”. Запишите разность в строку ”Оставшаяся доступная мощность”.
4. Если ”Максимальная требуемая мощность” больше, чем ”Используемые процессор/блок расширения/удаленное ведомое устройство”, то баланс мощности превышен. Небезопасно использовать такую конфигурацию, вам следует изменить вашу конфигурацию подсистемы ввода/вывода.

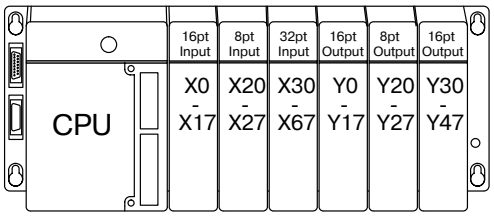
Расширение локального ввода/вывода

√	√	√
430	440	450

Следующие конфигурации каркасов ввода/вывода помогут вам понять доступные в серии DL405 варианты. Локальные и расширенные каркасы - это наиболее общие и эффективные по стоимости способы установки ввода/вывода. Процессор может автоматически сконфигурировать для вас локальный и расширенный ввод/вывод. Используйте удаленный и секционированный ввод/вывод, когда вам необходимо разместить точки ввода/вывода на достаточном расстоянии от процессора. Но для работы как удаленного, так и секционированного ввода/вывода требуется дополнительное программирование.

Локальный каркас и ввод/вывод

Локальным каркасом является каркас, в котором постоянно находится процессор. Модули локального ввода/вывода постоянно находятся в том же каркасе, в каком находится процессор. Например, размещение модулей на 32 точки во всех восьми слотах каркаса на 8 слотов позволит использовать 256 точек ввода/вывода. Состояние каждой точки обновляется в каждом цикле сканирования процессора.

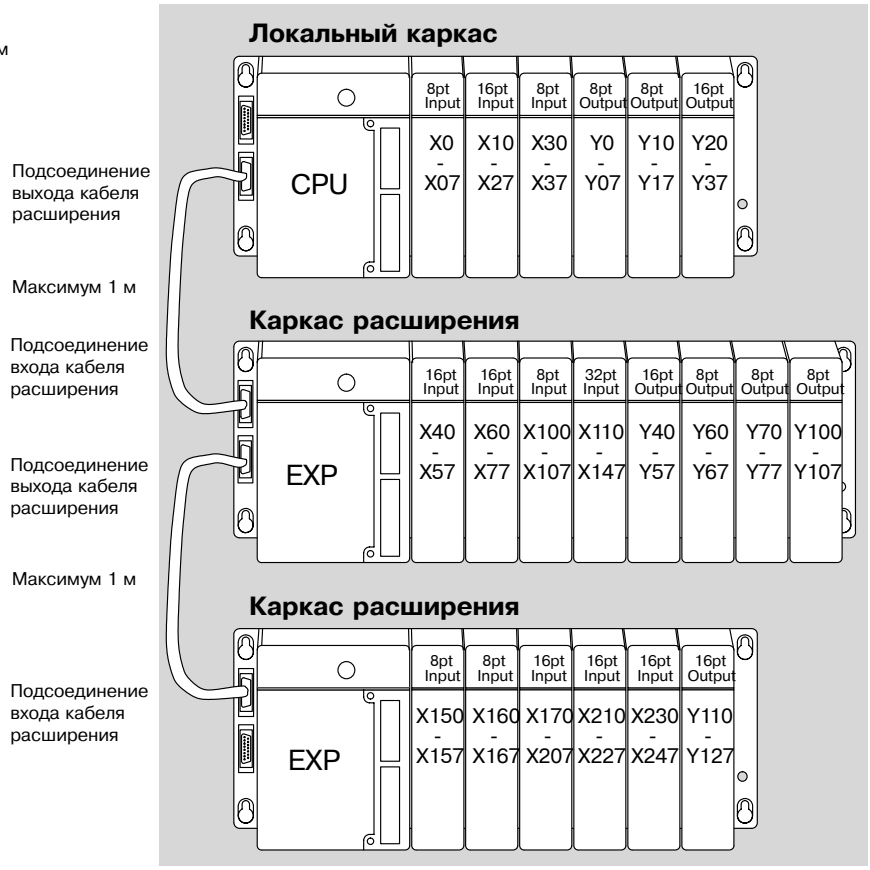


Локальный каркас расширения и расширение ввода/вывода

Используйте локальное расширение, когда вам необходимо большее число точек ввода/вывода или большая мощность, чем может обеспечить локальный каркас. Каркасы расширения требуют вместо процессора локальный блок расширения и кабель (D4-EXCBL-1 или D4-EXCBL-2) для соединения с процессором локального каркаса. На следующем рисунке показан один каркас процессора, два каркаса расширения и нумерация ввода/вывода

DL430/440: поддерживает максимум 3 каркаса расширения и максимум 320 входных точек и 320 выходных точек (включая точки ввода/вывода локального каркаса)

DL450: поддерживает максимум 3 каркаса расширения и максимум 1024 входные точки и 1024 выходные точки (включая точки ввода/вывода локального каркаса)



Расширение удаленного ввода/вывода

Как добавить каналы удаленного ввода/вывода

√	√	√
430	440	450

Удаленный ввод/вывод применяется для систем, в которых имеется достаточное число датчиков и других полевых устройств, находящихся на достаточно большом расстоянии (до 1000м) от центрального процессора. Подключение удаленных точек ввода/вывода производится следующим образом:

- Процессоры DL430/DL440: Удаленный ввод/вывод требует установки ведущего модуля (D4-RM) в центральном каркасе. Процессор обновляет данные в ведущем модуле, который затем управляет передачей данных в каркас удаленного ввода/вывода и из него через линии связи с ведомыми модулями удаленного ввода/вывода (D4-RS), которые расположены в каждом удаленном каркасе.
- **Процессор DL450:** Коммуникационный порт 3 процессора имеет встроенный канал удаленного ввода/вывода. Кроме того, вы можете использовать один или два ведущих модуля удаленного ввода/вывода D4-RM в центральном каркасе, как это описано выше (вы можете использовать либо один из этих способов, либо оба).

	DL430	DL440	DL450
Максимальное число Ведущих модулей удаленного ввода/вывода, поддерживаемых в центральном каркасе процессора (1 канал на ведущий модуль)	2	2	2
Число встроенных каналов удаленного ввода/вывода процессора	нет	нет	1
Максимальное число точек ввода/вывода, поддерживаемых каждым каналом	512	512	512
Максимальное число поддерживаемых точек удаленного ввода/вывода	512	1024	1536
Максимальное число каркасов удаленного ввода/вывода на один канал	7	7	7

Использование удаленного ввода/вывода не ограничивает применение расширения локального ввода/вывода, которое рассматривалось в предыдущем разделе. Фактически нумерация точек удаленного ввода/вывода производится независимо. Обновление удаленного ввода/вывода в зависимости от времени сканирования процессора может производится медленнее, чем точек локального ввода/вывода и его расширения из-за наличия последовательных передач.

Точки удаленного ввода/вывода имеют свои собственные ячейки памяти процессора, отличные от ячеек памяти локального ввода/вывода и его расширения Поэтому добавление точек удаленного ввода/вывода не уменьшает числа точек локального ввода/вывода. Обратитесь к руководству по подсистеме удаленного ввода/вывода DL405 для получения детальной информации по конфигурированию и нумерации удаленных точек ввода/вывода.

На следующем рисунке показан 1 каркас процессора и 1 канал удаленного ввода/вывода с семью удаленными каркасами. Если это процессор DL450, то первый канал удаленного ввода/вывода не требует установки ведущего модуля удаленного ввода/вывода (используется встроенный канал порта 3).

Конфигурирование канала удаленного ввода/вывода

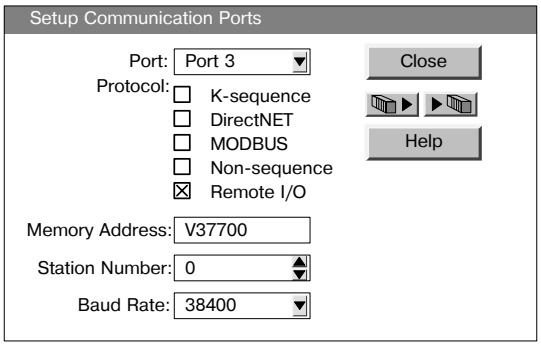
X	X	√
430	440	450

В данном разделе описывается, как сконфигурировать встроенный канал удаленного ввода/вывода DL450. Дополнительная информация содержится в руководстве по удаленному вводу/выводу, D4-REMIO-M. Это руководство понадобится вам при конфигурировании ведомых блоков удаленного ввода/вывода при работе в сети. Руководством D4-REMIO-M вы можете пользоваться только тогда, когда постоянно применяете в системе DL405 ведущие и ведомые устройства для удаленного ввода/вывода.

Встроенный канал удаленного ввода/вывода процессора DL450 имеет те же функциональные возможности, что и ведущий модуль удаленного ввода/вывода D4-RM. В частности, он может взаимодействовать с семью удаленными каркасами, включающими максимум 512 точек ввода/вывода с максимальным расстоянием 1000 м. При необходимости вы можете дополнительно использовать ведущие модули удаленного ввода/вывода в центральном каркасе процессора (по 512 точек ввода/вывода на каждый канал), что в сумме по трем каналам обеспечивает общее число 1536 точек удаленного ввода/вывода. Но сначала вам необходимо отладить коммуникации с удаленным вводом/выводом.

В спецификациях процессоров в главе 3 указывалось, что порт 3 DL450 поддерживает несколько протоколов. Для конфигурации этого порта с помощью ручного программатора используйте AUX 56 и выполняйте действия, указанные ниже. Для конфигурации этого порта с помощью DirectSOFT войдите в меню ПЛК, далее выберите Setup, затем - Setup Secondary Comm Port...

- **Порт (Port):** Из списка номеров портов в верхней части окна выберите "Port 3".
- **Протокол (Protocol):** Нажмите на метку "Remote I/O" в левой части окна (на ручном программаторе она называется "M-NET") и вы увидите следующее окно.



- **Адрес блока памяти (Memory address):** Выберите адрес в V-памяти для использования его в качестве начальной ячейки таблицы конфигурации удаленного ввода/вывода (по умолчанию это - V37700). Эта таблица расположена отдельно, она независима от соответствующей таблицы для ведущего(их) устройства удаленного ввода/вывода системы.
- **Номер станции (Station Number):** В качестве номера станции выберите "0", что сделает DL450 ведущим устройством. Номера станций 1 - 7 зарезервированы за ведомыми устройствами удаленного ввода/вывода.
- **Скорость передачи в бодах (Baud Rate):** Доступны скорости передачи 19200 и 38400 бод. Сначала для удаленного ввода/вывода выберите скорость 38400 бод, вернитесь к скорости 19200 бод, если при опытной передаче возникают ошибки в данных или помехи в линии.

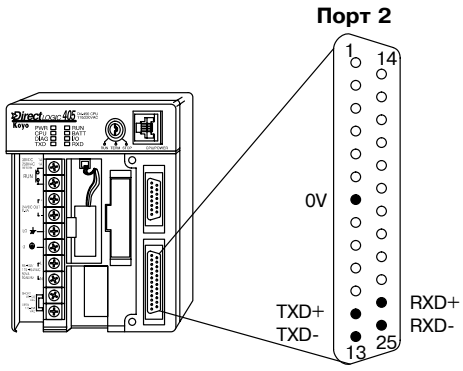
Важно: Вы должны установить скорость передачи для ведомых устройств удаленного ввода/вывода (через переключатели DIP) таким образом, чтобы согласовать все скорости передачи для порта 3 процессора.

Затем нажмите клавишу, указывающую на запоминание конфигурации порта 3 процессором, и нажмите "Close".

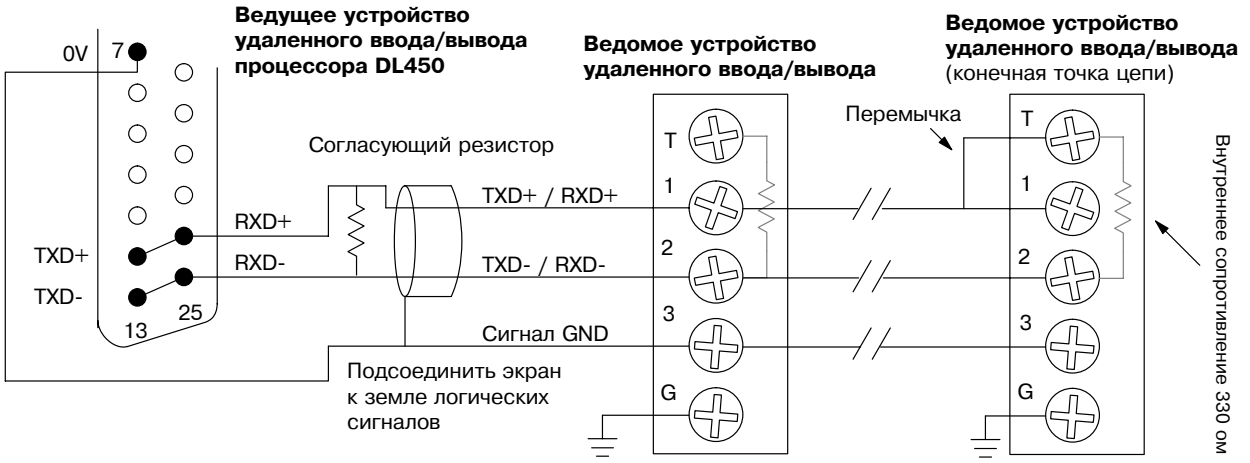
Для подсоединения всех устройств на линии удаленного ввода/вывода необходимо выполнить следующий этап.

Порт 3 DL450 имеет 25-контактный разъем, показанный на рисунке справа. Напоминаем, что порты 1 и 3 - это "логические" порты, которые совместно используют этот 25-контактный разъем. Контакты имеют следующее назначение:

- Контакт 7 Сигнал GND ("Земля")
- Контакт 12 TXD+
- Контакт 13 TXD-
- Контакт 24 RXD+
- Контакт 25 RXD-_

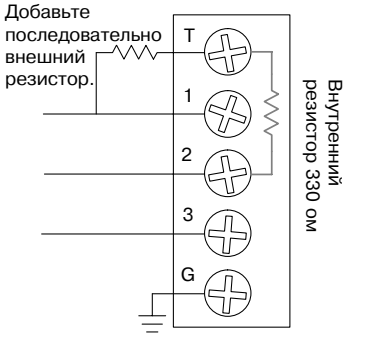


Далее рассмотрим подсоединение DL450 к удаленным ведомым устройствам в удаленном каркасе(ах). Линия связи с удаленным вводом/выводом является 3-х проводной полудуплексного типа. Поскольку порт 3 процессора DL450 является 5-и проводным дуплексным, то необходимо установить перемычку в схеме передачи порта и получить линию связи, показанную ниже (то есть преобразовать ее в 3-х проводную полудуплексную).



Витая / экранированная пара подсоединяется к порту 3 DL450 как показано на рисунке. Обеспечьте соединение экранированного провода кабеля к заземлению логических сигналов. Согласующий резистор необходимо устанавливать вне процессора, по возможности ближе к контактам разъема. Его назначение - минимизировать электрическое отражение сигналов в очень длинных кабелях. Обеспечьте установку перемычки на последнем ведомом устройстве с подсоединением необходимого внутреннего согласующего сопротивления.

Идеально, когда два согласующих резистора на противоположных концах кабеля и номинальный импеданс кабеля совпадают. Если полные сопротивления кабеля больше 330 ом, то добавьте последовательный резистор в последнее ведомое устройство, как показано справа. Если он меньше 330 ом, то вместо этого установите параллельный согласующий резистор между контактами 1 и 2 ведомого устройства.



Не забудьте определить величину согласующего резистора в порту 3. Значение сопротивления должно быть между 100 и 500 ом.

Конфигурирование ведомых устройств удаленного ввода/вывода

После конфигурирования порта 3 процессора DL450 и подсоединения его к ведомым устройствам удаленного ввода/вывода, используйте следующие контрольную таблицу для завершения конфигурации ведомых устройств удаленного ввода/вывода. Полные инструкции по этим шагам приведены в руководстве по удаленному вводу/выводу.

- Установите с помощью DIP-переключателя скорость передачи в бодах, чтобы согласовать настройку порта 3 процессора.
- Выберите адрес станции для каждого ведомого устройства, с 1 по 7. Каждое устройство в линии удаленной связи должно иметь уникальный адрес станции. Только одно ведущее устройство (с адресом 0) может быть в линии удаленной связи.

С помощью закрепленных в таблице ячеек V-памяти (V7404 - 7477) конфигурируется до 2 каналов ввода/вывода. Для конфигурирования встроенного канала удаленного ввода/вывода процессоров DL450 мы используем отдельную таблицу. Таблица с началом в V7404 потребуется вам еще для конфигурирования ведущих модулей удаленного ввода/вывода.

Конфигурирование с помощью таблицы удаленного ввода/вывода

В начале таблицы конфигурации встроенного канала удаленного ввода/вывода устанавливается адрес памяти, который был выбран при настройке порта 3.

Таблица состоит, как показано на рисунке справа, из блоков по четыре слова, которые соответствуют каждому ведомому устройству в системе. Первые четыре ячейки таблицы резервируются.

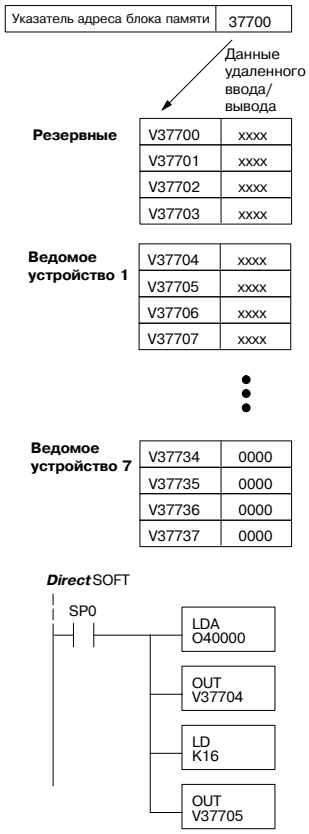
При включении питания процессор считывает данные из этой таблицы, интерпретируя слова этих данных следующим образом:

1. Начальный адрес входных данных ведомого устройства.
2. Число входных точек ведомого устройства.
3. Начальный адрес выходных данных ведомого устройства.
4. Число выходных точек ведомого устройства.

Таблица состоит из 32 слов. Если в системе меньше семи удаленных ведомых каркасов, то оставшиеся ячейки таблицы заполняются нулями. Например, система с 3-мя ведомыми устройствами соответствует таблице конфигурации удаленного ввода/вывода, которая содержит 4 резервных слова, 12 слов данных и 16 слов ”0000”.

При включении питания программное обеспечение должно конфигурировать эту таблицу (только один раз). Используйте команду LDA для загрузки адресов в таблицу, как показано на рисунке справа. Используйте стандартную константу команды LD для загрузки числа входных и выходных точек ведомого устройства.

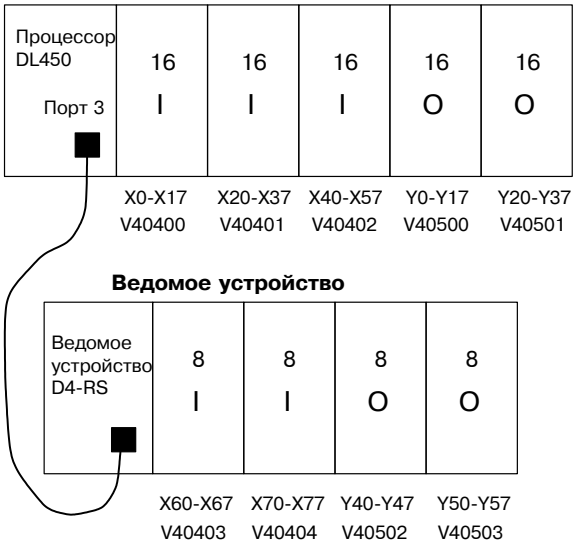
Руководство по D4-REMIO-M содержит исчерпывающие примеры конфигурирования таблицы в V7404, которую вы можете также приспособить для данной таблицы. На следующей страницу приведен пример короткой программы для одного ведомого устройства.



Ниже показан пример простой системы с удаленным вводом/выводом. Встроенный канал удаленного ввода/вывода DL450 подсоединен к одному ведомому каркасу, которому присвоен адрес станции = 1. Скорость в бодах для ведущего и ведомого устройств установлена 38400 бод.

Мы можем отобразить точки удаленного ввода/вывода, как и точки любого ввода/вывода, просто выбирая соответствующую область V-памяти. Напоминаем, что для DL450 доступны типы данных как GX, так и GY. Поскольку мы имеем множество доступных адресов стандартного ввода/вывода, выберем адреса точек удаленного ввода/вывода, которые непосредственно следуют за адресами X и Y точек основного каркаса (X60 и Y40 соответственно).

Основной каркас с процессором в качестве ведущего устройства



Рабочая таблица ведомого устройства удаленного ввода/вывода

Адрес удаленного каркаса 1 (Выбор из 1 - 7)

Номер слота	Обозначение модуля	ВХОД		ВЫХОД	
		Адрес вход	Число входов	Адрес выхода	Число выходов
0	08ND3S	X060	8		
1	08ND3S	X070	8		
2	08TD1			Y040	8
3	08TD1			Y050	8
4					
5					
6					
7					

Input Bit Start Address: X060 V-Memory Address:V 40403

Total Input Points 16

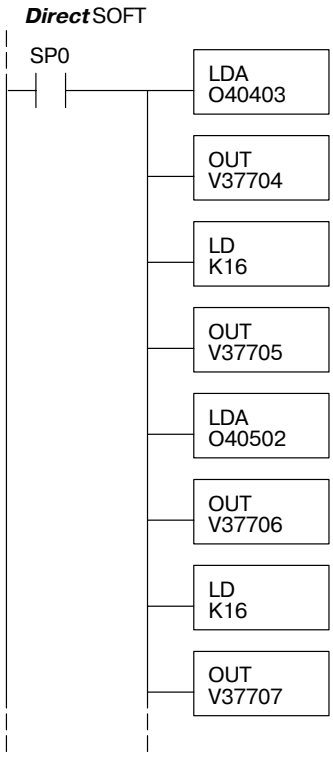
Output Bit Start Address: Y040 V-Memory Address:V 40502

Total Output Points 16

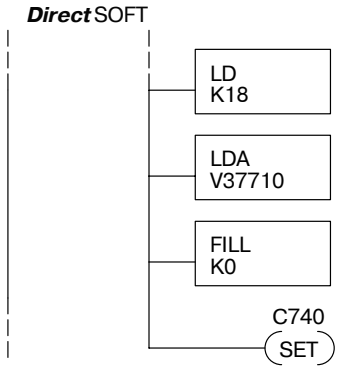
Программа настройки удаленного ввода/вывода

Используя приведенную рабочую таблицу ведомого устройства удаленного ввода/вывода, можно организовать ваши системные данные для подготовки программы релейной логики (незаполненная копия этой рабочей таблицы приведена в Приложении A руководства по D4-REMIO-M). В вашу таблицу конфигурации удаленного ввода/вывода, в ее нижний правый угол, необходимо занести указанные четыре ключевых параметра. Вы можете определить значения адресов, используя карту распределения памяти, приведенную в конце главы 3 "Спецификации и работа процессора".

На рисунке справа показан сегмент программы перевода данных рабочей таблицы в таблицу конфигурации удаленного ввода/вывода. Используйте соответственно команды LDA или LD. В нижней части рисунка показана программа, позволяющая получить такую линию связи для удаленного ввода/вывода и запустить ее.



При конфигурировании канала удаленного ввода/вывода с числом ведомых устройств, меньшим 7, следует заполнить остаток таблицы нулями. Сделать это необходимо, иначе процессор будет интерпретировать любое ненулевое число как информацию о ведомом устройстве.

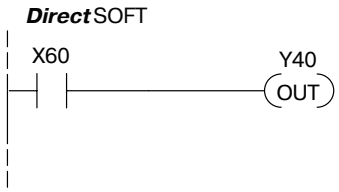


В нашу программу настройки добавим сегмент, который заполняет оставшуюся часть таблицы нулями. На примере справа нулями заполняются ведомые устройства с номерами 2 - 7, которых нет в нашем примере системы (6 каркасов ' 4 = 24 ячейки, = 18 в шестнадцатеричном формате).

В последней цепи приведенной выше программы мы установили контакты специального реле в состояние C740. Этот конкретный контакт указывает, что программа релейной логики процессора завершила спецификацию системы удаленного ввода/вывода. В этот момент процессор начинает обмен данными с удаленным вводом/выводом. Не забудьте включить данный контакт после программы настройки удаленного ввода/вывода.

Программа тестирования удаленного ввода/вывода

На данном этапе можно проверить линии связи удаленного ввода/вывода и результаты работы программы настройки. Простую быструю проверку можно сделать с помощью одной цепочки программы, показанной справа. В ней первый вход удаленного каркаса соединен с первым выходом. После перевода ПЛК в Рабочий режим (RUN) можно перейти к удаленному каркасу и активизировать его первый вход. После этого следует включить его первый выход.



Расширение секционного ввода/вывода

√	√	√
430	440	450

Секционный ввод/вывод подобен удаленному вводу/выводу, поскольку для него также необходим специальный модуль в локальном каркасе процессора, называемый Секционный Ведущий модуль (D4-SM), для управления секционными ведомыми устройствами. Секционное ведомое устройство - это блок с фиксированным числом точек ввода/вывода и коммуникационный порт в одном корпусе.

	DL430	DL440	DL450
Максимальное число секционных ведущих модулей, поддерживаемых в центральном каркасе процессора (число каналов)	2	2	2
Максимальное число секционных ведомых устройств, поддерживаемых одним секционным ведущим модулем	15	15	15
Максимальное число секционных ведомых устройств	30	30	30

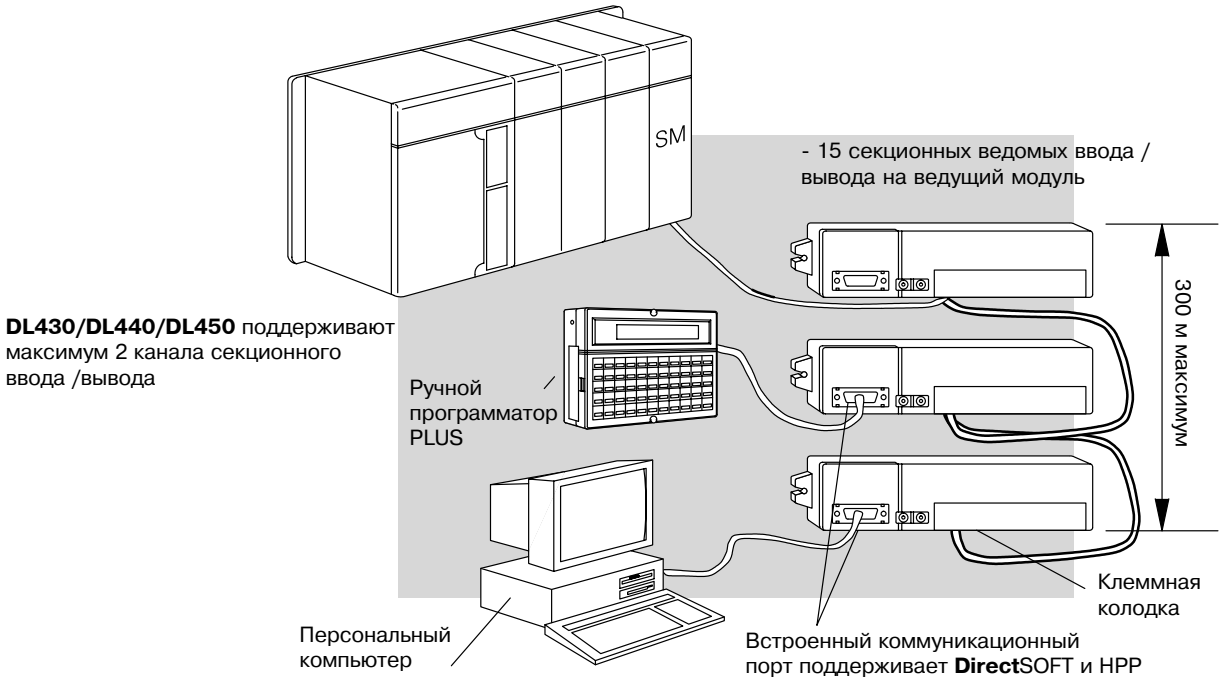
Максимальное расстояние между секционным ведущим модулем и ведомым устройством - 300 м. Возможно, что секционный ввод/вывода не будет обновляться в каждом цикле сканирования процессора из-за дополнительного времени на связь. Эти и другие характеристики рассматриваются в руководстве по секционному вводу/выводу DL405.

Распределение памяти для секционного ввода/вывода

Распределение памяти для секционного ввода/вывода аналогично распределению памяти для удаленного ввода/вывода, поскольку секционный ввод/вывод не использует те же ячейки V-памяти, какие использует локальный ввод/вывод и его расширение. За более детальной информацией по конфигурированию и нумерации секционного ввода/вывода обращайтесь в соответствующее руководство.

Пример секционного ввода/вывода

На следующем рисунке показан один каркас процессора, три секционных ведомых устройства и средства программирования с помощью ручного программатора или DirectSOFT.



Сетевые подключения к MODBUS и DirectNET

Сетевые подключения к MODBUS и DirectNET

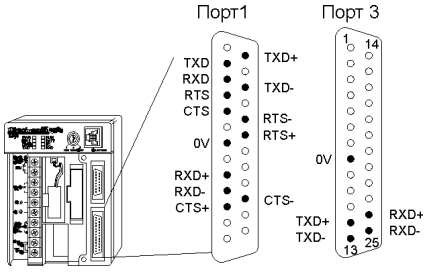
✓	✓	✓
430	440	450

Данный раздел содержит описание конфигурирования встроенных сетевых портов процессора либо под MODBUS, либо под DirectNET. Это даст вам возможность подсоединить систему DL405 ПЛК непосредственно к сети MODBUS через протокол RTU или к другим устройствам в сети DirectNET. Ведущее устройство MODBUS сети должна иметь возможность выдавать команды MODBUS на чтение и запись соответствующих данных. За подробной информацией по протоколу MODBUS обратитесь к Справочному руководству по протоколу Gould MODBUS (P1-MBUS 300 п. B). Перед заказом документации проверьте, что это последнее издание данного руководства. За подробной информацией по DirectNET обратитесь к руководству по DirectNET (DA-DNET-M).

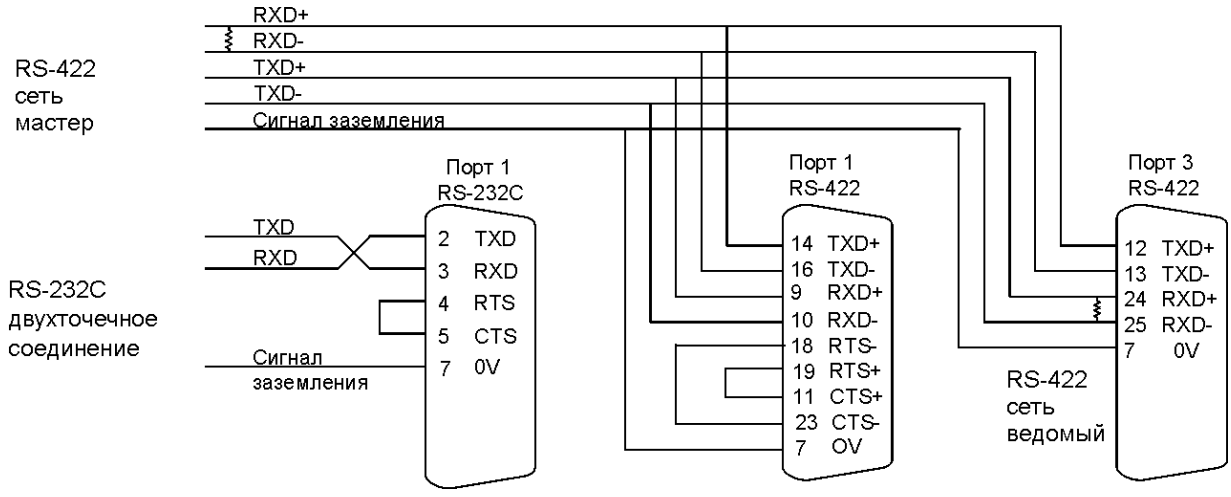
DL430 и DL440 через порт 1 могут быть ведомыми устройствами DirectNET. Порт 1 и порт 3 DL450 могут функционировать как ведущее или ведомое устройства как в MODBUS, так и в DirectNET. В порту 1 доступны уровни сигналов RS-232 и RS-422 на раздельных контактах, а порт 3 (DL450) использует уровни сигнала RS-422.

Порты 1 и 3 DL450 совместно используют 25-контактный разъем типа D, как показано на рисунке справа. Подсоедините один или оба порта как показано ниже. Следует отметить, что Вы не можете одновременно использовать сигналы RS-232 и RS-422 в порту 1.

	Порт1	Порт 3
DL430 и DL440	DirectNET, только как ведомое	Отсутствует
DL450	DirectNET или MODBUS, ведущее/ ведомое	DirectNET или MODBUS, ведущее/ ведомое



Вам необходимо определить, делаете ли вы сетевое подключение через 3-проводный RS-232 или через 5-проводный RS-422. Обычно сигналы RS-232 используются на более коротких расстояниях (максимум 15 метров) для соединения только между двумя устройствами. Сигналы RS-422 применяются на больших расстояниях (максимально 1000 метров) и для многоабонентских сетей (от 2 до 248 устройств). Используйте согласующие резисторы на обоих концах соединения RS-422 с сетью, чтобы согласовать номиналы импедансов кабеля (между 100 и 500 Ом).

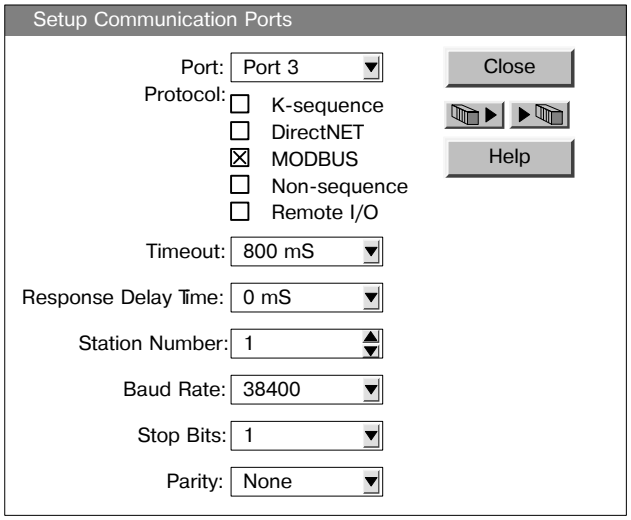


Конфигурирование порта для MODBUS

X	X	√
430	440	450

В DirectSOFT выберите меню ПЛК, далее Setup, затем ”Secondary Comm Port”.

- **Порт (Port):** Из списка номеров портов в верхней части окна выберите ”Port 1 or 3” (Порт 1 или 3).
- **Протокол (Protocol):** Нажмите на метку ”MODBUS” в левой части окна (на Ручном Программаторе используйте AUX 56 и выберите ”MBUS”) и вы увидите следующее окно.



- **Время ожидания (Timeout):** время ожидания порта после посылки им сообщения и до получения реакции перед регистрацией ошибки.
- **Время задержки ответа (Response Delay Time):** время, которое порт ожидает перед посылкой сообщения после его готовности послать его. Для порта 1 это - время активации линии RTS перед началом передачи данных (предполагается, что CTS уже активное). Порт не может передавать данные, если вход CTS ложный.
- **Номер станции (Station Number):** Для того чтобы сделать порт процессора ведущим устройством MODBUSв выберите в качестве номера станции ”1”. Допустимый диапазон номеров станций для ведомых устройством MODBUSв - от 1 до 247, но сетевые команды DL450, используемые в режиме ведущего устройства, допускают для ведомых устройств только номера с 1 по 90. Каждое ведомое устройство должно иметь уникальный номер. При включении питания порт автоматически устанавливается в режим ведомого устройства, который сохраняется до тех пор, пока DL450 не выполнит сетевые команды релейной логики, использующие порт как ведущее устройство. После этого порт возвращается в режим ведомого устройства, пока релейная логика снова не использует этот порт.
- **Скорость передачи в бодах (Baud Rate):** Доступны скорости передачи 300, 600, 900, 2400, 4800, 9600, 19200 и 38400 бод. Сначала выберите наибольшую скорость передачи, возвращаясь к меньшим скоростям, если при опытной передаче имеются ошибки в данных или помехи в линии.

Важно: Вы должны установить одинаковую скорость передачи данных для всех устройств сети. За более подробной информацией обратитесь к соответствующему руководству устройств.

- **Стоповые биты (Stop Bits):** Для использования в протоколе выберите в качестве стоповых битов 1 или 2.
- **Контроль четности (Parity):** Для контроля ошибок выберите ”нет”, ”четное” или ”нечетное”.

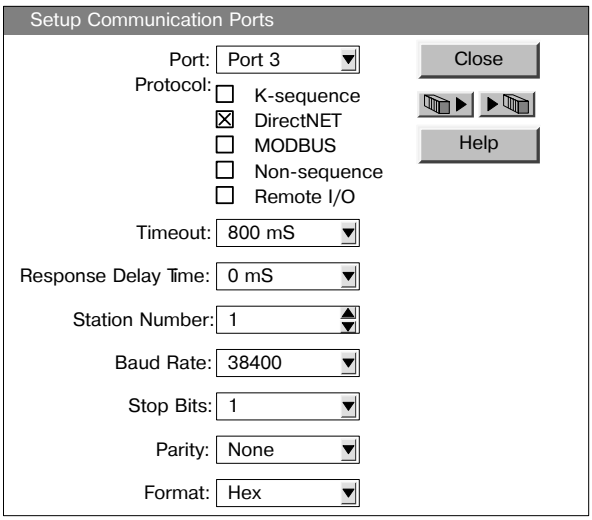
Затем нажмите клавишу, указывающую на запоминание конфигурации порта процессором, и далее ”Close”.

Конфигурирование порта для DirectNET

√	√	√
430	440	450

В DirectSOFT выберите меню ПЛК, далее Setup, затем "Secondary Comm Port".

- **Порт (Port):** Из списка номеров портов в верхней части окна выберите "Port 1 or 3" (Порт 1 или 3).(только для DL450).
- **Протокол (Protocol):** Нажмите на метку "DirectNET" в левой части окна (на ручном программаторе используйте AUX 56 и выберите "DNET") и вы увидите следующее окно.



- **Время ожидания (Timeout):** время ожидания порта после посылки им сообщения и до получения реакции перед регистрацией ошибки.
- **Время задержки ответа (Response Delay Time):** время, которое порт ожидает перед посылкой сообщения после его готовности послать его. Для порта 1 это - время активации линии RTS перед началом передачи данных (предполагается, что CTS уже активное). Порт не может передавать данные, если вход CTS ложный.
- **Номер станции (Station Number):** Для того, чтобы сделать порт Процессора ведущим устройством DirectNET выберите в качестве номера станции "1". Разрешенный диапазон номеров станций для ведомых устройством DirectNET - от 1 до 90 (каждое ведомое устройство должно иметь уникальный номер). При включении питания порт автоматически устанавливается в режим ведомого устройства, который сохраняется до тех пор, пока DL450 не выполнит сетевые команды релейной логики, использующие порт как Ведущее устройство. После этого порт возвращается в режим ведомого устройства, пока релейная логика снова не использует этот порт.
- **Скорость передачи в бодах (Baud Rate):** Доступны скорости передачи 300, 600, 900, 2400, 4800, 9600, 19200 и 38400 бод. Сначала выберите наибольшую скорость передачи, возвращаясь к меньшим скоростям, если при опытной передаче имеются ошибки в данных или помехи в линии.

Важно: Вы должны установить одинаковую скорость передачи данных для всех устройств сети.

- **Стоповые биты (Stop Bits):** Для использования в протоколе выберите в качестве стоповых битов 1 или 2.
- **Контроль четности (Parity):** Для контроля ошибок выберите "нет", "четное" или "нечетное".
- **Формат (Format):** Выберите либо шестнадцатеричный, либо формат ASCII.

Затем нажмите клавишу, указывающую на запоминание конфигурации порта процессором, и нажмите "Close".

Функционирование ведомого устройства в сети

√	√	√
430	440	450

В данном разделе рассматривается, каким образом другие устройства в сети могут обмениваться информацией через порт процессора, который сконфигурирован вами как ведомое устройство DirectNETSlave или MODBUSSlave (DL450). Ведущая машина MODBUS должна использовать протокол MODBUS RTU для установления связи с DL450 в качестве ведомого устройства. Программы этой ведущей машины должны посылать код функции MODBUS и адрес MODBUS для определения ячейки памяти ПЛК, которые DL450 понимает. Ведущая машина DirectNET использует обычные адреса для доступа к процессору DL405 и к системе. Не требуется никакой релейной логики процессора для поддержания функционирования MODBUS или DirectNET в качестве ведомого устройства.

Коды поддерживаемых функций MODBUS

X	X	√
430	440	450

Коды функций MODBUS определяют, получен ли доступ для чтения или для записи, а также получен ли доступ к одной точке данных, либо для их группы. Ниже приводятся коды функций MODBUS, поддерживаемых DL450.

Коды функций MODBUS	Функция	Доступные типы данных DL405
01	Чтение группы обмоток	Y, CR, T , CT, GY
02	Чтение группы входов	X, SP , GX
05	Установка/сброс одной обмотки	Y, CR, T , CT
15	Установка/сброс группы обмоток	Y, CR, T , CT
03, 04	Чтение значения с одного или большего числа регистров	V
06	Запись значения в один регистр	V
16	Запись значения в группу регистров	V

Типы поддерживаемых данных MODBUS

Типы памяти в системе DL405 включают входы X, выходы Y, управляющие реле C, регистры V-памяти и др. MODBUS использует типы данных с другими именами. Поэтому вам необходимо определить, какой тип данных MODBUS соответствует ячейке памяти ПЛК. Это делается с помощью приведенной ниже таблицы перекрестных ссылок.

Тип памяти DL450	Кол-во (десятичное)	Диапазон ПЛК (восьмеричный)	Соответствующий тип данных MODBUS	Код функции RX
Входы X	1024	X0 - X1777	Вход	02
Глобальные входы (GX)	1536	GX0 - GX2777	Вход	02
Специальные Реле (SP)	512	SP0 - SP137 SP320 - SP717	Вход	02
Выходы (Y)	1024	Y0 - Y1777	Обмотка	01
Глобальные выходы (GY)	1536	GY0 - GY2777	Обмотка	01
Управляющие Реле (CR)	2048	C0 - C3777	Обмотка	01
Контакты Таймера (T)	256	T0 - T377	Обмотка	01
Контакты Счетчика (CT)	256	CT0 - CT377	Обмотка	01
Биты состояния стадии (S)	1024	S0 - S1777	Обмотка	01
Текущие значения Таймера (V)	256	V0 - V377	Регистр входа	03
Текущие значения Счетчика (V)	256	V1000 - V1377	Регистр входа	03
V -память, данные пользователя	3072 12288	V1400 - V7377 V10000 - V37777	Регистр входа	03
V -память, системные данные (V)	320	V700 - V777 V7400 - V7777	Регистр входа	03

Определение адресов MODBUS

Обычно существуют два способа, с помощью которых большинство соглашений по базовому программному обеспечению позволяет специфицировать ячейку памяти ПЛК. Ими являются:

- Указание типа данных и адреса MODBUS,
- Указание только адреса MODBUS.

Если ваше базовое программное обеспечение требует тип данных и адрес

Многие базовые программные пакеты позволяют определить тип данных MODBUS и адрес MODBUS, которые соответствуют ячейке памяти ПЛК. Это наиболее легкий способ, но не все пакеты имеют такую возможность. Различные типы данных MODBUS присутствовали ранее, но они снова включены в следующую таблицу.

Уравнение, применяемое для вычисления адреса, учитывает тип данных ПЛК, который вы используете. В этих целях типы данных ПЛК разделяются на две категории.

- Дискретные - X, SP, Y, CR, S, T, C (контакты)
- Слова - V, текущее значение таймера, текущее значение счетчика

В любом случае вы по существу преобразуете восьмеричный адрес ПЛК в десятичный и добавляете соответствующий адрес MODBUS (если необходимо). В таблице ниже приведено точное уравнение, используемое для каждой группы данных.

Тип памяти DL450	Кол-во (десят)	Диапазон ПЛК (восьмеричный)	Диапазон адресов MODBUS (десятичный)	Тип данных MODBUS
Для Дискретного типа данных ..... Преобразовать Адрес ПЛК в десятичный + Начало диапазона + Тип данных				
Входы X	1024	X0 - X1777	2048 - 3071	Вход
Специальные Реле (SP)	512	SP0 - SP137 SP320 - SP717	3072 - 3167 3280 - 3535	Вход
Выходы (Y)	1024	Y0 - Y1777	2048 - 3071	Обмотка
Управляющие Реле (CR)	2048	C0 - C3777	3072 - 5119	Обмотка
Контакты Таймера (T)	256	T0 - T377	6144 - 6399	Обмотка
Контакты Счетчика (CT)	256	CT0 - CT377	6400 - 6655	Обмотка
Биты состояния стадии (S)	1024	S0 - S1777	5120 - 6143	Обмотка
Глобальные входы (GX)*	1536	GX0 - GX2777	0 - 1535	Вход
Глобальные выходы (GY)*	1536	GY0 - GY2777	0 - 1535	Обмотка
Для типов данных "Слова" .... Преобразовать Адрес ПЛК в десятичный + Тип данных				
Текущие значения Таймера (V)	256	V0 - V377	0 - 255	Регистр входа
Текущие значения Счетчика (V)	256	V1000 - V1377	512 - 767	Регистр входа
V-память, данные пользователя	3072 12288	V1400 - V7377 V10000 - V37777	768 - 3839 4096 - 16383	Регистр хранения
V-память, системные данные (V)	320	V700 - V777 V7400 - V7777	448 - 768 3480 - 3735	Регистр хранения

* ***Примечание.*** *Общее число глобальных точек ввода/вывода типов GX и Gyne может превышать 1536.*

На следующих примерах показано, как сформировать адрес и тип данных MODBUS для базового программного обеспечения, в котором требуется этот формат.

Пример 1: V2100

Найти адрес MODBUS для Пользователя, записанного в ячейке V2100 V-памяти.

1. Найти V-память в таблице.
2. Преобразовать V2100 в
десятичный код (1088).
3. Применить тип данных MODBUS
из таблицы.

Адрес ПЛК (десятич.)+ Тип данных
V2100 = 1088 (десятичных)

1088 + Регистр хранения = **Регистр хранения 1088**

V-память, пользовательские данные (V)	**3072 12288**	**V1400 - V7377 V10000 - V17777**	**768 - 3839 4096 - 16383**	Регистр хранения

Пример 2: Y20

Адрес ПЛК (десятич.) + Начальный Адрес + Тип данных

Найти адрес MODBUS для выхода Y20.

1. Найти выходы Y в таблице.
2. Преобразовать Y20 в десятичный код (16).
3. Добавить начальный адрес диапазона (2048)
4. Применить тип данных MODBUS из таблицы.

Y20 = 16 десятичных
16 + 2048 + Обмотка = Обмотка 2064

Выходы (Y)	**1024**	**Y0 - Y1777**	**2048 - 3071**	**Обмотка**

Пример 3: Текущее значение T10

Адрес ПЛК (десятич.) + Тип данных

Найти адрес MODBUS для получения текущего значения Таймера .T10

1. Найти Текущие Значения Таймера T10.
2. Преобразовать T10 в десятичный код (8).
3. Применить тип данных MODBUS из таблицы.

T10 = 8 десятичных
8 + Регистр входов = **Регистр входов 8**

Текущие значения Таймера (Y)	256	V0 - V377	0 - 255	Регистр входов

Пример 4: C54

Адрес ПЛК (десятич.) + Начальный Адрес + Тип данных

Найти адрес MODBUS для Управляющего Реле C54

1. Найти Управляющее Реле в таблице..
2. Преобразовать C54 в десятичный код (44).
3. Добавить начальный адрес диапазона (3072)
4. Применить тип данных MODBUS из таблицы.

C54 = 44 десятичных
44 + 3072 + Обмотка = **Обмотка 3116**

Управляющие Реле(CR)	2048	C0 - C3777	3072 - 5119	Обмотка

Если ваше базовое программное обеспечение MODBUS требует ТОЛЬКО адрес

Некоторые базовые программы не позволяют указывать тип данных и адрес MODBUS. Они могут указать только адрес. Эта ситуация требует других шагов при указании адреса, но они также достаточно просты. По существу MODBUS также разделяет типы данных по диапазонам адресов. Отсюда следует, что только адрес описывает тип данных и ячейку памяти. Это часто называют "добавление смещения". Важно запомнить, что в вашем базовом программном обеспечении доступны два различных режима адресации:

- Режим 484
- Режим 584/984

Мы рекомендуем вам использовать режим адресации 584/984, если ваше базовое программное обеспечение это позволяет. Режим 584/984 позволяет иметь доступ к большему числу ячеек памяти для каждого типа данных. Если ваше программное обеспечение поддерживает только режим 484, то некоторые ячейки памяти ПЛК могут быть недоступны.

Уравнение, применяемое для вычисления адреса, учитывает тип данных, который вы используете. В этих целях типы данных ПЛК разделяются на две категории.

- Дискретные - X, GX, SP, Y, CR, S, T, C (контакты)
- Слова - V, Текущее значение таймера, Текущее значение счетчика

В любом случае вы по существу преобразуете восьмеричный адрес ПЛК в десятичный и добавляете соответствующий адрес MODBUS (если необходимо). В таблице ниже приведено строгое уравнение, используемое для каждой группы данных.

Тип памяти DL250	Кол-во (десят)	Диапазон ПЛК (восьмерич)	Диапазон адресов MODBUS (десятич)	Адрес режима 484	Адрес режима 584/984	Тип данных MODBUS
Для Дискретного типа данных	Преобразовать Адрес ПЛК в десятичный + Начало диапазона + Адрес соответствующего режима					
Входы X	512	X0 – X777	2048 – 2560	1001	10001	Входные
Специальные Реле (SP)	512	SP0 – SP137 SP320 – SP717	3072 – 3167 3280 – 3535	1001	10001	Входные
Выходы (Y)	512	Y0 – Y777	2048 – 2560	1	1	Обмотка
Управляющие Реле (CR)	1024	C0 – C3777	3072 – 4095	1	1	Обмотка
Контакты Таймера (T)	256	T0 – T377	6144 – 6399	1	1	Обмотка
Контакты Счетчика (CT)	128	CT0 – CT177	6400 – 6527	1	1	Обмотка
Биты состояния Stage (S)	1024	S0 – S1777	5120 – 6143	1	1	Обмотка
Для типов данных "Слова" .... Преобразовать Адрес ПЛК в десятичный + Адрес соответствующего режима						
Текущие значения Таймера (V)	256	V0 – V377	0 – 255	3001	30001	Регистр входов
Текущие значения Счетчика (V)	128	V1000 – V1 177	512 – 639	3001	30001	Регистр входов
V -память, данные пользователя	3072 4096	V1400 – V7377 V10000 – V17777	768 – 3839 4096 – 8192	4001	40001	Регистр хранения
V -память, данные пользователя	320	V700 – V777 V7400 – V7777	448 – 768 3840 – 3735	4001	40001	Регистр хранения

На следующих примерах показано, как сформировать адрес MODBUS для базового программного обеспечения, которое требует этот формат.

Пример 1: V2100 Режим 584/984

Найти адрес MODBUS для Пользователя, записанного в ячейке V2100 V-памяти.

1. Найти V-память в таблице.
2. Преобразовать V2100 в десятичный код (1088).
3. Применить тип данных MODBUS для режима (40001).

Адрес ПЛК (десятич.)+ Адрес режима
V2100 = 1088 (десятичных)
1088 + 40001 = **41089**

V-память, системные данные (V)	320	V1400 - V7377 V10000 - V17777	448 - 768 3840 - 3735	4001	40001	Регистр хранения

Пример 2: Y20 режим 584/984

Адрес ПЛК (десятич.) + Начальный адрес + Режим

Найти адрес MODBUS для выхода Y20.

1. Найти выходы Y в таблице.
2. Преобразовать Y20 в десятичный код (16).
3. Добавить начальный адрес диапазона (2048)
4. Добавить адрес MODBUS для режима (1).

Y20 = 16 десятичных
16 + 2048 + 1 = **2065**

Выходы (Y)	1024	Y0 – Y1777	2048 – 3071	1	1	Обмотка

Пример 3: Текущее значение T10 режим 484

Найти адрес MODBUS для получения текущего значения таймера T10.

1. Найти текущие значения таймера T10.
2. Преобразовать T10 в десятичный код (8).
3. Добавить начальный адрес MODBUS для режима (3001).

Адрес ПЛК (десятич.) + Адрес режима
T10 = 8 десятичных
8 + 3001 = **3009**

Текущие значения таймера (Y)	256	V0 – V377	0 – 255	3001	30001	Регистр входов

Пример 4: C54 режим 584/984

Адрес ПЛК (десятич.) + Начальный адрес + Режим

Найти адрес MODBUS для управляющего реле C54.

1. Найти управляющее реле в таблице.
2. Преобразовать C54 в десятичный код (44).
3. Добавить начальный адрес диапазона (3072)
4. Добавить адрес MODBUS для режима (1).

C54 = 44 десятичных
44 + 3072 + 1 = **3117**

Управляющие реле(CR)	2048	C0 – C3777	3072 – 5119	1	1	Обмотка

Определение адреса DirectNET

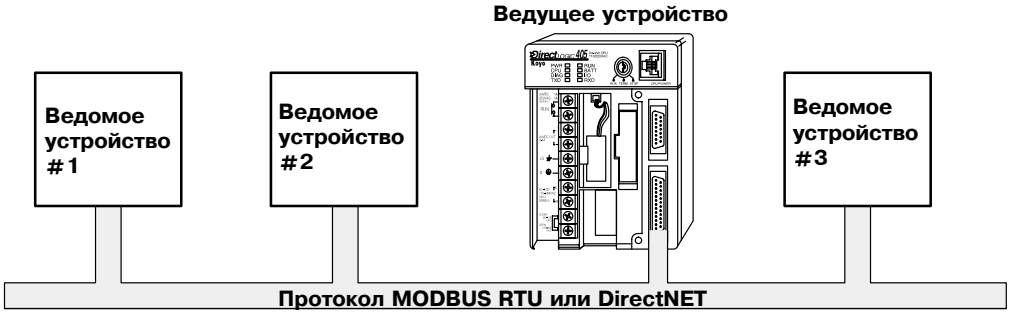
√	√	√
430	440	450

Адресация типов памяти для ведомых устройств DirectNET довольно проста. Просто используйте сам собственный адрес ведомого устройства. Например, чтобы получить доступ к ячейке памяти V1400 ведомого устройства ПЛК через DirectNET, ведущее устройство сети должно запросить из ведомого устройства V1400.

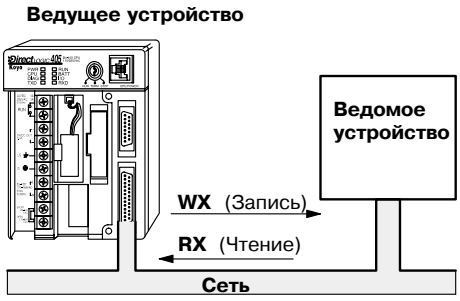
Функционирование ведущего устройства в сети

X	X	√
430	440	450

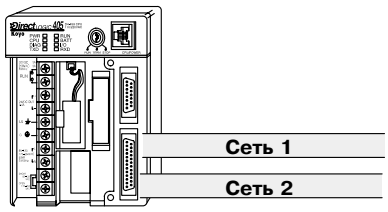
В данном разделе описывается, как DL450 может взаимодействовать с сетью MODBUS или DirectNET в качестве ведущего устройства. Для сетей MODBUS используется протокол MODBUS RTU, который должен интерпретироваться всеми ведомыми устройствами в сети. Как MODBUS, так и DirectNET имеют в сети единственное ведущее устройство и множество ведомых устройств. Ведущим устройством является только такой элемент сети, который может инициировать запросы в сети. В данном разделе показывается, как вы можете создать необходимую релейную логику для работы ведущего устройства сети.



При использовании процессора DL450 в качестве ведущей станции применяйте простые команды RLL для инициализации запросов. Команда WX инициирует сетевые операции записи, а команда RX инициирует сетевые операции чтения. Перед выполнением этих команд вам необходимо загрузить данные, относящиеся к операции чтения или записи в стек аккумулятора процессора. При выполнении команды WX или RX процессор использует информацию в этом стеке наряду с данными поля команды, чтобы полностью определить задачу, которая выдается порту.



Можно использовать оба порта 1 и 3 как для MODBUS, так и для DirectNET, применять любой из них или оба как ведущие устройства. Вы должны указывать командам WX и RX назначенный порт для каждой операции связи.

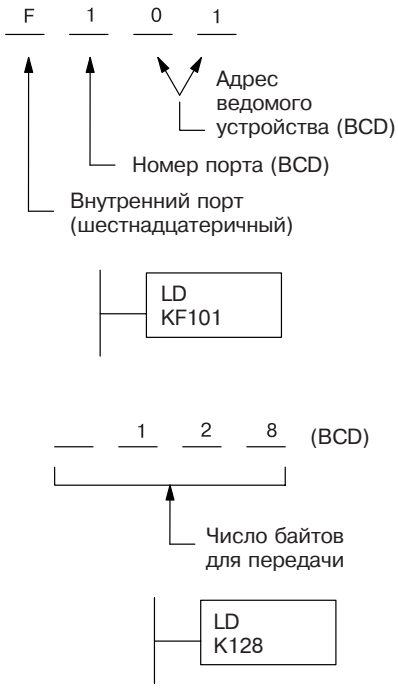


В итоге, команды RLL определяют следующие элементы:

1. Номер порта на ведущем устройстве (порт 1 или 3) и адрес ведомой станции. (Команда LD).
2. Объем данных (в байтах), которые Вы хотите передать. (Команда LD).
3. Область памяти, используемую ведущим устройством. (Команда LDA).
4. Область V-памяти процессора, используемую для связи с ведомыми устройствами, а также является ли операция чтением или записью. (Команда WX или RX).
5. Блокировки на время связи при многократных WX и RX операциях.

Шаг 1: Определить номер порта ведущего устройства и номер ведомого устройства

Первая команда Загрузки (LD) задает номер коммуникационного порта ведущего устройства сети (DL450) и адрес ведомой станции. Эта команда может адресовать до 90 ведомых устройств MODBUS или до 90 ведомых устройств DirectNET. Справа показан формат слова. "F" в старшем полубайте указывает на признак внутреннего порта в процессоре (но не в слоте каркаса). Второй полубайт содержит номер порта, 1 или 3. Нижний байт содержит число с адресом ведомого устройства в двоично-десятичном формате (от 01 до 90).



Шаг 2: Загрузить число байтов для передачи

Вторая команда Загрузки (LD) определяет число байтов, которые необходимо передать между ведущим и ведомым устройствами в последующей команде WX или RX. Загружаемое значение имеет двоично-десятичный формат (десятичный) с 1 до 128 байт.

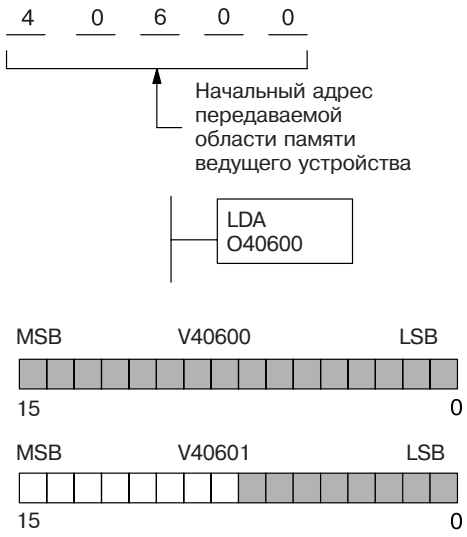
Число задаваемых байтов зависит от типа данных, который вы хотите получить. Например, входные точки DL405 могут выбираться как ячейки V-памяти или как ячейки входов X. Но если вы желаете только X0 - X27, то вы должны использовать тип данных "входы X", так как ячейки V-памяти могут выбираться только с шагом в 2 байта. В следующей таблице приводятся диапазоны байтов для различных типов продуктов DirectLOGICTM

Память DL205 / 405	Битов в блоке	Байтов
V-память	16	2
Текущее значение T / C	16	2
Входы (X, SP)	8	1
Выходы (биты Y, C, Stage, T/C)	8	1
Рабочая Память Клавиатуры	8	1
Диагностическое Состояние	8	1

Память DL305	Битов в блоке	Байтов
Регистры данных	8	1
Накапливающий сумматор T / C	16	2
Биты Ввода/Вывода, внутренних реле, регистр со сдвигами, биты T/C биты Stage	1	1
Рабочая Память Клавиатуры	8	2
Диагностическое Состояние (5 слов R/W)	16	10

Шаг 3: Определить область памяти ведущего устройства

Третьей командой в последовательности RX или WX является команда Загрузить Адрес (LDA). Ее назначение - загрузить начальный адрес области памяти, которая должна быть передана. Получая на входе восьмеричное число, команда LDA преобразует его в шестнадцатеричное число, а результат помещает в аккумулятор.



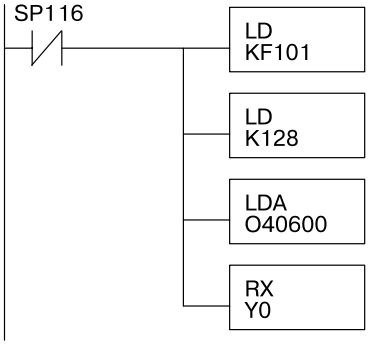
По команде WX процессор DL450 посылает предварительно определенное число байтов из области памяти, начиная с определенного LDA адреса

По команде RX процессор DL450 считывает предварительно определенное число байтов из ведомого устройства, помещает полученные данные в область памяти, начиная с определенного LDA адреса.

ПРИМЕЧАНИЕ. Поскольку слова V-памяти всегда имеют 16 битов, то Вы не всегда можете использовать все слово. Например, если Вы определили только 3 байта и читаете выходы Y с ведомого устройства, то вы желаете получить только 24 бита данных. В этом случае только 8 битов наименьшей значимости ячейки последнего слова будут нести информацию. Остальные 8 битов не имеют значения.

Шаг 4: Определить область памяти ведомого устройства

Последней командой в нашей последовательности является сама команда WX или RX. Используйте WX для записи в ведомое устройство, а RX - для чтения с ведомого устройства. Все четыре наши команды показаны справа. С помощью последней команды вы должны определить начальный адрес и действительный тип данных для ведомого устройства.



Команда RX считывает данные с ведомого устройства, начиная с определенного адреса.

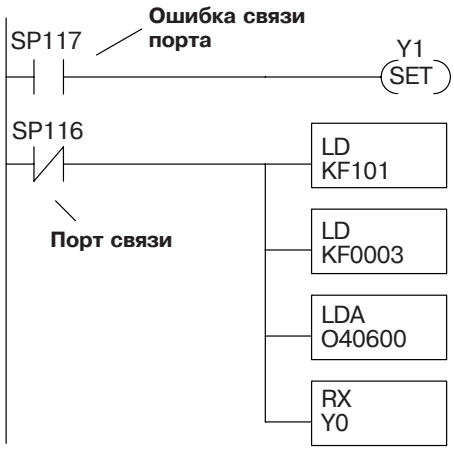
Команда WX записывает данные с ведомого устройства, начиная с определенного адреса.

- Ведомые устройства DirectNET - команды WX и RX используют тот же адрес, что и собственный адрес ввода/вывода ведомого устройства.
- Ведомые устройства MODBUS DL405 или DL205 - команды WX и RX используют тот же адрес, что и собственный адрес ввода/вывода ведомого устройства.
- Ведомые устройства MODBUS DL305 - используют следующую таблицу для преобразования адресов DL305 в адреса MODBUS

Тип памяти Процессора серии DL305 - в - Перекрестную ссылку MODBUS					
Тип памяти ПЛК	Базовый адрес ПЛК	Базовый адрес MODBUS	Тип памяти ПЛК	Базовый адрес ПЛК	Базовый адрес MODBUS
Текущие значения TMR/CNT	R600	V0	Биты состояния TMR/CNT	CT600	GY600
Входные точки Ввода/Вывода	IO 000	GY0	Управляющие Реле	CR160	GY160
Регистры данных	R401, R400	V100	Регистры со сдвигами	SR400	GY400
Биты состояния Stage (только D3-33P)	S0	GY200			

Передачи данных из программы релейной логики

В некоторых приложениях процессор DL450, как ведущее устройство в сети, только периодически взаимодействует с ведомыми устройствами сети. Однако для большинства приложений желательно иметь "непрерывное" обнлвление областей памяти данными, передаваемыми из ведомого в ведущее устройство.



Обычно это означает запуск задачи в каждом цикле СКАНИРОВАНИЯ ПЛК. Однако одна связь по сети WX или RX может длится дольше, чем время одного цикла сканирования ПЛК. Поэтому перед выполнением следующей команды WX или RX мы должны ожидать, пока порт закончит передачу данных по предыдущей команде WX или RX.

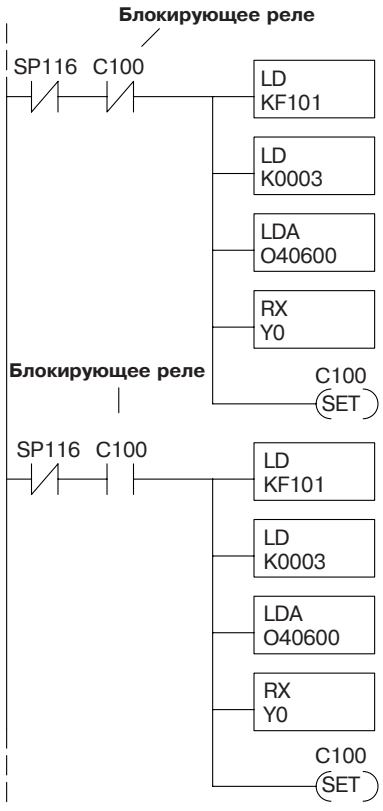
Каждый порт, который может стать ведущим устройством, имеет два связанных с ним контакта специального реле (См. Приложение D по специальным реле коммуникационных портов). Один из них указывает "Порт занят" (SP116), другой указывает "Ошибка связи порта". Приведенный выше пример показывает использование этих контактов для сетевого ведущего устройства, который только считывает с устройства (RX). Контакт "Порт занят" обеспечивает завершение одной сетевой операции перед началом другой.

Использование реле SP "Ошибка связи порта" не обязательно. Когда оно используется, то оно должен находиться в начале процедур связи, поскольку реле ошибки всегда сбрасывается (выключается) при выполнении команд RX или WX, использующих тот же порт.

Блокировки многократного чтения и записи

Если вы применяете многократные чтения и записи в программе RLL, то вы должны взаимно блокировать подпрограммы, чтобы обеспечить их выполнение. Если вы не применяете взаимоблокировки, то процессор выполнит только первую подпрограмму. Это происходит потому, что порт может одновременно управлять только одной транзакцией.

На примере справа после выполнения команды RX устанавливается C0. Когда порт завершает коммуникационную задачу, выполняется вторая подпрограмма и C0 сбрасывается.



Если вы используете Стадийное программирование RLLplus, то вы можете только включить каждую подпрограмму в отдельную стадию программы, чтобы обеспечить ее надлежащее выполнение. В большинстве случаев RLLplus является наиболее эффективным способом создания программ для автоматики.

Руководство по DirectNET содержит примеры ведущего/ведомого устройств с описанием для RLL и для Стадийных программ (они могут быть легко адаптированы для применения с MODBUS).

Стандартные команды RLL

5

В этой главе...

— Введение
— Булевские команды
— Булевские команды сравнения
— Немедленные команды
— Команды регистра сдвига, таймера и счетчика
— Команды загрузки и вывода аккумулятора/стека данных
— Логические команды аккумулятора
— Математические команды
— Команды работы с битами
— Команды преобразования чисел
— Табличные команды
— Команды времени / даты
— Команды управления процессором
— Команды управления программой
— Команды прерывания
— Команды интеллектуального ввода / вывода
— Сетевые команды
— Команды работы с сообщениями

Введение

С помощью команд DL405 можно выполнять различные типы операций. Эта глава показывает, как использовать конкретные команды. В следующей таблице в целях быстрого поиска содержится список мнемонических обозначений команд и номера страниц, на которых эти команды определяются. В каждом определении команды в круглых скобках показаны наборы клавиш на ручном программаторе, используемые для ввода этой команды. Имеются два способа для быстрого поиска нужной команды.

- Если Вы знаете категорию команды (булевские команды, булевские команды сравнения и т.д.), то используйте только заголовок вверху страницы для поиска той страницы, где приведено описание команд этой категории.
- Если Вы знаете индивидуальную мнемонику команды, то используйте следующую таблицу для поиска страницы, где дано описание этой команды.

DL450 обеспечивает все команды, перечисленные в таблице. DL440 обеспечивает подмножество этих команд, DL430 - еще меньшее подмножество. При определении команды указывается, на каких процессорах реализуется эта команда. В примере 1 только DL440 и DL450 имеют данную команду. В примере 2 все процессоры имеют указанную команду.

X	√	√
430	440	450

√	√	√
430	440	450

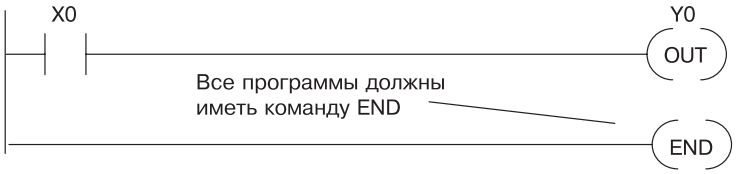
Использование булевских команд

Вы когда-нибудь удивлялись, почему многие производители ПЛК всегда приводят время выполнения 1K булевских операций? Они делают это потому, что большинство программ используют много булевских команд. Это обычно очень простые команды, созданные для соединения входных и выходных контактов в различных последовательных и параллельных комбинациях. Так как пакет Direct SOFTд позволяет использовать графические символы для построения программы, то Вам совершенно не надо знать мнемонику команд. Однако это может быть полезным в некоторых случаях, особенно если Вы должны искать неисправности с помощью ручного программатора.

Следующие параграфы показывают, как эти команды используются для построения простых программ.

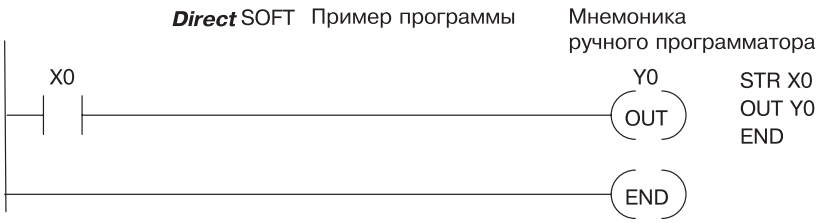
Команда END

Все программы для DL405 требуют, чтобы последней командой была команда END. Эта команда сообщает процессору об окончании программы. Обычно любые команды, находящиеся после команды END, не будут выполняться. Но бывают исключения из этого, например, подпрограммы прерывания и т.п. Это будет обсуждаться подробнее в конце этой главы.



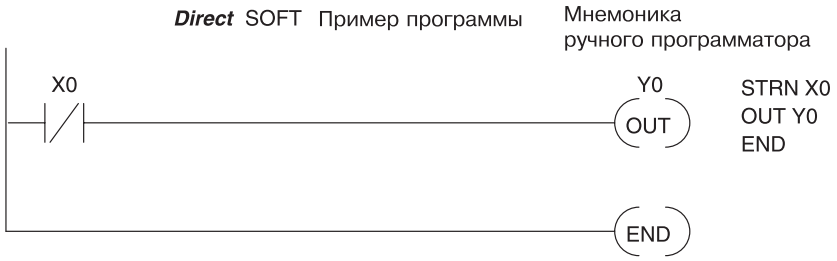
Простая логическая цепь с нормально-открытым контактом

Контакт используется, как начало логической цепи, содержащей контакты и реле. Команда STR (Store) выполняет эту функцию. Конец цепи- обмотка реле формируется командой OUT (Output). Следующий пример показывает, как реализовать эти команды.



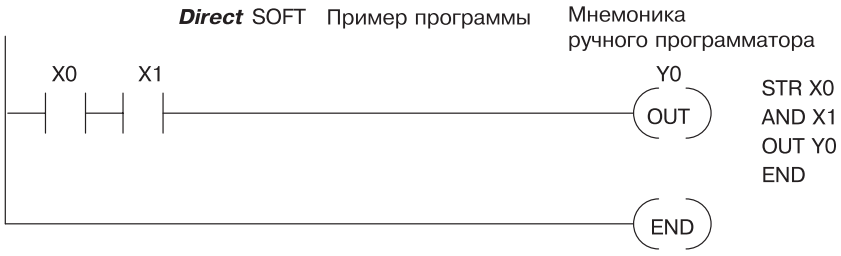
Нормально закрытый контакт

Часто употребляются нормально-закрытые контакты. Для этого используется команда STRN (Store Not). Следующий пример показывает простую логическую цепь с нормально-закрытым контактом.



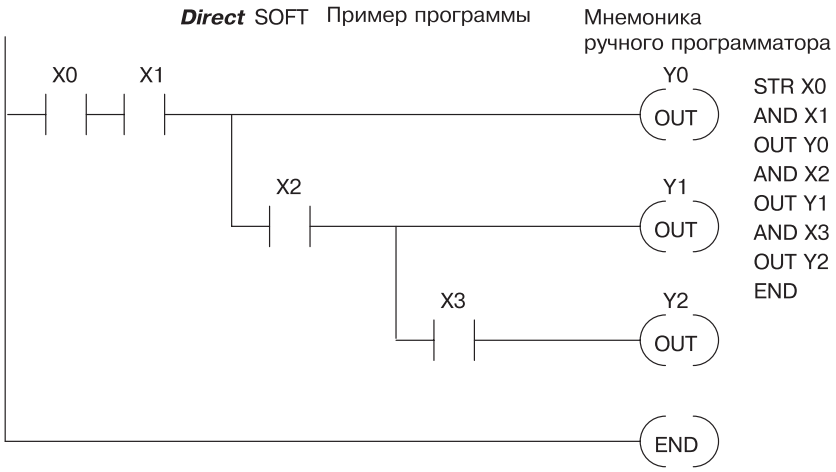
Последовательно соединенные контакты

Чтобы объединить два или более контактов последовательно, используется команда AND. Следующий пример показывает два последовательно соединенных контакта. Используемые команды - STR X0, AND X1, OUT Y0.



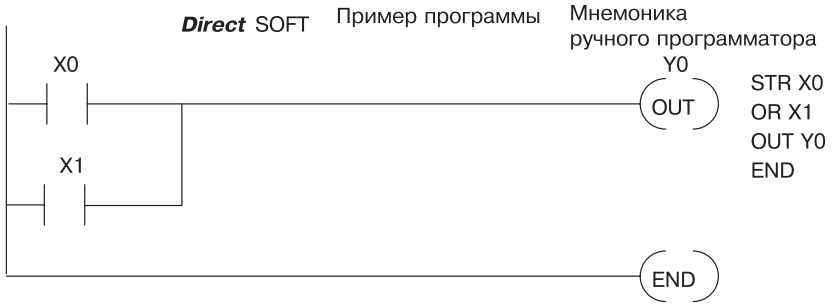
Промежуточные выходы

Иногда необходимо использовать промежуточные выходы, чтобы получить дополнительные выходы, которые зависят от других контактов. Следующий пример показывает, как использовать команду AND, чтобы продолжить логическую цепь с другими дополнительными выходами.



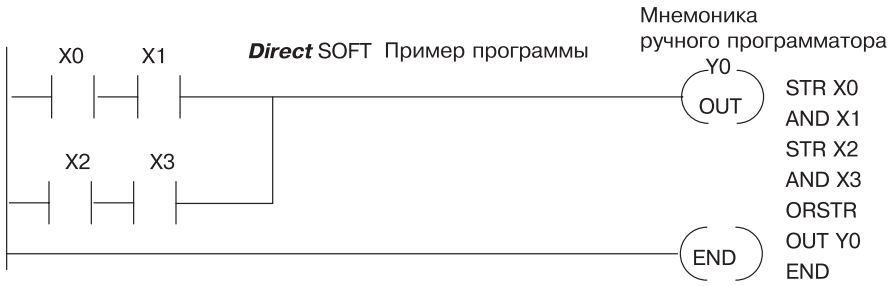
Параллельно соединенные элементы

Контакты могут быть соединены параллельно. Это позволяет реализовать команду OR. Следующий пример показывает два параллельно соединенных контакта. Команды будут такими - STR X0, OR X1, OUT Y0.



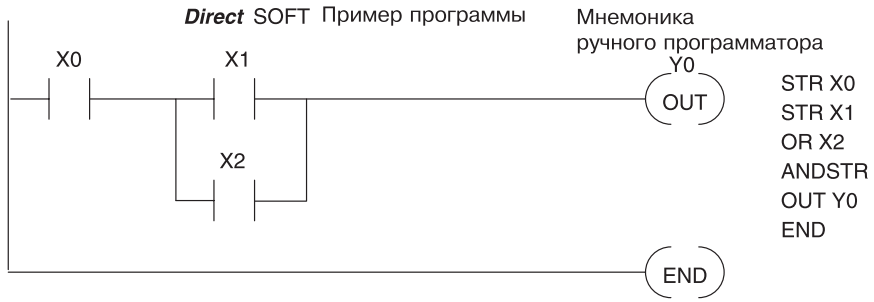
Объединение последовательных цепей параллельно

Достаточно часто необходимо объединить несколько групп последовательных элементов параллельно. Команда ORSTR (Or Store) позволяет делать эту операцию. Следующий пример показывает простую цепь, состоящую из последовательных элементов, объединенных параллельно.



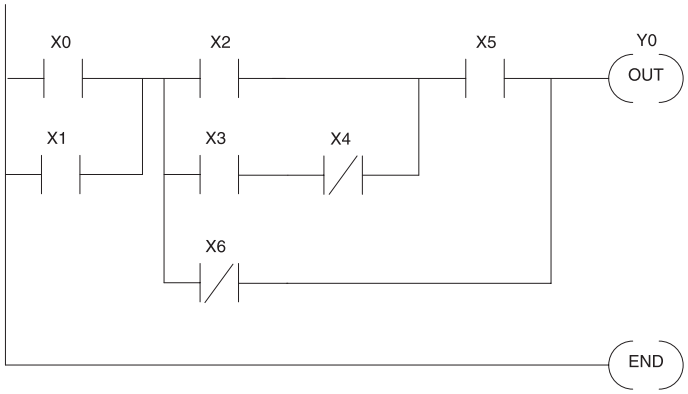
Объединение параллельных цепей последовательно

Вы также можете объединить одну или более параллельных цепей последовательно. Команда ANDSTR (And Store) позволяет делать эту операцию. Следующий пример показывает простую цепь, состоящую из контакта, соединенного последовательно с параллельно объединенными контактами,.



Комбинации цепей

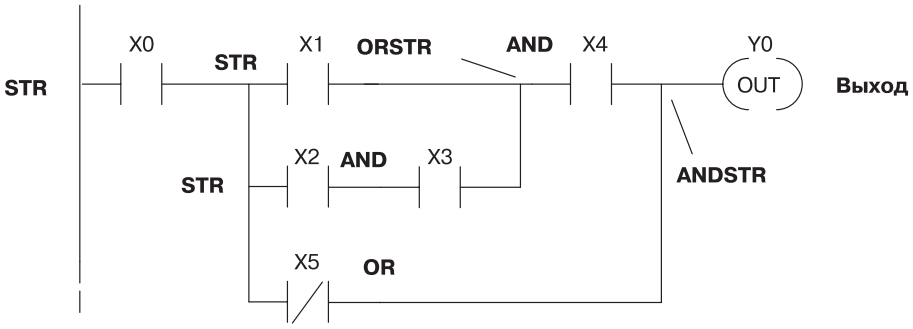
Вы можете комбинировать различные типы последовательных и параллельных цепей для решения любых приложений. Следующий пример показывает простую комбинацию цепей.



Булевский стек

Существуют ограничения на количество элементов в цепи. Это связано с тем, что процессор DL405 используют 8-уровневые булевские стеки для вычисления различных элементов логического выражения. Булевский стек - это временная область памяти, которая хранит и решает логические выражения цепи. Каждый раз, когда Вы вводите команду STR, эта команда располагается в верхней части стека. Любые другие команды STR в стеке опускают верхнюю команду на один уровень вниз. Команды ANDSTR и ORSTR производят соответствующую логическую операцию над соседними уровнями стека. Так как стек имеет только восемь уровней, то если процессор обнаружит цепь, которая использует больше восьми уровней, то случится ошибка.

Следующий пример показывает, как булевский стек используется для решения булевской логики.



STR X0

1	STR X0
2	
3	
4	
5	
6	
7	
8	

STR X1

1	STR X1
2	STR X0
3	
4	
5	
6	
7	
8	

STR X2

1	STR X2
2	STR X1
3	STR X0
4	
5	
6	
7	
8	

AND X3

1	X2 AND X3
2	STR X1
3	STR X0
4	
5	
6	
7	
8	

ORSTR

1	X1 OR (X2 AND X3)
2	STR X0
3	
⋮	
8	

AND X4

1	X4 AND [X1 OR (X2 AND X3)]
2	STR X0
3	
⋮	
8	

ORNOT X5

1	NOT X5 OR X4 AND [X1 OR (X2 AND X3)]
2	STR X0
3	
⋮	
8	

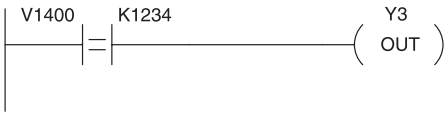
ANDSTR

1	X0 AND (NOT X5 OR X4) AND [X1 OR (X2 AND X3)]
2	
3	
⋮	
8	

Булевское сравнение

Булевские команды сравнения обеспечивают работу с двумя 4-знаковыми значениями в двоично-десятичном или в шестнадцатеричном формате, используя булевские контакты. Действующие операции сравнения: равно, не равно, равно или больше чем, равно или меньше чем.

В следующем примере, когда величина в ячейке памяти V1400 равна постоянной величине 1234, то Y3 срабатывает.

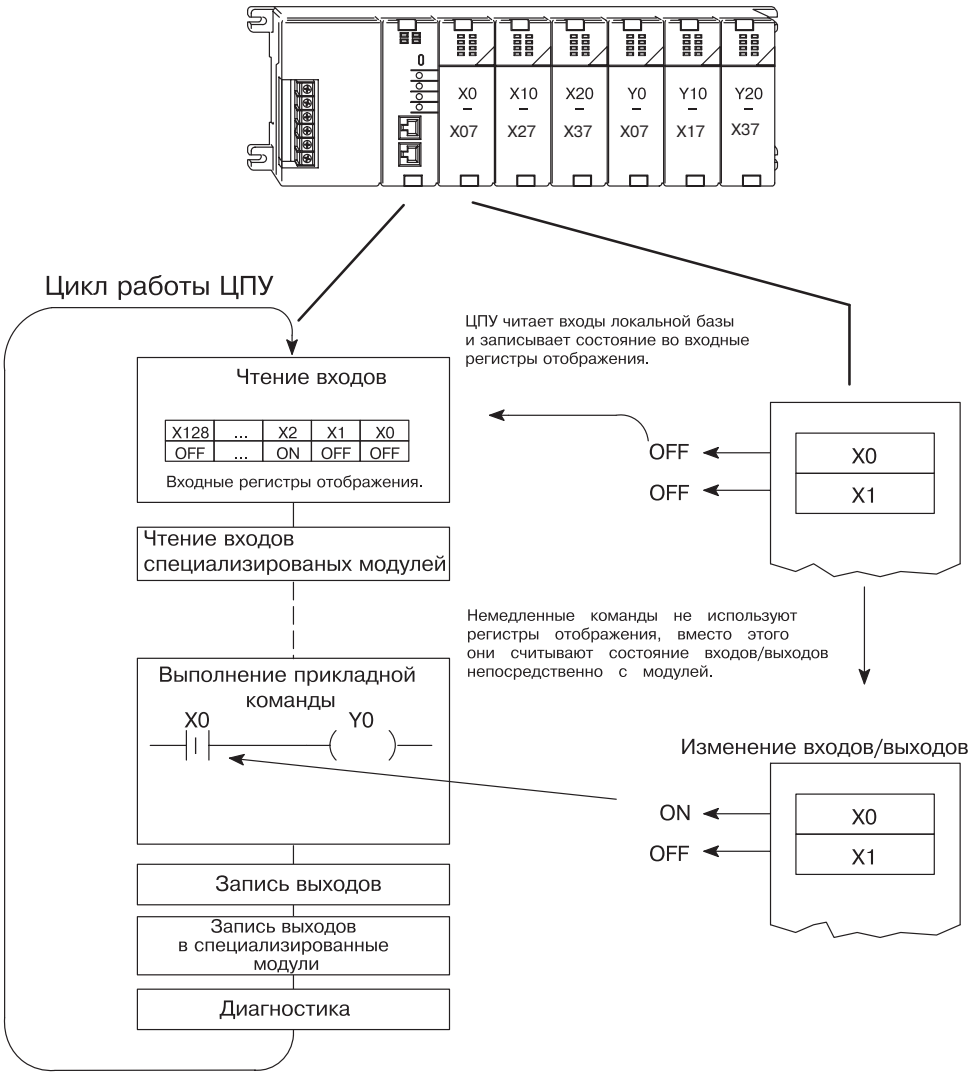


Прямые команды (Immediate Boolean)

Процессор DL405 обычно заканчивает цикл операций за несколько миллисекунд. Однако в некоторых приложениях у Вас не будет возможности ждать несколько миллисекунд, пока не случится следующее обновление входов/выходов. Процессоры DL405 имеют специальные немедленные команды, которые позволяют выполнять чтение непосредственно со входов и запись непосредственно на выходы во время процесса выполнения программы. Вы можете вспомнить, что это обычно делается во время обновления входов/выходов. Немедленные команды используют больше времени для выполнения, потому что процесс выполнения программы прерывается, когда процессор читает или записывает данные. Эта функция обычно не выполняется, когда идет процесс чтения входов и записи на выходы.

ПРИМЕЧАНИЕ. Несмотря на то, что команды немедленного ввода читают текущее состояние модуля, это состояние используется только внутри команды. Команда не обновляет состояние входных регистров отображения (X). Поэтому, любые обычные команды будут использовать неизмененное состояние входных регистров. Любые немедленные команды ввода будут повторно опрашивать состояние модуля.

Немедленная команда вывода изменит состояние модуля и обновит выходные регистры отображения (У).

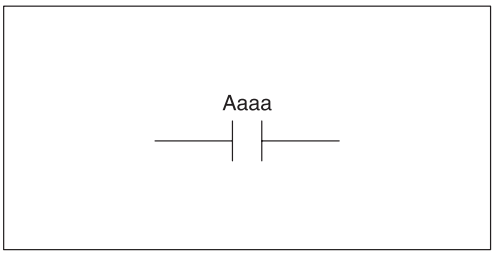


Булевские команды

Store (STR)

√	√	√
430	440	450

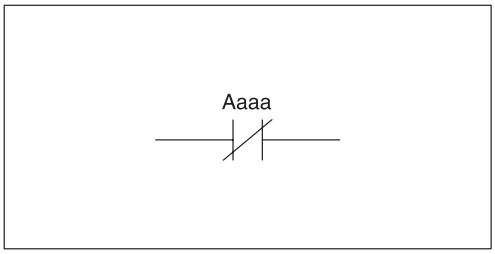
Команда Store начинает новую цепь или дополнительную ветвь в цепи с нормально открытым контактом. Состояние контакта будет тем же самым, что и состояние соответствующего регистра отображения или ячейки памяти.



Store Not (STRN)

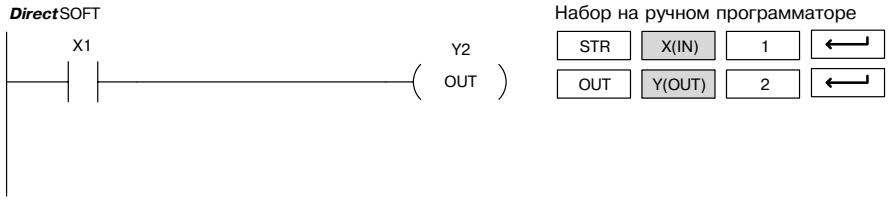
√	√	√
430	440	450

Команда Store Not начинает новую цепь или дополнительную ветвь в цепи с нормально закрытым контактом. Состояние контакта будет противоположно состоянию соответствующего регистра отображения или ячейки памяти.

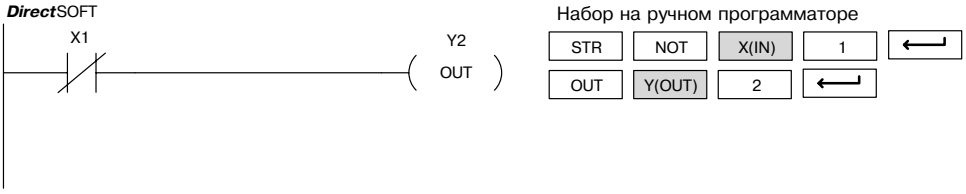


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	aaa	aaa	aaa
Входы	X	0-477	0-477	0-1777
Выходы	Y	0-477	0-477	0-1777
Реле управления	C	0-737	0-1777	0-3777
Стадия	S	0-577	0-1777	0-1777
Таймер	T	0-177	0-377	0-377
Счетчик	CT	0-177	0-177	0-377
Специальное реле	SP	0-137, 320-617	0-137 320-717	0-137, 320-717
Глобальные	GX	0-777	0-1777	0-2777
Глобальные	GY	-	-	0-2777

В следующем примере Store, когда вход X1 включен, выход Y2 срабатывает.



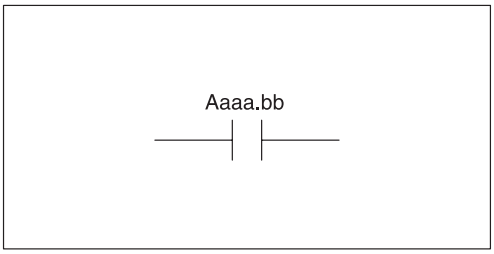
В следующем примере Store Not, когда вход X1 отключен, выход Y2 срабатывает.



Store Bit-of-Word (STRB)

X	X	√
430	440	450

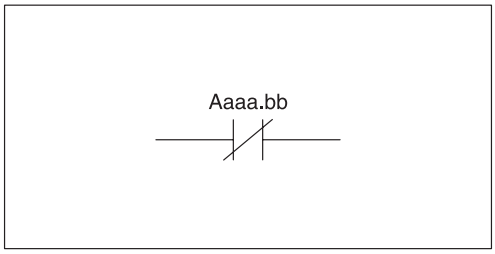
Команда Store Bit-of-Word начинает новую цепь или дополнительную ветвь в цепи с нормально открытым контактом. Состояние контакта будет тем же самым, что и состояние бита соответствующей ячейки памяти.



Store Not Bit-of-Word (STRNB)

X	X	√
430	440	450

Команда Store Not Bit-of-Word начинает новую цепь или дополнительную ветвь в цепи с нормально закрытым контактом. Состояние контакта будет противоположно положению бита, относящегося к соответствующей ячейке памяти.



Тип данных операнда		Диапазон DL450	
	A	aaa	bb
V-память	B	Все (см. стр. 3-42)	BCD, 0-15
Пойнтер	PB	Все (см. стр. 3-42)	BCD, 0-15

В следующем примере Store Bit-of-Word, когда бит 12 ячейки памяти V1400 включен, выход Y2 срабатывает.

Набор на ручном программаторе

STR	SHFT	B	SHFT	V	1	4	0	0
K(con)	1	2	↵					
OUT	Y(OUT)	2	↵					

В следующем примере Store Not Bit-of-Word, когда бит 12 ячейки памяти V1400 выключен, выход Y2 срабатывает.

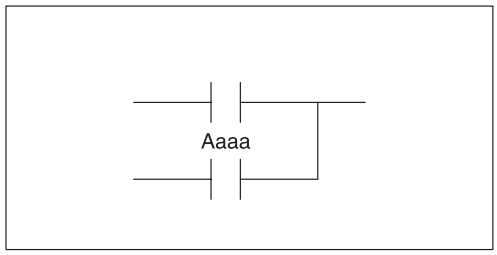
Набор на ручном программаторе

STR	STR	SHFT	B	SHFT	V	1	4	0	0
K(con)	1	2	↵						
OUT	Y(OUT)	2	↵						

Or
(OR)

√	√	√
430	440	450

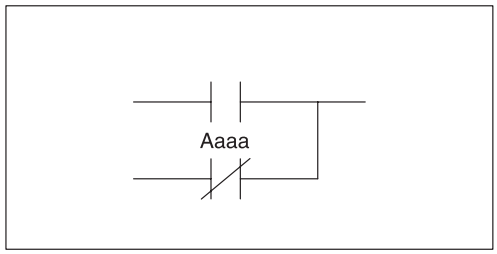
Команда Or выполняет операцию логического ИЛИ для цепи, состоящей из нормально открытого контакта, соединенного параллельно с другим контактом. Состояние контакта будет тем же самым, что и соответствующая точка регистра отображения или ячейки памяти.



Or Not
ORN

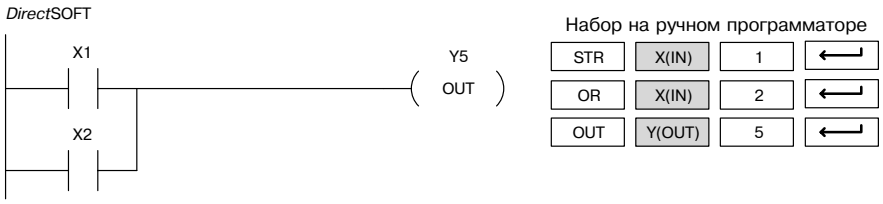
√	√	√
430	440	450

Команда Or Not выполняет операцию логического ИЛИ для цепи, состоящей из нормально закрытого контакта, соединенного параллельно с другим контактом. Состояние контакта будет противоположно состоянию соответствующей точки регистра отображения или ячейки памяти.

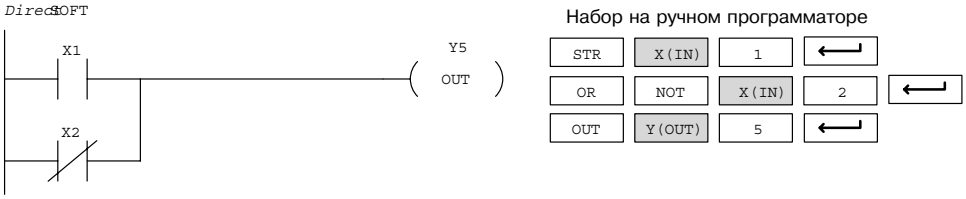


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	aaa	aaa	aaa
Входы	X	0-477	0-477	0-1777
Выходы	Y	0-477	0-477	0-1777
Реле управления	C	0-737	0-1777	0-3777
Стадия	S	0-577	0-1777	0-1777
Таймер	T	0-177	0-377	0-377
Счетчик	CT	0-177	0-177	0-377
Специальное реле	SP	0-137, 320-617	0-137 320-717	0-137, 320-717
Глобальные	GX	0-777	0-1777	0-1777
Глобальные	GY	-	-	0-1777

В следующем примере Or, когда вход X1 или X2 включен, выход Y5 срабатывает.



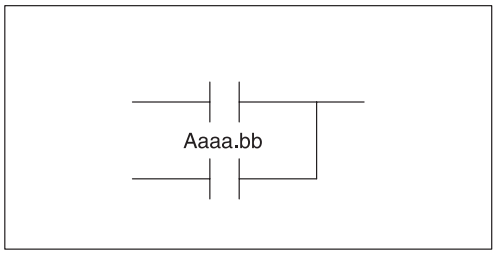
В следующем примере Or Not, когда вход X1включен или X2 выключен, выход Y5 срабатывает.



Or Bit-of-Word (ORB)

X	X	√
430	440	450

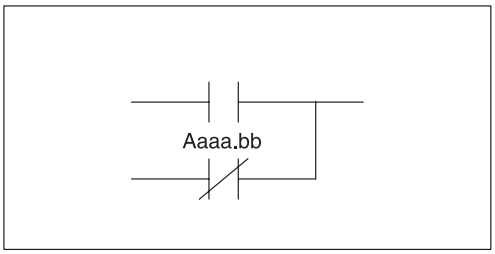
Команда Or Bit-of-Word выполняет операцию логического ИЛИ для цепи, состоящей из нормально открытого контакта, соединенного параллельно с другим контактом. Состояние контакта будет тем же самым, что и состояние бита соответствующей ячейки памяти.



Or Not Bit-of-Word (ORNB)

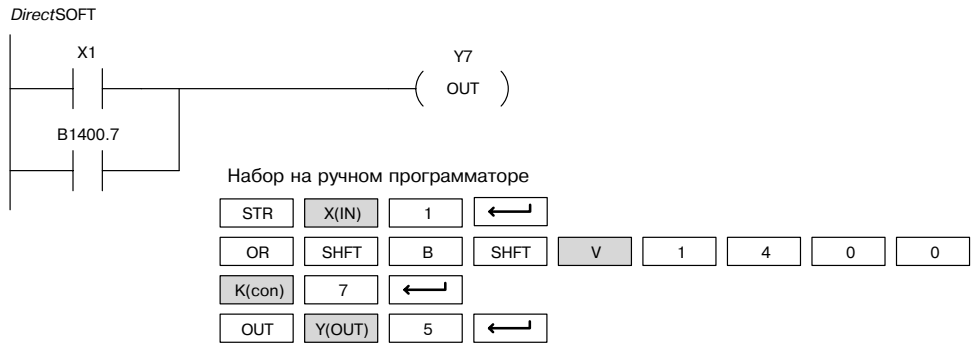
X	X	√
430	440	450

Команда Or Not Bit-of-Word выполняет операцию логического ИЛИ для цепи, состоящей из нормально закрытого контакта, соединенного параллельно с другим контактом. Состояние контакта будет противоположно состоянию бита соответствующей ячейки памяти.

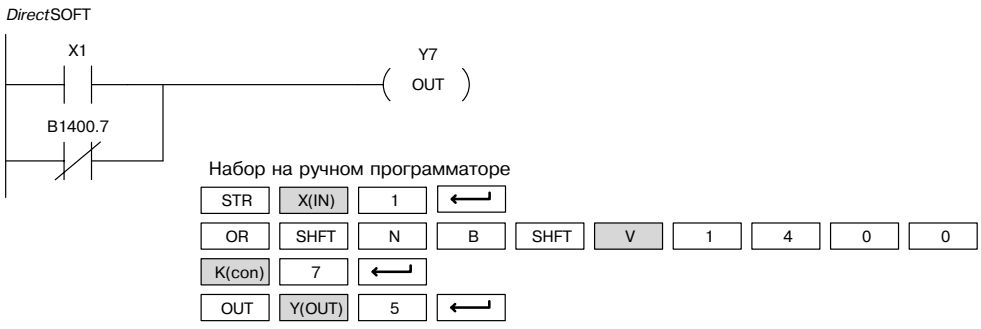


Тип данных операнда		Диапазон DL450	
	A	aaa	bb
V-память	B	Все (см. стр. 3-42)	BCD, 0-15
Пойнтер	PB	Все (см. стр. 3-42)	BCD,

В следующем примере Or Bit-of-Word, когда вход X1 или бит 7 ячейки памяти V1400 включен, выход Y5 срабатывает.



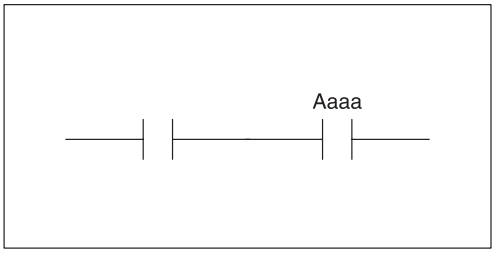
В следующем примере Or Bit-of-Word, когда вход X1 или бит 7 ячейки памяти V1400 выключен, выход Y5 срабатывает.



And (AND)

√	√	√
430	440	450

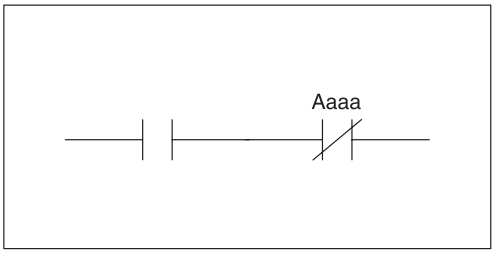
Команда And выполняет операцию логического И для цепи, состоящей из нормально открытого контакта, соединенного последовательно с другим контактом. Состояние контакта будет тем же самым, что и состояние соответствующего регистра отображения или ячейки памяти.



And Not (ANDN)

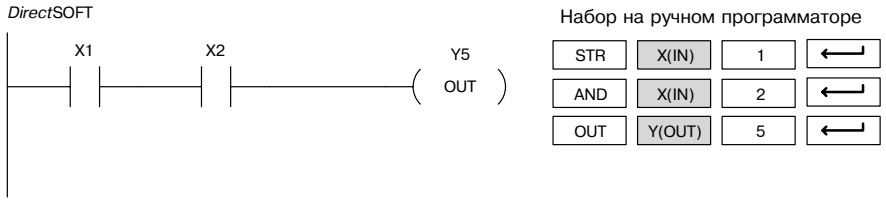
√	√	√
430	440	450

Команда And Not выполняет операцию логического И для цепи, состоящего из нормально закрытого контакта, соединенного последовательно с другим контактом. Состояние контакта будет противоположным состоянию соответствующего регистра отображения или ячейки памяти.

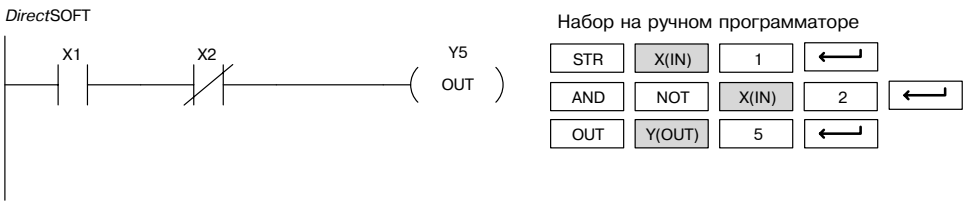


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	aaa	aaa	aaa
Входы	X	0-477	0-477	0-1777
Выходы	Y	0-477	0-477	0-1777
Реле управления	C	0-737	0-1777	0-3777
Стадия	S	0-577	0-1777	0-1777
Таймер	T	0-177	0-377	0-377
Счетчик	CT	0-177	0-177	0-377
Специальное реле	SP	0-137, 320-617	0-137 320-717	0-137, 320-717
Глобальные	GX	0-777	0-1777	0-1777
Глобальные	GY	-	-	0-1777

В следующем примере And, когда вход X1 или X2 включены, выход Y5 срабатывает.



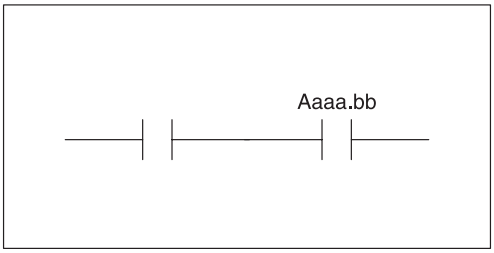
В следующем примере And Not, когда вход X1 включен и вход X2 выключен, выход Y5 срабатывает.



And Bit-of-Word (ANDB)

X	X	√
430	440	450

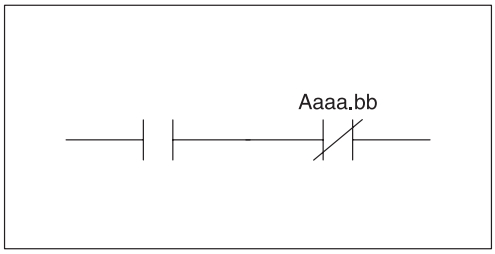
Команда And Bit-of-Word выполняет операцию логического И для цепи, состоящей из нормально открытого контакта, соединенного последовательно с другим контактом. Состояние контакта будет тем же самым, что и состояние бита соответствующей ячейки памяти.



And Not Bit-of-Word (ANDNB)

X	X	√
430	440	450

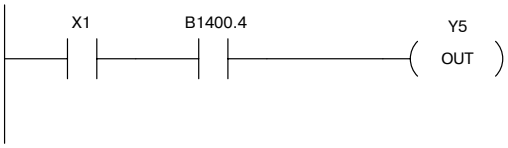
Команда And Not Bit-of-Word выполняет операцию логического И-нет для цепи, состоящей из нормально закрытого контакта, соединенного последовательно с другим контактом. Состояние контакта будет противоположно положению бита соответствующей ячейки памяти.



Тип данных операнда		Диапазон DL450	
	A	aaa	bb
V-память	B	Все (см. стр. 3-42)	BCD, 0-15
Пойнтер	PB	Все (см. стр. 3-42)	BCD,

В следующем примере And Bit-of-Word, когда вход X1 и бит 4 ячейки памяти V1400 включены, выход Y5 срабатывает.

*Direct*SOFT

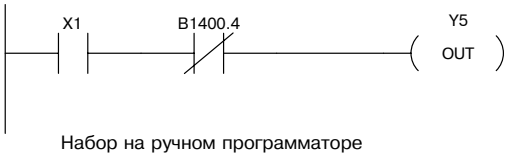


Набор на ручном программаторе

STR	X(IN)	1	↵					
AND	SHFT	B	SHFT	V	1	4	0	0
K(con)	4	↵						
OUT	Y(OUT)	5	↵					

В следующем примере And Not Bit-of-Word, когда вход X1 включен и бит 4 ячейки памяти V1400 выключен, выход Y5 срабатывает.

*Direct*SOFT



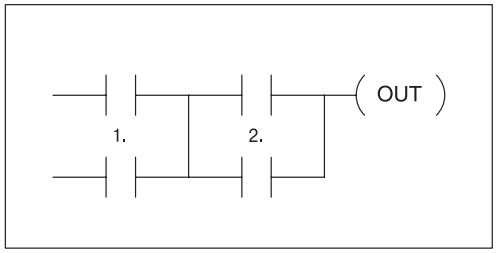
Набор на ручном программаторе

STR	X(IN)	1	↵						
AND	SHFT	N	B	SHFT	V	1	4	0	0
K(con)	4	↵							
OUT	Y(OUT)	5	↵						

And Store (AND STR)

√	√	√
430	440	450

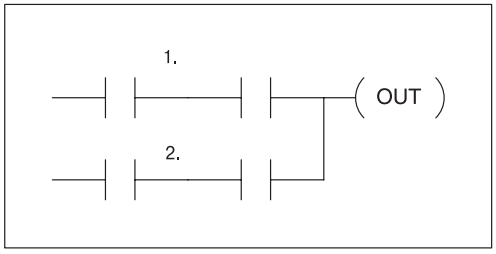
Команда And Store выполняет операцию логического И для цепи, состоящей из двух последовательных ветвей. Обе ветви должны начинаться с команды Store.



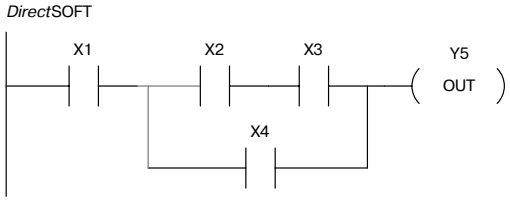
Or Store (OR STR)

√	√	√
430	440	450

Команда Or Store выполняет операцию логического ИЛИ для цепи, состоящеq из двух параллельных ветвей. Обе ветви должны начинаться с команды Store.



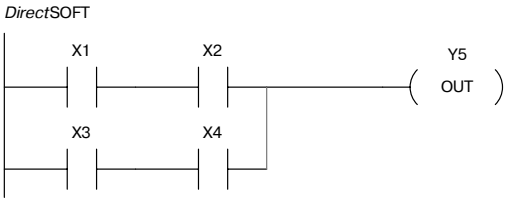
В следующем примере And Store была выполнена операция AND над ветвью, состоящей из контактов X2, X3 и X4, и ветвью, состоящей из контакта X1.



Набор на ручном программаторе

STR	X(IN)	1	↵
STR	X(IN)	2	↵
AND	X(IN)	3	↵
OR	X(IN)	4	↵
AND	STR	↵	
OUT	Y(OUT)	5	↵

В следующем примере Or Store была выполнена операция OR над ветвью, состоящей из контактов X1 и X2, и ветвью, состоящей из контактов X3 и X4.



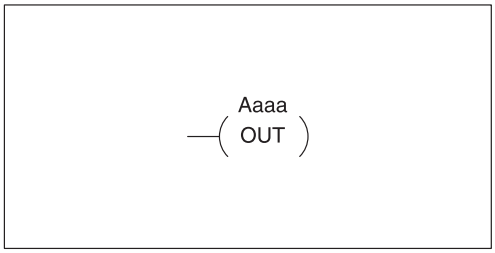
Набор на ручном программаторе

STR	X(IN)	1	↵
AND	X(IN)	2	↵
STR	X(IN)	3	↵
AND	X(IN)	4	↵
OR	STR	↵	
OUT	Y(OUT)	5	↵

Out (OUT)

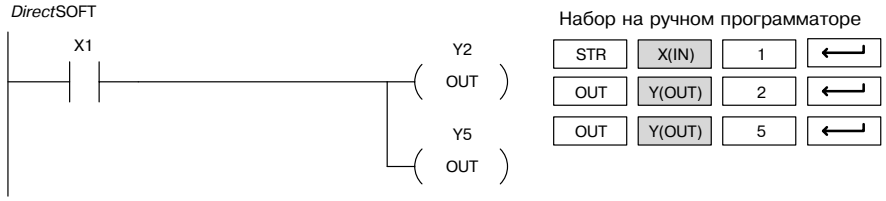
√	√	√
430	440	450

Команда Out отражает состояние цепи (вкл/выкл) и выводит дискретное (вкл/выкл) состояние в определенное место регистра отображения или ячейку памяти. Нельзя использовать несколько команд Out, ссылающихся на одну и ту же дискретную ячейку, потому что только последняя в программе команда Out будет управлять физической точкой выхода.

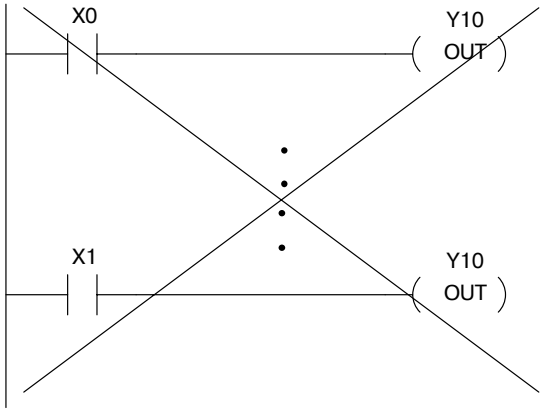


Тип данных операнда	A	Диапазон DL430	Диапазон DL440	Диапазон DL450
		aaa	bb	bb
Входы	X	0-477	0-477	0-1777
Выходы	Y	0-477	0-477	0-1777
Реле управления	C	0-737	0-1777	0-3777
Глобальный ввод/вывод	GX	0-777	0-1777	0-2777 (GX + GY)

В следующем примере Out, когда вход X1 включен, выходы Y2 и Y5 срабатывают.



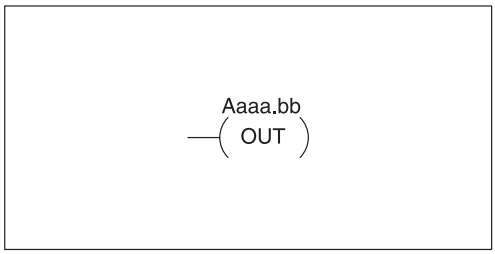
В следующем примере Out программа содержит две команды Out, использующие одну и ту же ячейку (Y10). Физический выход Y10 в конечном счете управляется последним цепью логики, ссылающейся на Y10. X1 отменит выход Y10, управляемый X0. Чтобы избежать этого, в программах не должны использоваться многократные выходы к одной и той же ячейке.



Out Bit-of-Word (OUTB)

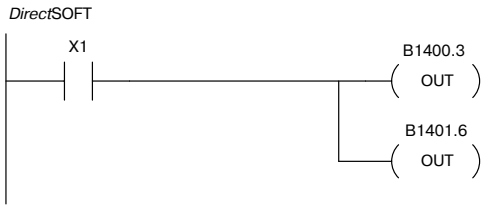
X	X	√
430	440	450

Команда Out Bit-of-Word отражает состояние цепи (вкл/выкл) и выводит дискретное (вкл/выкл) состояние на определенный бит соответствующей ячейки памяти. Нельзя использовать несколько команд Out Bit-of-Word, ссылающихся на один и тот же бит одного и того же слова, потому что только последняя в программе команда Out будет управлять состоянием бита.



Тип данных операнда		Диапазон DL450	
	A	aaa	bb
V-память	B	Все (см. стр. 3-42)	BCD, 0-15
Пойнтер	PB	Все (см. стр. 3-42)	BCD,

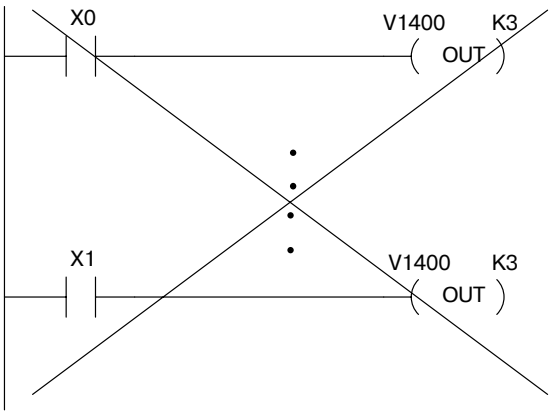
В следующем примере Out Bit-of-Word, когда вход X1 включен, бит 3 ячейки памяти V1400 и бит 6 ячейки памяти V1401 будут включены.



Набор на ручном программаторе

STR	X(IN)	1	↵								
OUT	SHFT	B	SHFT	V	1	4	0	0	K(con)	3	↵
OUT	SHFT	B	SHFT	V	1	4	0	1	K(con)	6	↵

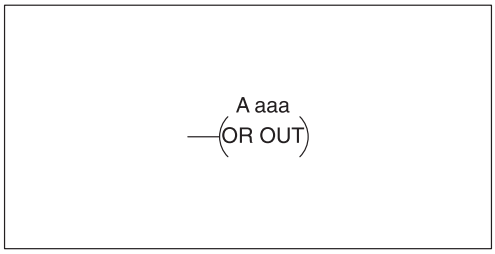
Следующий пример Out Bit-of-Word содержит две команды Out Bit-of-Word, использующие один и тот же бит в одном и том же слове. Конечное состояние бита 3 ячейки памяти V1400 в конечном счете управляется последней цепью логики, ссылающейся на него. X1 заменит логическое состояние, управляемое X0. Чтобы избежать этого, в программах не должны использоваться многократные выходы к одной и той же ячейке.



Or Out (OR OUT)

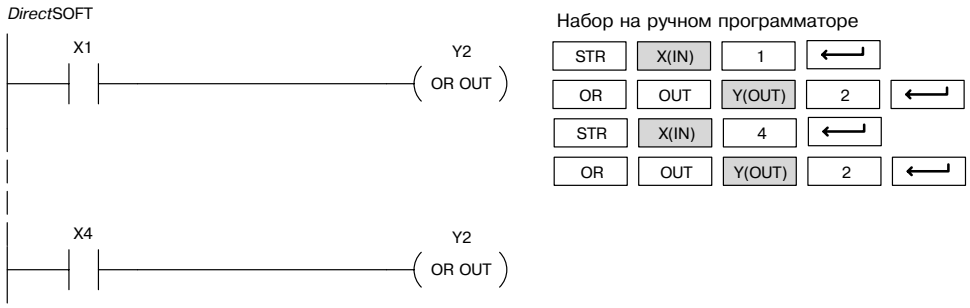
√	√	√
430	440	450

Команда Or Out была разработана для использования нескольких цепей дискретной логики для управления одним выходом. Возможно использование нескольких команд Or Out, ссылающихся на один и тот же выход, так как операция ИЛИ выполняется одновременно над всеми контактами, управляющими выходом. Если состояние любой из цепей включено, то выход будет также включен.



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**bb**	**bb**
Входы	X	0-477	0-477	0-1777
Выходы	Y	0-477	0-477	0-1777
Реле управления	C	0-737	0-1777	0-3777
Глобальный ввод/вывод	GX	0-777	0-1777	0-2777 (GX + GY)

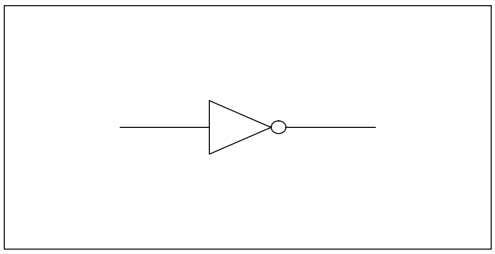
В следующем примере, когда X1 или X4 включены, Y2 сработает.



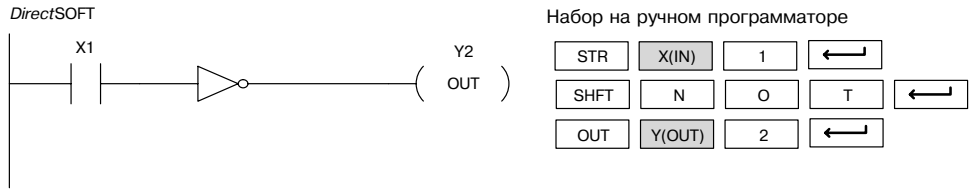
Not (NOT)

√	√	√
430	440	450

Команда Not инвертирует состояние цепи в точке команды.



В следующем примере, когда X1 выключен, Y2 сработает. Это потому, что команда Not инвертирует состояние цепи в команде Not.

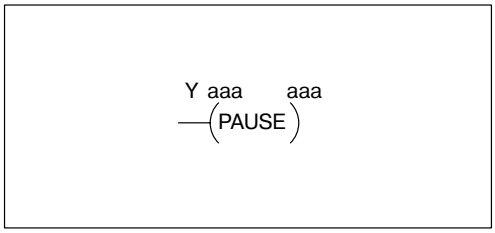


ПРИМЕЧАНИЕ. Direct SOFT версии 1.1i и более поздние версии поддерживает использование команды NOT. Вышеупомянутый пример цепи просто предназначен для того, чтобы показать визуальное представление команды NOT. Это цепь не может быть создана или отображена в версиях Direct SOFT более ранних, чем 1.1i.

Pause (PAUSE)

√	√	√
430	440	450

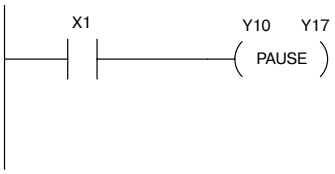
Команда Pause отключает обновление выхода на заданном диапазоне выходов. Программа RLL логики будет продолжать работать и обновлять регистр отображения, однако выходы модулей будут выключены в диапазоне, заданном в команде Pause.



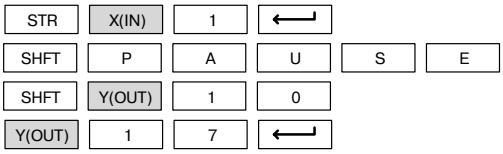
Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**bb**	**bb**
Выходы	Y	0-477	0-477	0-1777

В следующем примере, когда X1 в состоянии ВКЛ., Y10-Y17 будут выключены в выходном модуле. Выполнение программы не будет отражаться на них.

*Direct*SOFT Display



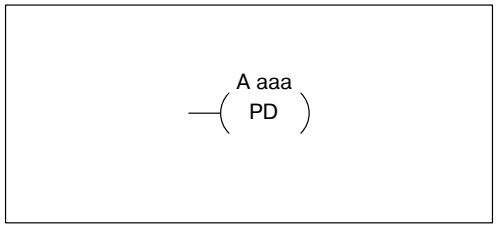
Набор на ручном программаторе



Positive Differential (PD)

√	√	√
430	440	450

Команду Positive Differential обычно называют "Одиночный импульс". Когда входная логика меняет сигнал с ВЫКЛ на ВКЛ., выход сработает на время одного цикла сканирования процессора. Затем выход остается в состоянии ВЫКЛ. до тех пор, пока для входа не произойдет следующий перехода с ВЫКЛ на ВКЛ.



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**bb**	**bb**
Входы	X	0-477	0-477	0-1777
Выходы	Y	0-477	0-477	0-1777
Реле управления	C	0-737	0-1777	0-3777

В следующем примере каждый раз, когда X1 переходит из состояния ВЫКЛ. в состояние ВКЛ., C0 сработает на время одного цикла сканирования.

*Direct*SOFT

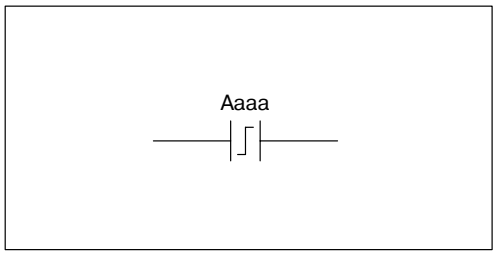
Набор на ручном программаторе

Следует отметить, что Вы можете поставить команду NOT непосредственно перед командой PD, чтобы сгенерировать "одноразовый" импульс при переходе из ВКЛ в ВЫКЛ.

Store Positive Differential (STRPD)

X	X	√
430	440	450

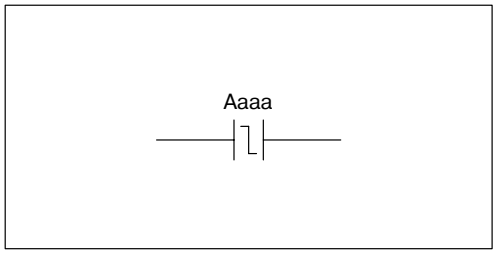
Команда Store Positive Differential начинает новую цепочку или дополнительную ветвь в цепочке при нормально открытом контакте. Контакт закрывается на один цикл сканирования процессора, когда состояние соответствующей точки регистра отображения переходит из ВЫКЛ в ВКЛ. После этого контакт остается открытым до тех пор, пока не произойдет следующий переход из ВЫКЛ в ВКЛ (символ внутри контакта означает этот переход).



Store Negative Differential (STRND)

X	X	√
430	440	450

Команда Store Negative Differential начинает новую цепочку или дополнительную ветвь в цепочке при нормально закрытом контакте. Контакт закрывается на один цикл сканирования процессора, когда состояние соответствующей точки регистра отображения переходит из ВКЛ в ВЫКЛ. После этого контакт остается открытым до тех пор, пока не произойдет следующий переход из ВКЛ в ВЫКЛ (символ внутри контакта обозначает данный переход).



Тип данных операнда		Диапазон DL450
	A	aaa
Входы	X	0-1777
Выходы	Y	0-1777
Реле управления	C	0-3777
Стадия	S	0-1777
Таймер	T	0-377
Счетчик	CT	0-377
Глобальные	GX	0-2777 (GX+GY)

В следующем примере каждый раз, когда X1 делает переход из ВКЛ в ВЫКЛ, Y4 включается на одно сканирование.

*Direct*SOFT

Набор на ручном программаторе

STR	SHFT	P	D	SHFT
X(IN)	1	↵		
OUT	Y(OUT)	4	↵	

В следующем примере каждый раз, когда X1 делает переход из ВЫКЛ в ВКЛ, Y4 включается на одно сканирование.

*Direct*SOFT

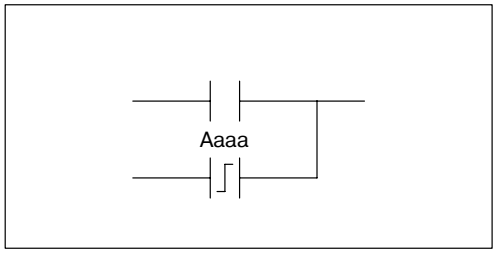
Набор на ручном программаторе

STR	SHFT	N	D	SHFT
X(IN)	1	↵		
OUT	Y(OUT)	4	↵	

Or Positive Differential (ORPD)

X	X	√
430	440	450

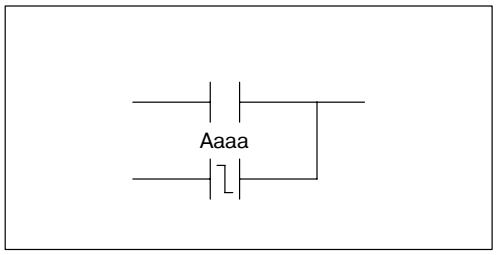
Команда Or Positive Differential является логическим ”ИЛИ” нормально открытого контакта, параллельного другому контакту в цепочке. Состояние контакта остается открытым до тех пор, пока точка соответствующего регистра отображения не сделает переход от состояния ВЫКЛ к состоянию ВКЛ, закрывая ее на один цикл сканирования процессора. После этого контакт остается открытым до следующего перехода из ВЫКЛ в ВКЛ.



Or Negative Differential (ORND)

X	X	√
430	440	450

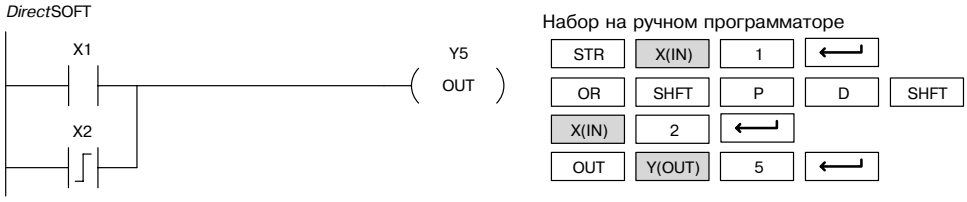
Команда Or Negative Differential является логическим ”ИЛИ” нормально открытого контакта, параллельного другому контакту в цепочке. Состояние контакта остается открытым до тех пор, пока точка соответствующего регистра отображения не сделает переход от состояния ВКЛ к состоянию ВЫКЛ, закрывая ее на один цикл сканирования процессора.



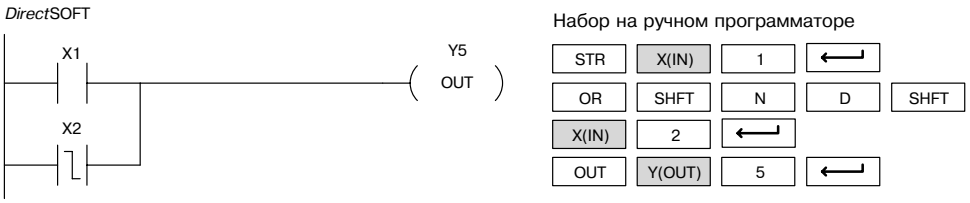
После этого контакт остается открытым до следующего перехода из ВКЛ в ВЫКЛ.

Тип данных операнда		Диапазон DL450
	A	aaa
Входы	X	0-1777
Выходы	Y	0-1777
Реле управления	C	0-3777
Стадия	S	0-1777
Таймер	T	0-377
Счетчик	CT	0-377
Глобальные	GX	0-2777 (GX+GY)

В следующем примере Y5 срабатывает всякий раз, когда X1 включен, или только в одном цикле сканирования, когда X2 переходит из ВЫКЛ в ВКЛ



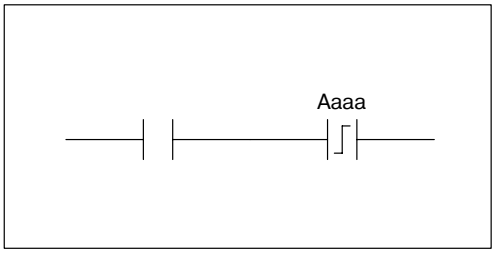
В следующем примере Y5 срабатывает всякий раз, когда X1 включен, или только в одном цикле сканирования, когда X2 переходит из ВКЛ в ВЫКЛ



AND Positive Differential (ANDPD)

X	X	√
430	440	450

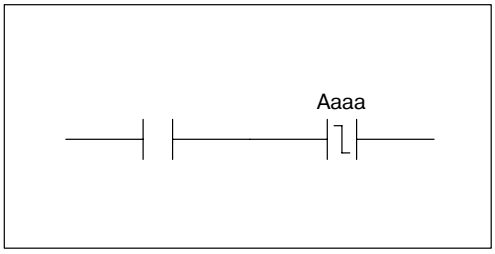
Команда AND Positive Differential является логическим ”И” нормально открытого контакта, последовательного другому контакту в цепочке. Состояние контакта остается открытым до тех пор, пока точка соответствующего регистра отображения не сделает переход от состояния ВЫКЛ к состоянию ВКЛ, закрывая ее на один цикл сканирования процессора. После этого контакт остается открытым до следующего перехода из ВЫКЛ в ВКЛ.



AND Negative Differential (ANDPD)

X	X	√
430	440	450

Команда AND Negative Differential является логическим ”И” нормально открытого контакта, последовательного другому контакту в цепочке. Состояние контакта остается открытым до тех пор, пока точка соответствующего регистра отображения не сделает переход от состояния ВКЛ к состоянию ВЫКЛ, закрывая ее на один цикл сканирования процессора. После этого контакт остается открытым до следующего перехода из ВКЛ в ВЫКЛ.



Тип данных операнда		Диапазон DL450
	A	aaa
Входы	X	0-1777
Выходы	Y	0-1777
Реле управления	C	0-3777
Стадия	S	0-1777
Таймер	T	0-377
Счетчик	CT	0-377
Глобальные	GX	0-2777 (GX+GY)

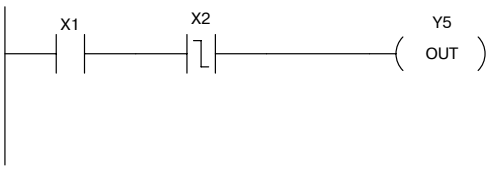
В следующем примере Y5 срабатывает в одном цикле сканирования всякий раз, когда X1 включен и X2 переходит из ВЫКЛ в ВКЛ.

Набор на ручном программаторе

STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	4	9	3	3	↵			
OUT	Y(OUT)	3	↵					

В следующем примере Y5 срабатывает в одном цикле сканирования всякий раз, когда X1 включен и X2 переходит из ВКЛ в ВЫКЛ.

*Direct*SOFT



Набор на ручном программаторе

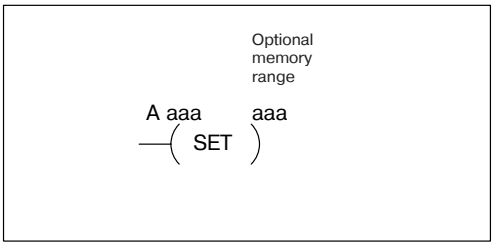
STR	X(IN)	1	↵	
AND	SHFT	N	D	SHFT
X(IN)	2	↵		
OUT	Y(OUT)	5	↵	

Set (SET)

√	√	√
430	440	450

Команда Set устанавливает или включает точку регистра отображения / ячейку памяти или последовательный диапазон точек регистра отображения / ячеек памяти. Как только точка/ячейка установлена, она будет оставаться включенной до тех пор, пока она не будет сброшена командой Reset.

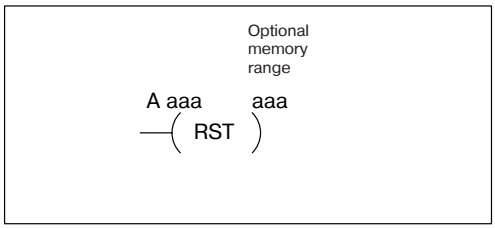
Необязательно вход, управляющий командой Set, должен сохраняться включенным.



Reset (RST)

√	√	√
430	440	450

Команда Reset сбрасывает или выключает точку регистра отображения / ячейку памяти или диапазон точек регистра отображения / ячеек памяти. Как только точка/ячейка сброшена необязательно сохранять вход включенным.



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
Входы	X	0-477	0-477	0-1777
Выходы	Y	0-477	0-477	0-1777
Реле управления	C	0-737	0-1777	0-3777
Стадия	S	0-577	0-1777	0-1777
Таймер *	T	0-177	0-377	0-377
Счетчик *	CT	0-177	0-177	0-377
Глобальные	GX	0-777	0-1777	0-2777 (GX + GY)

* Типы данных операнда таймера и счетчика нельзя использовать в команде Set

В следующем примере, когда X1 включен, Y5 - Y22 будут установлены во включенное состояние.

Набор на ручном программаторе

STR	X(IN)	1	↵			
SET	Y(OUT)	5	Y(OUT)	2	2	↵

В следующем примере, когда X1 включен, Y5 - Y22 будут сброшены или отключены.

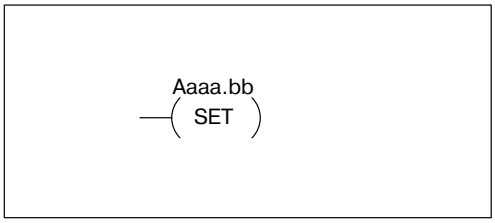
Набор на ручном программаторе

STR	X(IN)	1	↵			
RST	Y(OUT)	5	Y(OUT)	2	2	↵

Set Bit-of-Word (SETB)

X	X	√
430	440	450

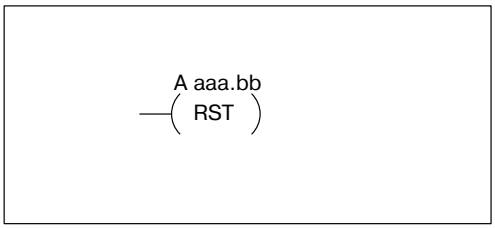
Команда Set Bit-of-Word устанавливает или включает бит в ячейке V-памяти. Как только бит установлен, он будет оставаться включенным до тех пор, пока его не сбросят командой Reset Bit-of-Word. Необязательно сохранение входного управления командой Set Bit-of-Word.



Reset Bit-of-Word (RSTB)

X	X	√
430	440	450

Команда Reset Bit-of-Word сбрасывает или выключает бит в ячейке V-памяти. Как только бит сброшен, необязательно входу оставаться в состоянии ВКЛ.



Тип данных операнда		Диапазон DL450	
	A	aaa	bb
V-память	B	Все (см. стр. 3-42)	0-15
Пойнтер	PB	Все (см. стр. 3-42)	0-15

В следующем примере, когда X1 включается, бит 0 в ячейке памяти V1400 устанавливается в положение ВКЛ.

*Direct*SOFT

Набор на ручном программаторе

STR	X(IN)	1	↵					
SET	SHFT	B	SHFT	V	1	4	0	0
K(con)	0	↵						

В следующем примере, когда X1 включается, бит 15 в ячейке памяти V1400 сбрасывается в положение ВЫКЛ.

*Direct*SOFT

Набор на ручном программаторе

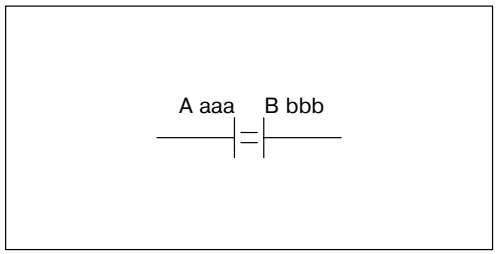
STR	X(IN)	1	↵					
RST	SHFT	B	SHFT	V	1	4	0	0
K(con)	1	5	↵					

Булевские команды сравнения

Store If Equal (STRE)

√	√	√
430	440	450

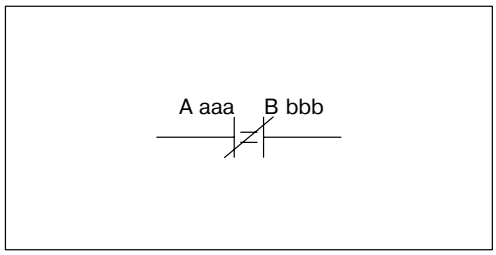
Команда Store If Equal начинает новую цепь или дополнительную ветвь в цепи с нормально открытым контактом сравнения. Контакт будет включен, когда Vaaa = Bbbb.



Store If Not Equal (STRNE)

√	√	√
430	440	450

Команда Store If Not Equal начинает новую цепь или дополнительную ветвь в цепи с нормально закрытым контактом сравнения. Контакт будет включен, когда Vaaa № Bbbb.



Тип данных операнда		Диапазон DL430		Диапазон DL440		Диапазон DL450	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-41)	Все (см. стр. 3-42)	Все (см. стр. 3-42)
Указатель	P	—	—	Все (см. стр. 3-41)	Все (см. стр. 3-41)	Все (см. стр. 3-42)	Все (см. стр. 3-42)
Константа	K	—	0-FFFF	—	0-FFFF	—	0-FFFF

В следующем примере, когда значение в ячейке памяти V1400 = 4933, Y3 сработает.

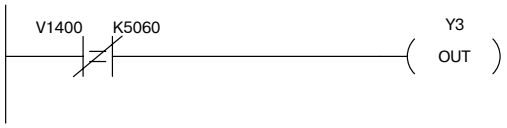
*Direct*SOFT

Набор на ручном программаторе

STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	4	9	3	3	↵			
OUT	Y(OUT)	3	↵					

В следующем примере, когда значение в ячейке памяти V1400 ≠ 5060, Y3 сработает.

*Direct*SOFT



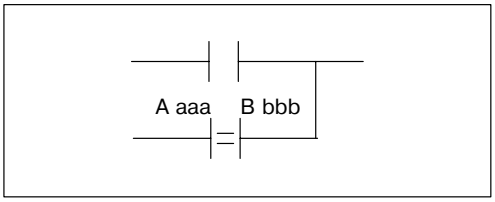
Набор на ручном программаторе

STR	NOT	SHFT	E	SHFT	V	1	4	0	0
K(CON)	5	0	6	0	↵				
OUT	Y(OUT)	3	↵						

Or If Equal (ORE)

√	√	√
430	440	450

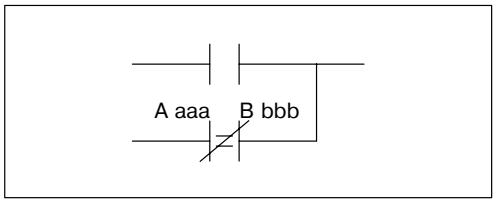
Команда Or If Equal соединяет нормально открытый контакт сравнения параллельно с другим контактом. Контакт будет включен, когда Vaaa = Bbbb.



Or If Not Equal (ORNE)

√	√	√
430	440	450

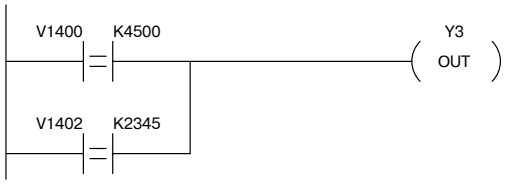
Команда Or If Not Equal соединяет нормально закрытый контакт сравнения параллельно с другим контактом. Контакт будет включен, когда Vaaa № Bbbb.



Тип данных операнда		Диапазон DL430		Диапазон DL440		Диапазон DL450	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-41)	Все (см. стр. 3-42)	Все (см. стр. 3-42)
Указатель	P	—	—	Все (см. стр. 3-41)	Все (см. стр. 3-41)	Все (см. стр. 3-42)	Все (см. стр. 3-42)
Константа	K	—	0-FFFF	—	0-FFFF	—	0-FFFF

В следующем примере, когда значение в ячейке памяти V1400 = 4500 или в ячейке V1402 = 2345, Y3 сработает.

*Direct*SOFT

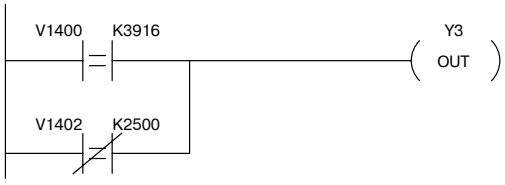


Набор на ручном программаторе

STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	4	5	0	0	↵			
OR	SHFT	E	SHFT	V	1	4	0	2
K(CON)	2	3	4	5	↵			
OUT	Y(OUT)	3	↵					

В следующем примере, когда значение в ячейке памяти V1400 = 3916 или в ячейке V1402 ≠ 2500, Y3 сработает.

*Direct*SOFT



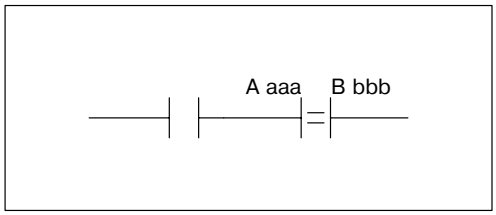
Набор на ручном программаторе

STR	SHFT	E	SHFT	V	1	4	0	0	
K(CON)	3	9	1	6	↵				
OR	NOT	SHFT	E	SHFT	V	1	4	0	2
K(CON)	2	5	0	0	↵				
OUT	Y(OUT)	3	↵						

And If Equal (ANDE)

√	√	√
430	440	450

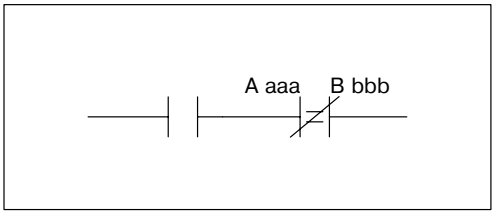
Команда And If Equal соединяет нормально открытый контакт сравнения последовательно с другим контактом. Контакт будет включен, когда Vaaa = Bbbb



And If Not Equal (ANDNE)

√	√	√
430	440	450

Команда And If Not Equal соединяет нормально закрытый контакт сравнения последовательно с другим контактом. Контакт будет включен, когда Vaaa ≠ Bbbb.



Тип данных операнда		Диапазон DL430		Диапазон DL440		Диапазон DL450	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-41)	Все (см. стр. 3-42)	Все (см. стр. 3-42)
Указатель	P	—	—	Все (см. стр. 3-41)	Все (см. стр. 3-41)	Все (см. стр. 3-42)	Все (см. стр. 3-42)
Константа	K	—	0-FFFF	—	0-FFFF	—	0-FFFF

В следующем примере, когда значение в ячейке памяти V1400 = 5000 и в ячейке V1402 = 2345, Y3 сработает.

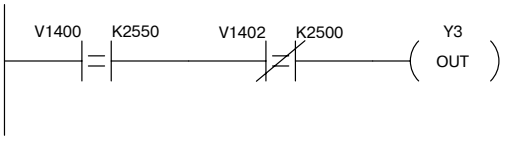
*Direct*SOFT

Набор на ручном программаторе

STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	5	0	0	0	↵			
AND	SHFT	E	SHFT	V	1	4	0	2
K(CON)	2	3	4	5	↵			
OUT	Y(OUT)	3	↵					

В следующем примере, когда значение в ячейке памяти V1400 = 2550 и в ячейке V1402 ≠ 2500, Y3 сработает.

*Direct*SOFT



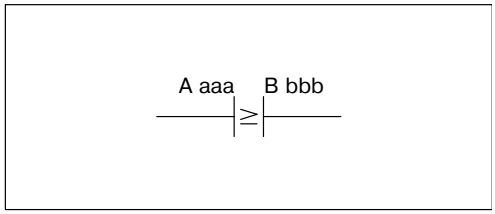
Набор на ручном программаторе

STR	SHFT	E	SHFT	V	1	4	0	0	
K(CON)	2	5	5	0	↵				
AND	NOT	SHFT	E	SHFT	V	1	4	0	2
K(CON)	2	5	0	0	↵				
OUT	Y(OUT)	3	↵						

Store (STR)

√	√	√
430	440	450

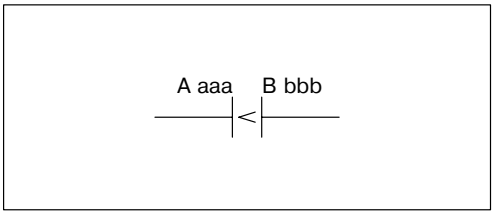
Команда Comparative Store начинает новую цепь или дополнительную ветвь в цепи с нормально открытым контактом сравнения. Контакт будет включен, когда Aaaa ≥ Bbbb.



Store Not (STRN)

√	√	√
430	440	450

Команда Comparative Store Not начинает новую цепь или дополнительную ветвь в цепи с нормально закрытым контактом сравнения. Контакт будет включен, когда Aaaa < Bbbb.



Тип данных операнда		Диапазон DL430		Диапазон DL440		Диапазон DL450	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-41)	Все (см. стр. 3-42)	Все (см. стр. 3-42)
Указатель	P	—	—	Все (см. стр. 3-41)	Все (см. стр. 3-41)	Все (см. стр. 3-42)	Все (см. стр. 3-42)
Константа	K	—	0-FFFF	—	0-FFFF	—	0-FFFF
Таймер	T	0-177	—	0-377	—	0-377	
Счетчик	CT	0-177	—	0-177	—	0-377	

В следующем примере, когда значение в ячейке памяти V1400 ≥ 1000, Y3 сработает.

*Direct*SOFT

Набор на ручном программаторе

STR	V	1	4	0	0
K(CON)	1	0	0	0	↵
OUT	Y(OUT)	3	↵		

В следующем примере, когда значение в ячейке памяти V1400 < 4050, Y3 сработает.

*Direct*SOFT

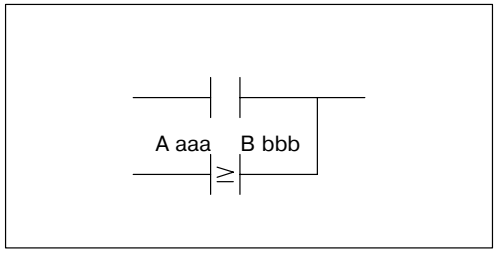
Набор на ручном программаторе

STR	NOT	V	1	4	0	0
K(CON)	4	0	5	0	↵	
OUT	Y(OUT)	3	↵			

Or (OR)

√	√	√
430	440	450

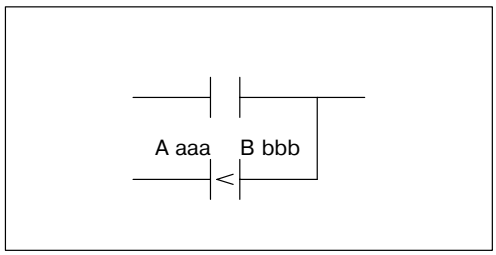
Команда Comparative Or соединяет нормально открытый контакт сравнения параллельно с другим контактом. Контакт будет включен, когда Aaaa ≥ Bbbb.



Or Not (ORN)

√	√	√
430	440	450

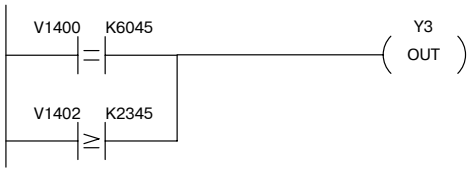
Команда Comparative Or Not соединяет нормально открытый контакт сравнения параллельно с другим контактом. Контакт будет включен, когда Aaaa < Bbbb.



Тип данных операнда		Диапазон DL430		Диапазон DL440		Диапазон DL450	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-41)	Все (см. стр. 3-42)	Все (см. стр. 3-42)
Указатель	P	—	—	Все (см. стр. 3-41)	Все (см. стр. 3-41)	Все (см. стр. 3-42)	Все (см. стр. 3-42)
Константа	K	—	0-FFFF	—	0-FFFF	—	0-FFFF
Таймер	T	0-177	—	0-377	—	0-377	
Счетчик	CT	0-177	—	0-177	—	0-377	

В следующем примере, когда значение в ячейке памяти V1400 = 6045 или значение в ячейке V1402 ≥ 2345, Y3 сработает.

*Direct*SOFT

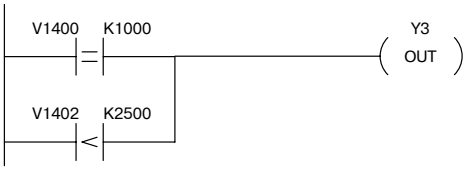


Набор на ручном программаторе

STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	6	0	4	5	↵			
OR	V	1	4	0	2			
K(CON)	2	3	4	5	↵			
OUT	Y(OUT)	3	↵					

В следующем примере, когда значение в ячейке памяти V1400 = 1000 или значение в ячейке V1402 < 2500, Y3 сработает.

*Direct*SOFT



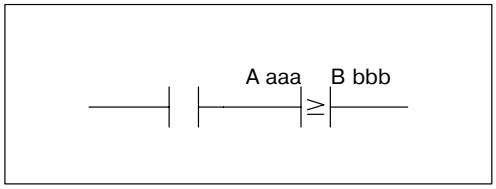
Набор на ручном программаторе

STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	1	0	0	0	↵			
OR	NOT	V	1	4	0	2		
K(CON)	2	5	0	0	↵			
OUT	Y(OUT)	3	↵					

And (AND)

√	√	√
430	440	450

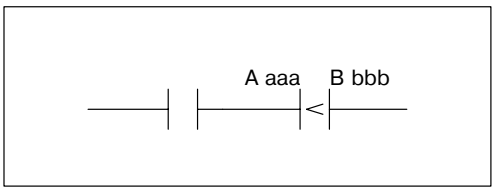
Команда Comparative And соединяет нормально открытый контакт сравнения последовательно с другим контактом. Контакт будет включен, когда Aaaa ≥ Bbbb.



And Not (ANDN)

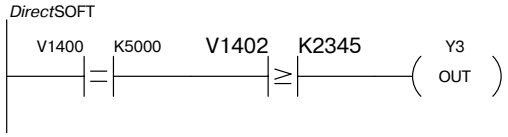
√	√	√
430	440	450

Команда Comparative And Not соединяет нормально открытый контакт сравнения последовательно с другим контактом. Контакт будет включен, когда Aaaa < Bbbb.



Тип данных операнда		Диапазон DL430		Диапазон DL440		Диапазон DL450	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-41)	Все (см. стр. 3-42)	Все (см. стр. 3-42)
Указатель	P	—	—	Все (см. стр. 3-41)	Все (см. стр. 3-41)	Все (см. стр. 3-42)	Все (см. стр. 3-42)
Константа	K	—	0-FFFF	—	0-FFFF	—	0-FFFF
Таймер	T	0-177	—	0-377	—	0-377	
Счетчик	CT	0-177	—	0-177	—	0-377	

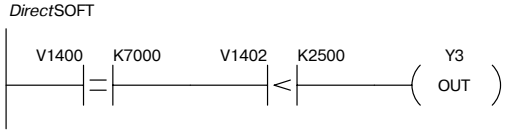
В следующем примере, когда значение в ячейке памяти V1400 = 5000 и значение в ячейке V1402 ≥ 2345, Y3 сработает.



Набор на ручном программаторе

STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	5	0	0	0	↵			
AND	V	1	4	0	2			
K(CON)	2	3	4	5	↵			
OUT	Y(OUT)	3	↵					

В следующем примере, когда значение в ячейке памяти V1400 = 7000 и значение в ячейке V1402 < 2500, Y3 сработает.



Набор на ручном программаторе

STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	7	0	0	0	↵			
AND	NOT	V	1	4	0	2		
K(CON)	2	5	0	0	↵			
OUT	Y(OUT)	3	↵					

Немедленные команды

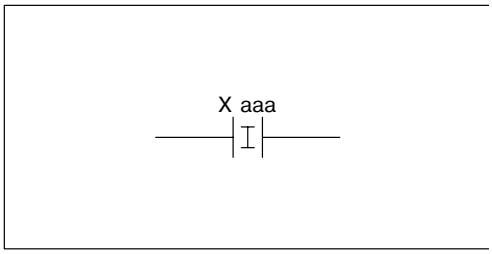
Store Immediate (STRI)

√	√	√
430	440	450

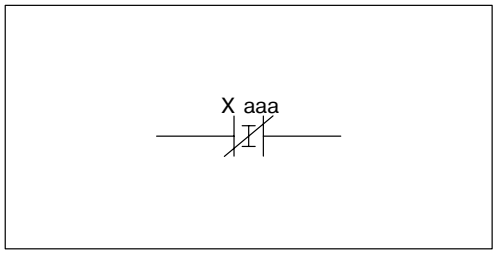
Store Not Immediate (STRNI)

√	√	√
430	440	450

Команда Store Immediate начинает новую цепь или дополнительную ветвь в цепи. Состояние контакта будет таким же, как состояние соответствующей входной точки модуля во время, когда команда выполняется. Регистр отображения не обновляется.



Команда Store Not Immediate начинает новую цепь или дополнительную ветвь в цепи. Состояние контакта будет противоположным состоянию соответствующей входной точки модуля во время, когда команда выполняется. Регистр отображения не изменяется.



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**bb**	**bb**
Входы	X	0-477	0-477	0-1777

В следующем примере, когда X1 включен, Y2 будет включен.

*Direct*SOFT

Набор на ручном программаторе

STR	SHFT	I	SHFT	X(IN)	1	↵
OUT	Y(OUT)	2	↵			

В следующем примере, когда X1 выключен, Y2 будет включен.

*Direct*SOFT

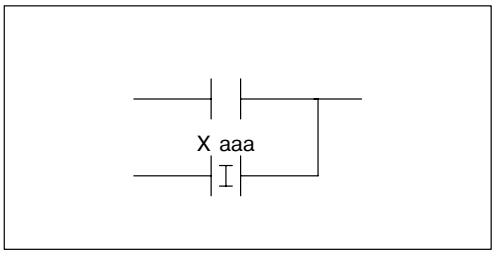
Набор на ручном программаторе

STR	NOT	SHFT	I	SHFT	X(IN)	1	↵
OUT	Y(OUT)	2	↵				

Or Immediate (ORI)

√	√	√
430	440	450

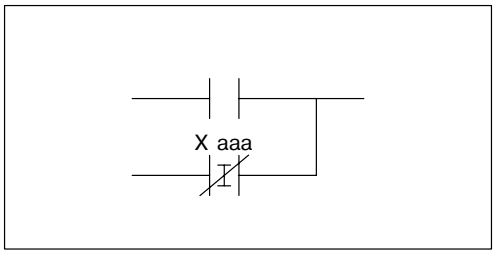
Команда Or Immediate соединяет два контакта параллельно. Состояние контакта будет таким же, как состояние соответствующей входной точки модуля во время, когда команда выполняется. Регистр отображения не изменяется.



Or Not Immediate (ORNI)

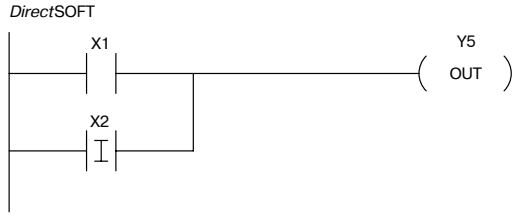
√	√	√
430	440	450

Команда Or Not Immediate соединяет два контакта параллельно. Состояние контакта будет противоположным состоянию соответствующей входной точки модуля во время, когда команда выполняется. Регистр отображения не обновляется.



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**bb**	**bb**
Входы	X	0-477	0-477	0-1777

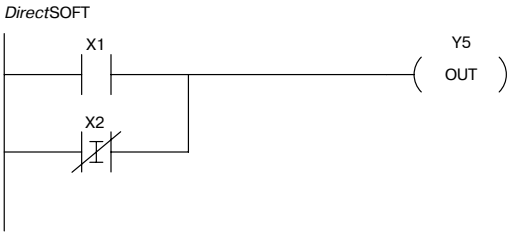
В следующем примере, когда X1 или X2 включены, Y5 будет включен.



Набор на ручном программаторе

STR	X(IN)	1	↵			
OR	SHFT	I	SHFT	X(IN)	2	↵
OUT	Y(OUT)	5	↵			

В следующем примере, когда X1 или X2 выключены, Y5 будет включен.



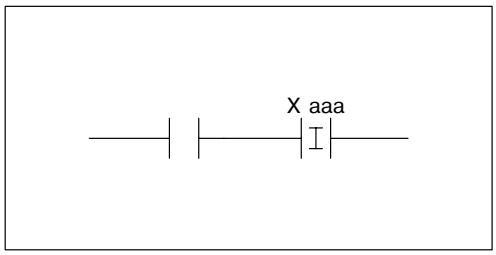
Набор на ручном программаторе

STR	X(IN)	1	↵				
OR	NOT	SHFT	I	SHFT	X(IN)	2	↵
OUT	Y(OUT)	5	↵				

And Immediate (ANDI)

√	√	√
430	440	450

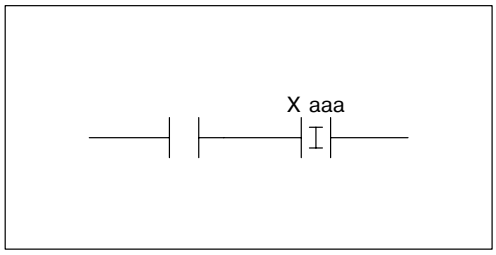
Команда And Immediate соединяет два контакта последовательно. Состояние контакта будет таким же, как состояние соответствующей входной точки модуля во время, когда команда выполняется. Регистр отображения не обновляется.



And Not Immediate (ANDNI)

√	√	√
430	440	450

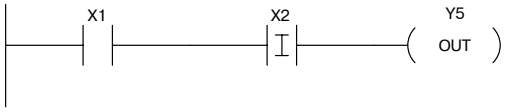
Команда And Not Immediate соединяет два контакта последовательно. Состояние контакта будет противоположным состоянию соответствующей входной точки модуля во время выполнения команды. Регистр отображения не обновляется.



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**bb**	**bb**
Входы	X	0-477	0-477	0-1777

В следующем примере, когда X1 или X2 включены, Y5 будет включен.

*Direct*SOFT

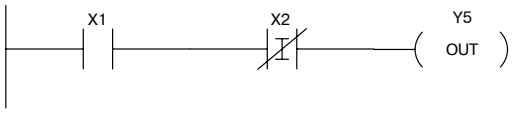


Набор на ручном программаторе

STR	X(IN)	1	↵				
AND	SHFT	I	SHFT	X(IN)	2	↵	
OUT	Y(OUT)	5	↵				

В следующем примере, когда X1 включен и X2 выключен, Y5 будет включен.

*Direct*SOFT



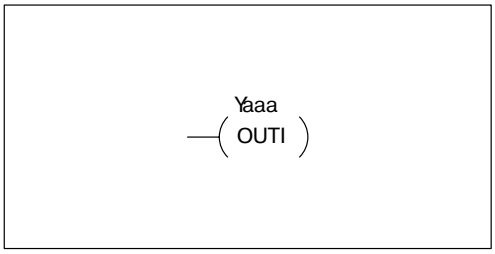
Набор на ручном программаторе

STR	X(IN)	1	↵				
AND	NOT	SHFT	I	SHFT	X(IN)	2	↵
OUT	Y(OUT)	5	↵				

Out Immediate (OUTI)

√	√	√
430	440	450

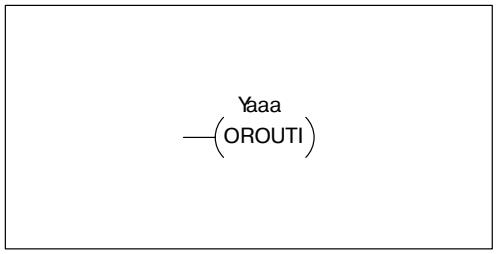
Команда Out Immediate отражает состояние цепи (вкл/выкл) и выводит дискретное состояние в определенную точку выхода модуля и регистр отображения во время выполнения команды. Можно многократно использовать команду Out Immediate, ссылающуюся на одну и ту же дискретную точку, для изменения состояния выходного модуля несколько раз за цикл сканирования процессора. Смотри Or Out Immediate.



Or Out Immediate (OROUTI)

√	√	√
430	440	450

Команда Or Out Immediate была разработана, чтобы использовать несколько цепей дискретной логики для управления одним выходом. Можно многократно использовать команду Or Out Immediate, ссылающуюся на одну и ту же выходную обмотку, так как операция ИЛИ выполняется вместе над всеми контактами, управляющими выходом. Если какая-либо цепь находится в состоянии ВКЛ. во время выполнения команды, выход будет также включен.



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**bb**	**bb**
Выходы	Y	0-477	0-477	0-1777

В следующем примере, когда X1 включен, выходная точка Y2 в выходном модуле будет включен.

Набор на ручном программаторе

STR	X(IN)	1	⏎					
OUT	SHFT	I	SHFT	Y(OUT)	2	⏎		

В следующем примере, когда X1 или X4 включены, Y2 будет включен.

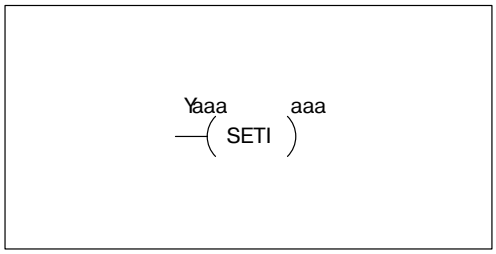
Набор на ручном программаторе

STR	X(IN)	1	⏎						
RST	SHFT	I	SHFT	Y(OUT)	5	Y(OUT)	2	2	⏎

Set Immediate (SETI)

√	√	√
430	440	450

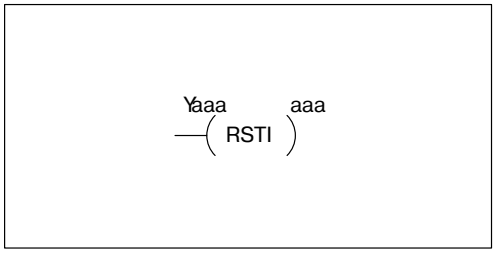
Команда Set Immediate немедленно устанавливает (включает) выход или группу выходов в регистре отображения и в соответствующем модуле (модулях) выхода во время выполнения команды. Если выходы установлены, то необязательно входам оставаться включенными. Команда Reset Immediate может использоваться для сброса выходов.



Reset Immediate (RSTI)

√	√	√
430	440	450

Команда Reset Immediate немедленно сбрасывает (выключает) выход или группу выходов и выходной модуль (модули) во время выполнения команды. Регистр отображения не обновляется. Если выходы сброшены, то необязательно входам оставаться включенными.



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**bb**	**bb**
Выходы	Y	0-477	0-477	0-1777

В следующем примере, когда X1 включен, Y5-Y22 будут установлены в состояние ВКЛ. для соответствующего) выходных модуля (модулей.

Набор на ручном программаторе

STR	X(IN)	1	↵						
SET	SHFT	I	SHFT	Y(OUT)	5	Y(OUT)	2	2	↵

В следующем примере, когда X1 включен, Y5-Y22 будут сброшены (выключены) для соответствующего выходного модуля (модулей).

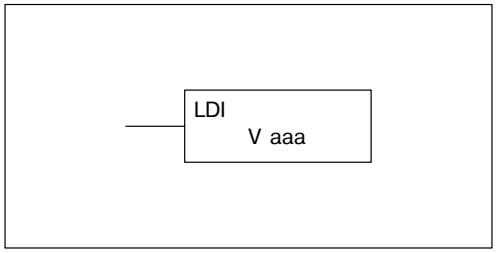
Набор на ручном программаторе

STR	X(IN)	1	↵						
RST	SHFT	I	SHFT	Y(OUT)	5	Y(OUT)	2	2	↵

Load Immediate (LDI)

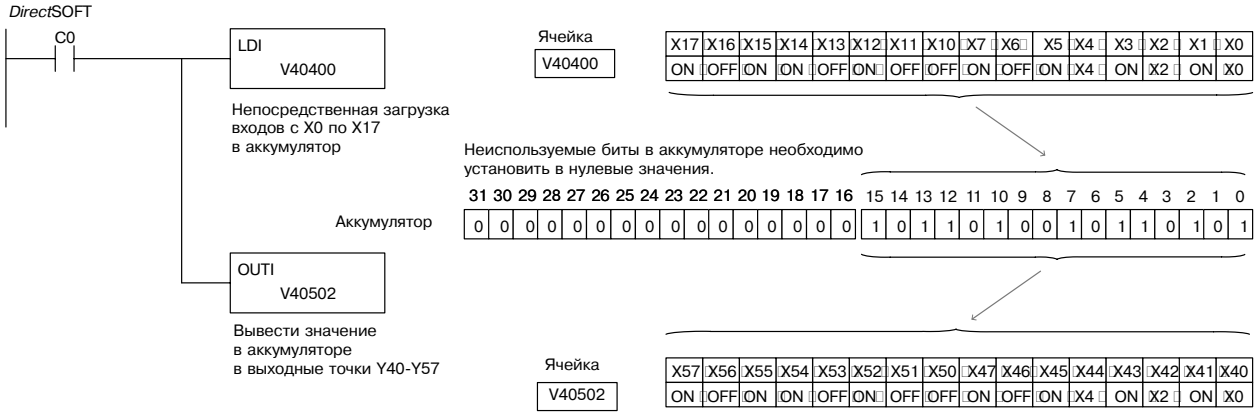
X	X	√
430	440	450

Команда Load Immediate загружает 16-битовое значение V-памяти в аккумулятор. Доступная область адресов включает адреса всех входных точек локального каркаса. Это значение отражает текущее состояние входных точек в процессе выполнения команды. Эта команда может использоваться вместо команды LDIF, которая требует, чтобы Вы определили число входных точек.



Тип данных операнда		Диапазон DL450
		aaaaa
Входы	V	40400 -40477

В следующем примере, когда C0 включен, набор двоичных разрядов X10 - X17 будет загружен в аккумулятор с помощью команды Load Immediate. Может использоваться команда Out Immediate для копирования 16 битов аккумулятора в выходные точки, такие как Y40-Y57. Данный метод полезен для быстрого копирования входной комбинации в выходные точки (не ожидая выполнения всего цикла сканирования процессора).



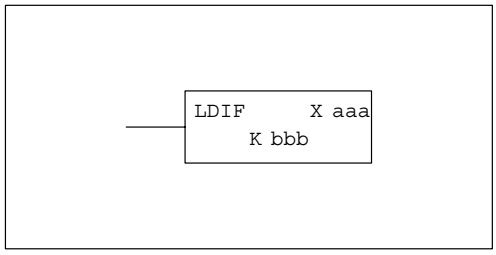
Набор на ручном программаторе

STR	C(CR)	0	⏎							
LD	SHFT	I	SHFT	V	4	0	4	0	0	⏎
OUT	SHFT	I	SHFT	V	4	0	5	4	0	⏎

Load Immediate Formatted (LDIF)

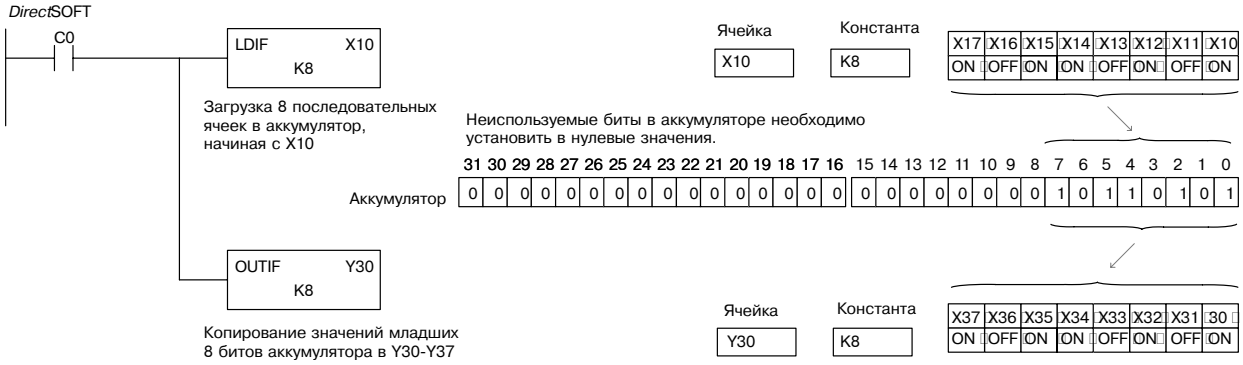
X	√	√
430	440	450

Команда Load Immediate Formatted загружает двоичное значение 1-32 разрядов в аккумулятор. Значение отражает текущее состояние входного модуля (модулей) во время выполнения команды. Неиспользуемые командой биты аккумулятора устанавливаются в нулевые значения.



Тип данных операнда		Диапазон DL440		Диапазон DL450	
		aaa	bbb	aaa	bbb
Входы	X	0-477	0-477	0-1777	—
Константа	K	—	1 - 32	—	1 - 32

В следующем примере, когда C0 включено, набор двоичных разрядов X10-X17 будет загружен в аккумулятор с помощью команды Load Immediate Formatted. Может использоваться команда Out Immediate Formatted для копирования определенного числа битов аккумулятора в заданные выходные точки выходного модуля, такие как Y30-Y37. Данный метод полезен для быстрого копирования входной комбинации в выходы (не ожидая выполнения всего цикла сканирования процессора).



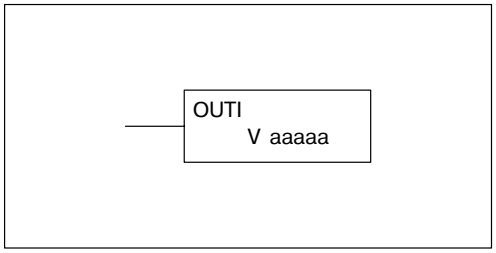
Набор на ручном программаторе

STR	C(CR)	0	↵							
LD	SHFT	I	F	SHFT	X(IN)	1	0	K(CON)	8	↵
OUT	SHFT	I	F	SHFT	Y(OUT)	3	0	K(CON)	8	↵

Out Immediate (OUTI)

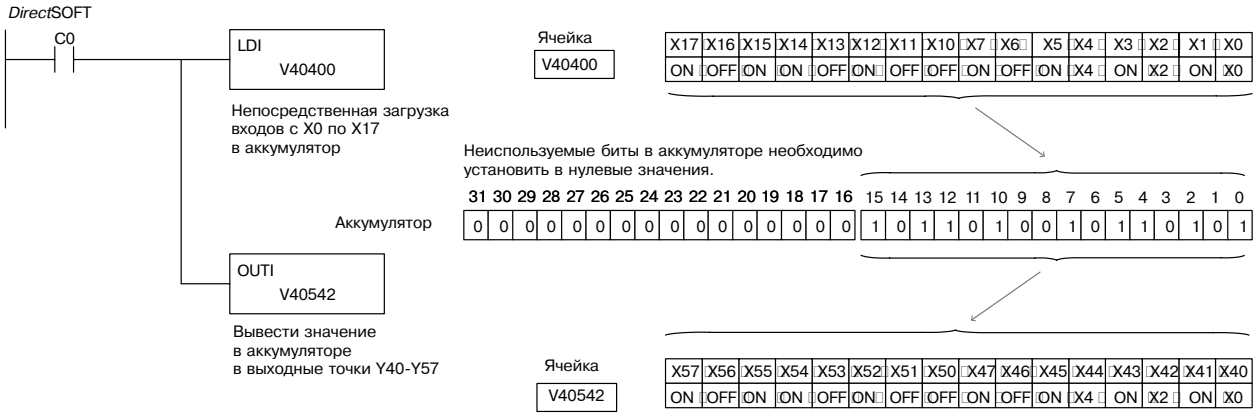
X	X	√
430	440	450

Команда Out Immediate выводит 16-битовое значение из аккумулятора в адрес V-памяти в процессе выполнения команды. Доступная область адресов включает адреса всех выходных точек локального каркаса. Это значение отражает текущее состояние входных точек в процессе выполнения команды. Эта команда может использоваться вместо команды OUTIF, которая требует, чтобы Вы определили число входных точек.



Тип данных операнда		Диапазон DL450
		aaaaa
Входы	V	40400 -40477

В следующем примере, когда C0 включено, набор двоичных разрядов X10 - X17 будет загружен в аккумулятор с помощью команды Load Immediate. Может использоваться команда Out Immediate для копирования 16 битов аккумулятора в выходные точки, такие как Y40-Y57. Данный метод полезен для быстрого копирования входной комбинации в выходы точки (не ожидая выполнения всего цикла сканирования процессора).



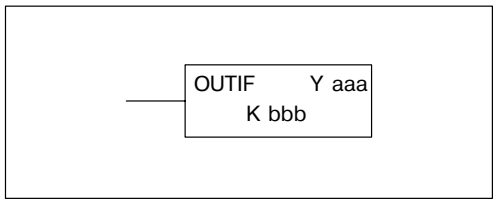
Набор на ручном программаторе

STR	C(CR)	0	⟵							
LD	SHFT	I	SHFT	V	4	0	4	0	0	⟵
OUT	SHFT	I	SHFT	V	4	0	5	4	0	⟵

Out Immediate Formatted (OUTIF)

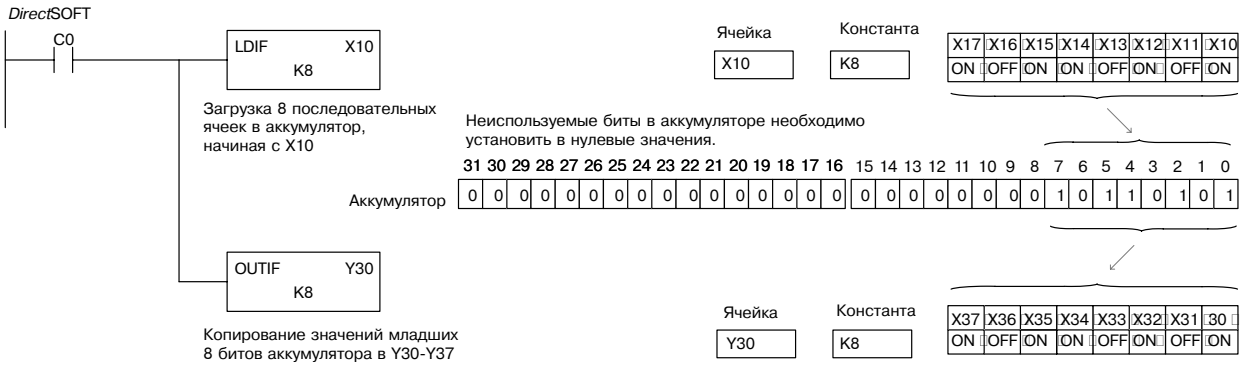
X	√	√
430	440	450

Команда Out Immediate Formatted выводит двоичное значение битов 1-32 из аккумулятора в заданные выходные точки во время выполнения команды. Неиспользуемые командой биты аккумулятора устанавливаются в нулевые значения.



Тип данных операнда		Диапазон DL440		Диапазон DL450	
		aaa	bbb	aaa	bbb
Входы	X	0-477	0-477	0-1777	—
Константа	K	—	1 - 32	—	1 - 32

В следующем примере, когда C0 включено, набор двоичных разрядов X10 -X17 будет загружен в аккумулятор с помощью команды Load Immediate Formatted. Набор двоичных разрядов аккумулятора записывается в Y30-Y37 с использованием команды Out Immediate Formatted. Данный метод полезен для быстрого копирования входной комбинации в выходы (не ожидая выполнения всего цикла сканирования процессора).



Набор на ручном программаторе

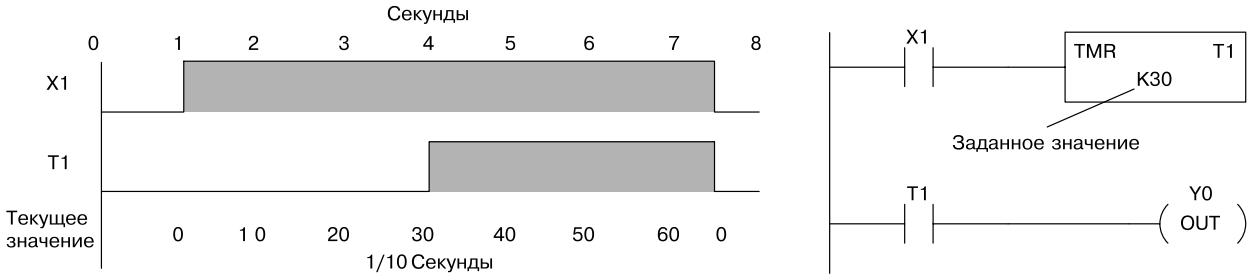
STR	C(CR)	0	↵							
LD	SHFT	I	F	SHFT	X(IN)	1	0	K(CON)	8	↵
OUT	SHFT	I	F	SHFT	Y(OUT)	3	0	K(CON)	8	↵

Команды таймера, счетчика и регистра сдвига

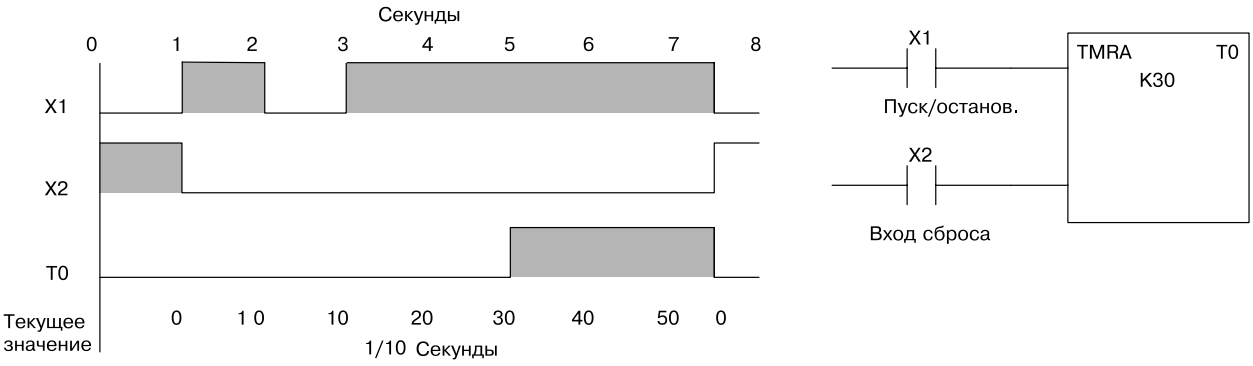
Использование таймеров

Таймеры используются для хронометрирования событий в целях получения желаемого периода времени. Существуют такие приложения, которым требуется накапливающий таймер, способный измерять время, останавливаться и затем продолжать отсчет с момента, когда он был остановлен.

Таймер с одним входом будет считать время столько, сколько вход находится во включенном состоянии. Когда состояние входа изменится на противоположное, текущее значение таймера сбросится в 0. Существуют таймеры с максимальным временем 999.9 с и 99.99 с и с разрешением в десятые доли секунды и сотые доли секунды соответственно. Существуют биты, связанные с каждым таймером, показывающие, что текущее значение равно или больше заданного значения. Временная диаграмма ниже показывает связь между входом таймера, связанным с ним битом, текущим значением и заданным значением.



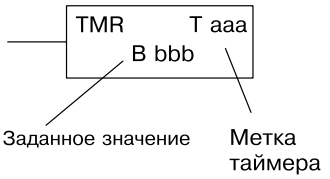
Таймер-накопитель работает подобно обычному таймеру, но он имеет два входа. Вход ”старт/стоп” запускает и останавливает таймер. Когда таймер остановлен, пройденное время сохраняется. Когда таймер запускается снова, счет времени продолжается с пройденного времени. Когда происходит сброс входа, пройденное время сбрасывается, и таймер при повторном запуске начнет считать с нуля. Существуют таймеры с точностью в десятые и сотые доли секунды при максимальном времени счета 9999999.9 с и 999999.99 с соответственно. Временная диаграмма ниже показывает связь между входом таймера, связанным с ним битом, сбросом таймера, текущим значением и заданным значением.

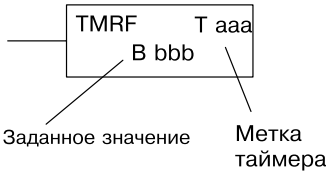


Timer (TMR) and Timer Fast (TMRF)

√	√	√
430	440	450

Команда Timer - это 0.1 секундный таймер с одним входом, который считает до 999.9 секунд максимум. Timer Fast - это 0.01 секундный таймер с одним входом, который считает до 99.99с максимум. Эти таймеры будут работать, если входная логика истинна (вкл.) и будут сброшены в 0, если входная логика ложна (выкл.).





Бит дискретного состояния таймера и текущее значение не определяются в команде таймера.

Спецификации команды

Метка таймера (Taaa): Определяет номер таймера.

Заданное значение (Bbbb): Константа (K) или ячейка V-памяти. (V-ячейки являются 16-битовыми словами).

Текущее значение: Текущие значения таймера доступны при обращении к соответствующим ячейкам V- или T-памяти*. Например, текущее значение таймера для T3 физически находится в ячейке V-памяти V3. (V-ячейки являются 16-битовыми словами).

Бит дискретного состояния: бит дискретного состояния доступен при обращении к соответствующей ячейке T-памяти. Он будет ВКЛ. если текущая величина равна или больше заданного значения. Например, бит состояния для таймера 2 был бы T2.

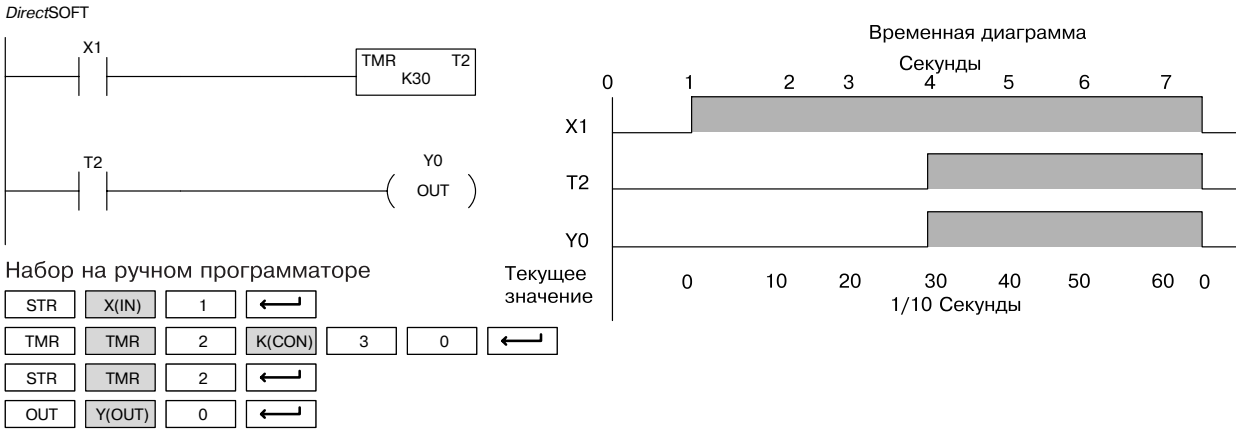
Тип данных операнда		Диапазон DL430		Диапазон DL440		Диапазон DL450	
	B	**aaa**	**bbb**	**aaa**	**bbb**	**aaa**	**bbb**
Таймеры	T	0-177	--	0-377	--	0-377	--
V-память для заданных значений	V	--	1400-7377	--	1400–7377 10000–17777	--	1400–7377 10000–17777
Пойнтеры-указатели (только заданные)	P	--	--	--	1400–7377 10000–17777	--	1400–7377 10000–17777
Константы (только заданные)	K	--	0–9999	--	0–9999	--	0–9999
Дискретные биты состояния таймера	T/V	0-177		0-377		0–377	
Текущие значения таймера	V /T*	0-177		0-377		0–377	

Существуют два метода программирования таймеров. Вы можете выполнять функции, когда таймер достигнет определенного заданного значения, используя бит состояния, или использовать контакты сравнения, чтобы выполнять функции на различных временных интервалах, используя один таймер. Следующие примеры показывают каждый метод использования таймеров.

ПРИМЕЧАНИЕ. * На ручном программаторе доступны как биты дискретных состояний таймера, так и текущие значения с одним типом данных (пример T2). Способ использования типа данных зависит от того, является ли это бит состояния или текущее значение. Для любой команды сравнения, использующей T2, будет доступно текущее значение, для всех других команд доступен бит состояния. Текущие значения доступны также через ячейку V-памяти. В DirectSOFT использование T2 относится к биту дискретного состояния таймера. Вы должны использовать V2 (или альтернативное имя TA2), чтобы обратиться к текущему значению.

Пример таймера, использующего биты дискретного состояния

В следующем примере таймер с одним входом используется с заданным значением 3 с. Бит дискретного состояния таймера (T2) включится, когда таймер отсчитает 3 с. Таймер сбрасывается, когда X1 выключится, выключая бит состояния и сбрасывая текущее значение таймера в 0.

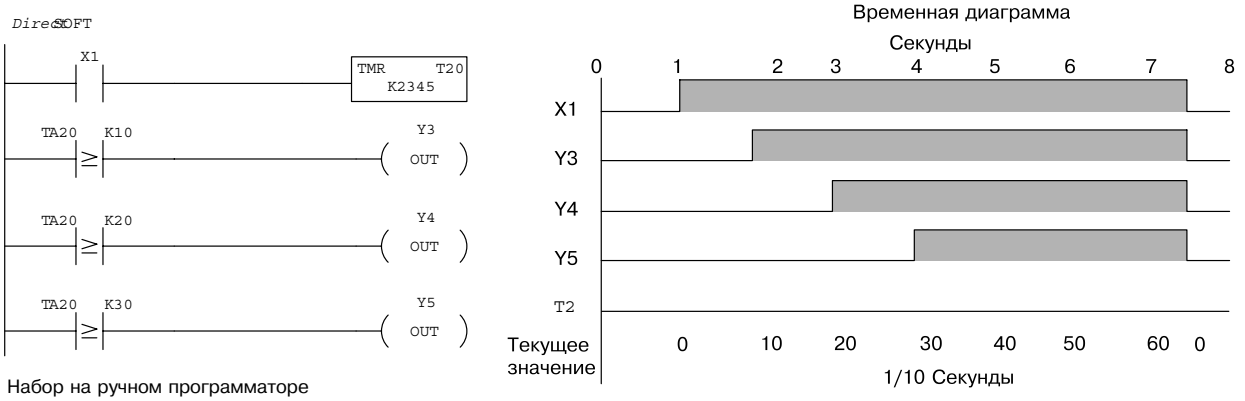


Набор на ручном программаторе

STR	X(IN)	1	↵			
TMR	TMR	2	K(CON)	3	0	↵
STR	TMR	2	↵			
OUT	Y(OUT)	0	↵			

Пример таймера, использующего контакты сравнения

В следующем примере таймер с одним входом используется с заданным значением 4.5 с. Контакты сравнения используются для включения Y3, Y4 и Y5 с интервалом в 1 с соответственно. Когда X1 выключится, таймер будет сброшен в 0 и контакты сравнения выключат Y3, Y4 и Y5.



Набор на ручном программаторе

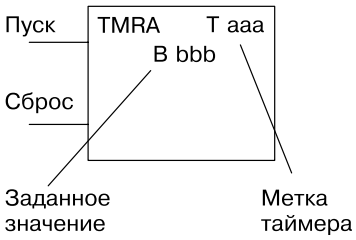
STR	X(IN)	1	↵						
TMR	TMR	2	0	K(CON)	2	3	4	5	↵
STR	TMR	2	0	K(CON)	1	0	↵		
OUT	Y(OUT)	3	↵						
STR	TMR	2	0	K(CON)	2	0	↵		
OUT	Y(OUT)	4	↵						
STR	TMR	2	0	K(CON)	3	0	↵		
OUT	Y(OUT)	5	↵						

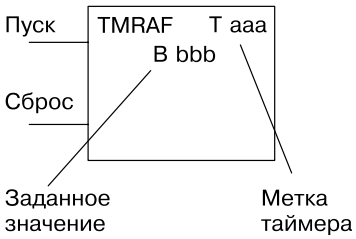
ПРИМЕЧАНИЕ. Когда этот пример показывается из DirectSOFT, Вы должны использовать альтернативное имя TA20 (или V20) вместо T20, которое требуется эквивалентной цепочкой при входе с ручного программатора.

Accumulating Timer (TMRA) Accumulating Fast Timer (TMRAF)

√	√	√
430	440	450

Accumulating Timer - это 0.1 секундный таймер с двумя входами, который считает до 9999999.9 секунд максимум. Команда Accumulating Fast Timer - это 0.01 секундный таймер с двумя входами, который считает до 999999.99 секунд максимум. Эти таймеры имеют два входа: Enable (Пуск) и Reset (Сброс). Таймер начнет отсчет, когда вход Enable включится, и остановится, когда вход Enable будет выключен без сброса текущего значения в 0. Когда вход Reset включен, то он сбрасывает таймер, когда он выключен, то таймер работает.





Бит дискретного состояния таймера и текущее значение не определяются в команде таймера.

Предостережение: TMRA использует две последовательных ячейки, так как заданное значение может состоять из 8 цифр, которые требуют две ячейки V-памяти. Например, если в программе используется TMRA T0, то следующим доступным таймером будет T2. Или, если T0 был обычный таймер, и T1 был накапливающий таймер, то следующий доступный таймер будет T3.

Спецификации команды

Метка таймера (Taaa): Определяет номер таймера.
Заданное значение (Bbbb): Константа (K) или ячейка V-памяти. (V-ячейки являются 32-битовыми словами).
Текущее значение: Текущие значения таймера является двойным словом, доступным при обращении к соответствующим ячейкам V- или T-памяти*. Например, текущее значение таймера для T0 физически находится в ячейке V-памяти V0 и V1. (V-ячейки являются 16-битовыми словами).
Бит дискретного состояния: Бит дискретного состояния доступен при обращении к соответствующей ячейке T-памяти. Он будет включен, если текущее значение равно или больше заданного значения. Например, бит дискретного состояния для таймера 2 был бы T2.

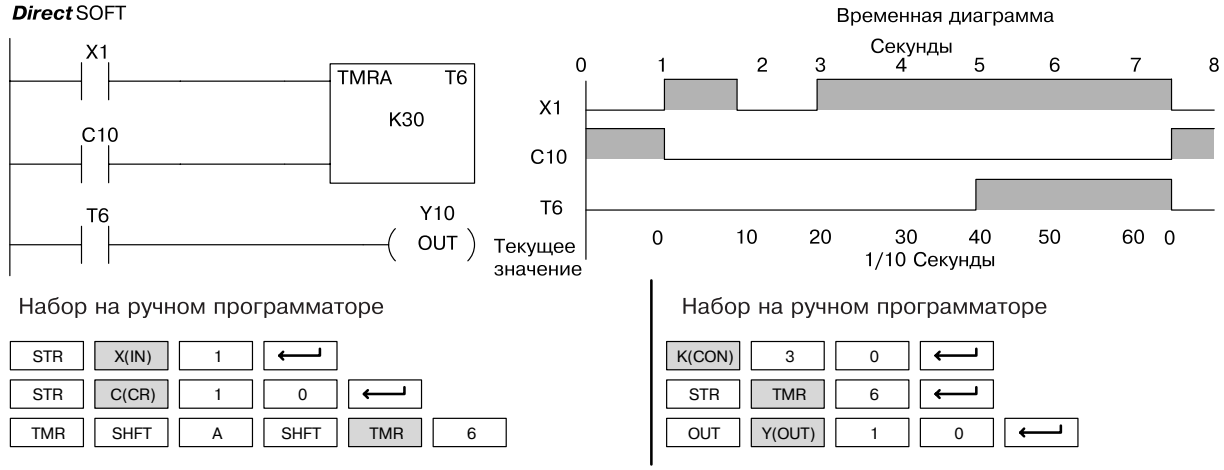
Существуют два метода программирования таймеров. Вы можете выполнять функции, когда таймер достигнет определенного заданного значения, используя бит дискретного состояния, или использовать контакты сравнения, чтобы выполнять функции на различных временных интервалах, используя один таймер. Следующие примеры показывают каждый метод использования таймеров.

Тип данных операнда		Диапазон DL430		Диапазон DL440		Диапазон DL450	
	B	**aaa**	**bbb**	**aaa**	**bbb**	**aaa**	**bbb**
Таймеры	T	0-177	––	0-377	––	0-377	––
V-память для заданных значений	V	––	1400-7377	––	1400–7377 10000–17777	––	1400–7377 10000–17777
Пойнтеры-указатели (только заданные)	P	––	––	––	1400–7377 10000–17777	––	1400–7377 10000–17777
Константы (только заданные)	K	––	0-99999999	––	0-99999999	––	0-99999999
Дискретные биты состояния таймера	T/V	0-177		0-377		0–377	
Текущие значения таймера	V /T*	0-177		0-377		0–377	

ПРИМЕЧАНИЕ. * На ручном программаторе доступны как биты дискретных состояний таймера, так и текущие значения с одним типом данных (пример T2). Способ использования типа данных зависит от того, является ли это бит состояния или текущее значение. Для любой команды сравнения, использующей T2, будет доступно текущее значение, для всех других команд доступен бит состояния. Текущие значения доступны также через ячейку V-памяти. В DirectSOFT использование T2 относится к биту дискретного состояния таймера. Вы должны использовать V2 (или альтернативное имя TA2), чтобы обратиться к текущему значению.

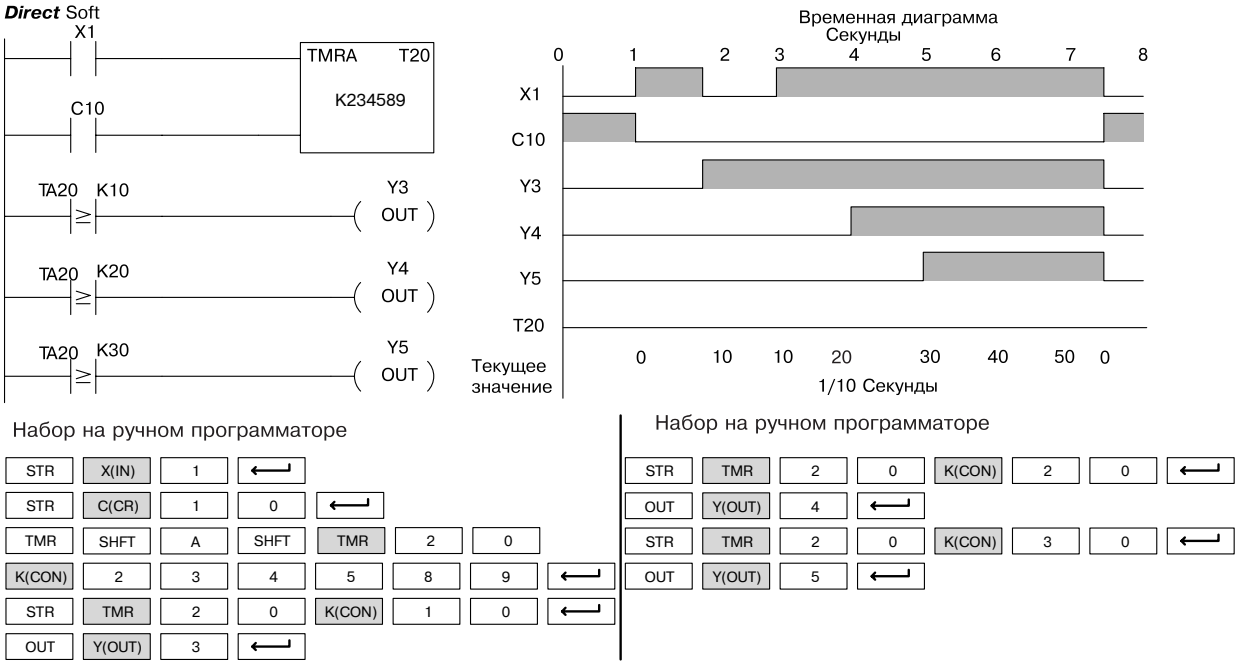
Пример Накапливающего Таймера (Accumulating Timer), использующего биты дискретного состояния

В следующем примере, таймер с двумя входами (Накапливающий таймер - Accumulating Timer) используется с заданным значением 3 секунды. Бит дискретного состояния таймера (T6) включится, когда таймер отсчитает 3 с. Обратите внимание, в этом примере таймер работает 1 секунду, останавливается на 1 секунду, затем продолжает счет. Таймер будет сброшен, когда включится C10 после 5.4 секунд, выключив тем самым бит состояния и сбросив текущее значение таймера в 0.



Пример Накапливающего Таймера (Accumulating Timer), использующего контакты сравнения

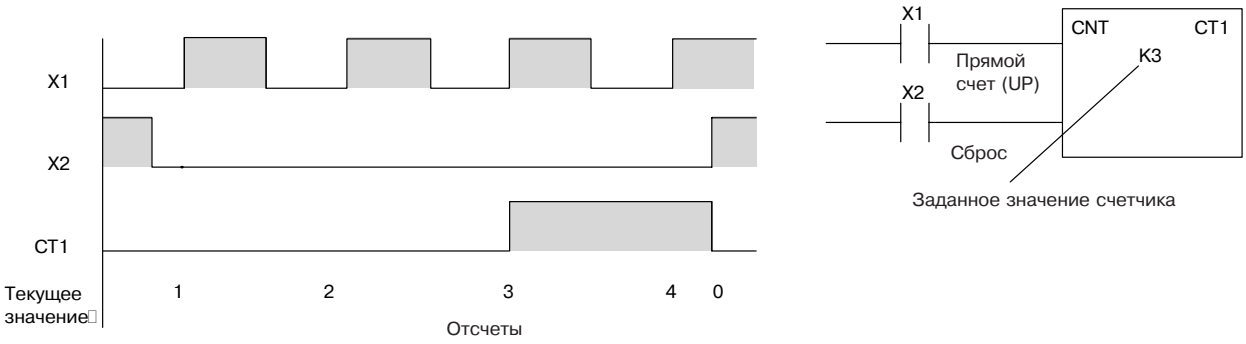
В следующем примере, таймер с одним входом используется с заданным значением 23458.9 с. Контакты сравнения используются, чтобы включить Y3, Y4 и Y5 с интервалом в 1 с соответственно. Контакты сравнения выключатся, когда таймер будет сброшен.



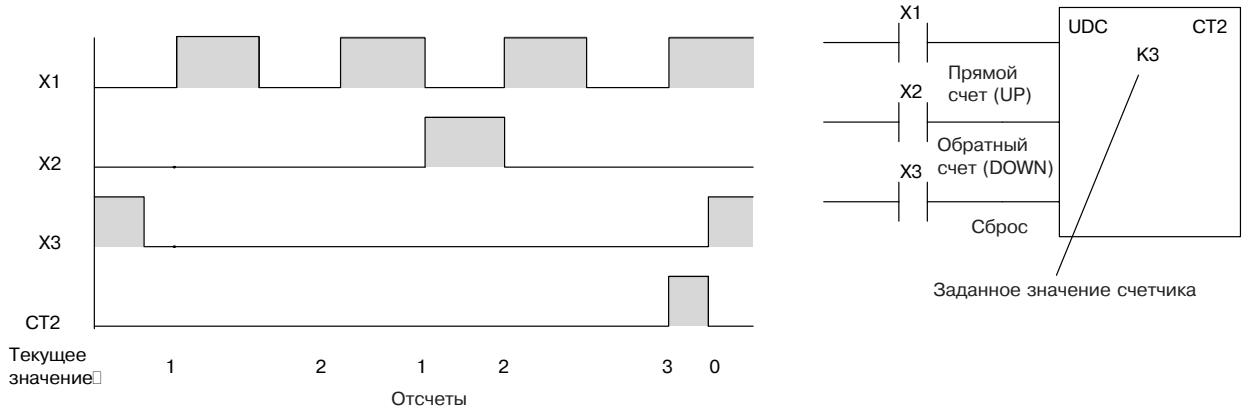
ПРИМЕЧАНИЕ. Когда этот пример показывается из DirectSOFT, Вы должны использовать альтернативное имя TA20 (или V20) вместо T20, которое требуется эквивалентной для цепи при вводе с ручного программатора.

Использование счетчиков

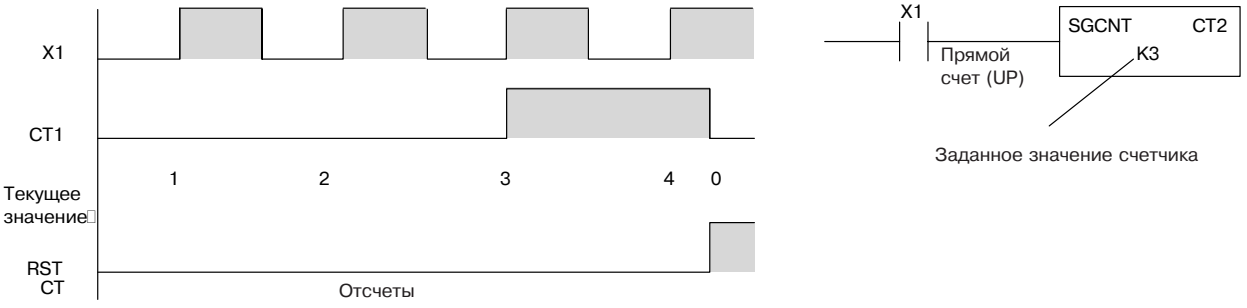
Счетчики применяются для счета событий. Имеют место суммирующие счетчики, реверсивные счетчики и счетчики стадий (используемые в программировании RLL^{PLUS}).



Суммирующие счетчики имеют два входа, вход счета и вход сброса. Максимальное число для подсчета равно 9999. На временной диаграмме ниже показано отношение между входом счетчика, сбросом счетчика, соответствующим дискретным битом, текущим значением и заданным значением счетчика.



Реверсивный счетчик имеет три входа: вход прямого счета (UP), вход обратного счета (DOWN) и вход сброса. На временной диаграмме ниже показано отношение между входом счетчика, сбросом счетчика, соответствующим дискретным битом, текущим значением и заданным значением счетчика.

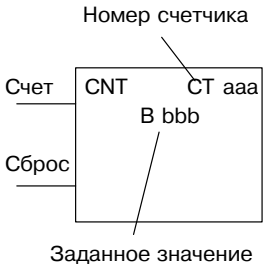


Счетчик стадий имеет вход счета и сброс с помощью команды RST. Эта команда полезна при программировании на RLL^{PLUS}, позволяя вам обращаться к одному и тому же счетчику из многих стадий. Максимальное число для подсчета равно 9999. На временной диаграмме ниже показано отношение между входом счетчика, соответствующим дискретным битом, текущим значением, заданным значением счетчика и командой сброса.

Counter (CNT)

√	√	√
430	440	450

Counter - это счетчик с двумя входами, который увеличивается, когда вход счета (Count) имеет логический переход из состояния ВЫКЛ. в состояние ВКЛ. Когда включается вход сброса (Reset), счетчик сбрасывается в 0. Когда текущее значение равно заданному значению, бит состояния счетчика включается, а счетчик продолжает считать до максимального значения 9999. Максимальное значение будет сохраняться до тех пор, пока счетчик не будет сброшен.



Бит дискретного состояния счетчика и текущее значение не определяются в команде счетчика.

Спецификации команды

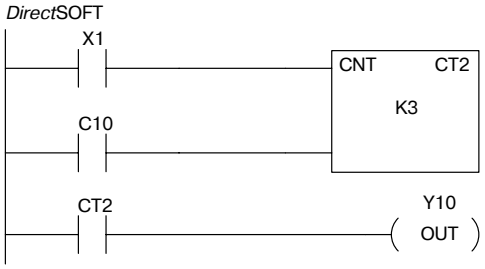
Метка счетчика (CTaaa): Указывает номер счетчика.
Заданное значение (Bbbb): Константа (K) или ячейка V-памяти. (V-ячейки являются 16-битовыми словами).
Текущие значения: Текущие значения счетчика доступны при обращении к соответствующим ячейкам V- или CT-памяти*. Ячейка V-памяти - ячейка счетчика + 1000. Например, текущее значение счетчика для CT3 находится в ячейке V-памяти V1003. (V-ячейки являются 16-битовыми словами).
Бит дискретного состояния: Бит состояния доступен при обращении к соответствующей ячейке CT-памяти. Он будет включен, если текущее значение равно или больше заданного значения. Например, бит дискретного состояния для счетчика 2 был бы CT2.

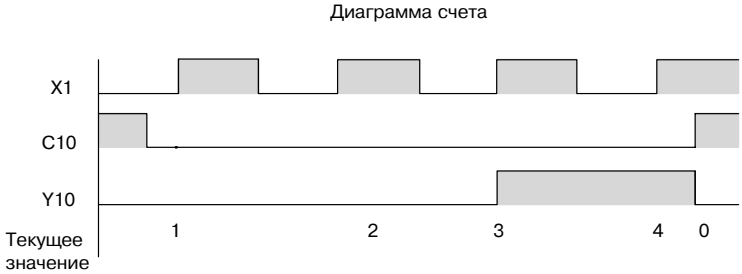
Тип данных операнда		Диапазон DL430		Диапазон DL440		Диапазон DL450	
	B	**aaa**	**bbb**	**aaa**	**bbb**	**aaa**	**bbb**
Таймеры	T	0-177	--	0-377	--	0-377	--
V-память для заданных значений	V	--	1400-7377	--	1400–7377 10000–17777	--	1400–7377 10000–17777
Пойнтеры-указатели (только заданные)	P	--	--	--	1400–7377 10000–17777	--	1400–7377 10000–17777
Константы (только заданные)	K	--	0–9999	--	0–9999	--	0–9999
Дискретные биты состояния таймера	T/V	0-177		0-377		0–377	
Текущие значения таймера	V /T*	1000-1177		1000-1177		1000-1377	

ПРИМЕЧАНИЕ. * На ручном программаторе доступны как биты дискретных состояний счетчика, так и текущие значения с одним типом данных (пример CT2). Способ использования типа данных зависит от того, является ли это бит состояния или текущее значение. Для любой команды сравнения, использующей CT2, будет доступно текущее значение, для всех других команд, использующих CT2, доступен бит состояния. Текущие значения доступны также через ячейку V-памяти. В DirectSOFT использование CT2 относится к биту дискретного состояния счетчика. Вы должны использовать V1002 (или альтернативное имя CTA2), чтобы обратиться к текущему значению.

Пример счетчика, использующего биты дискретного состояния

В следующем примере, когда X1 переходит из положения ВЫКЛ. в положение ВКЛ., счетчик CT2 увеличится на 1. Когда текущее значение достигнет заданного значения 3, бит состояния счетчика CT2 включится и включит Y10. Когда вход сброса C10 включится, бит состояния счетчика выключится и текущее значение станет 0. Текущее значение для счетчика CT2 будет храниться в ячейке V-памяти V1002.



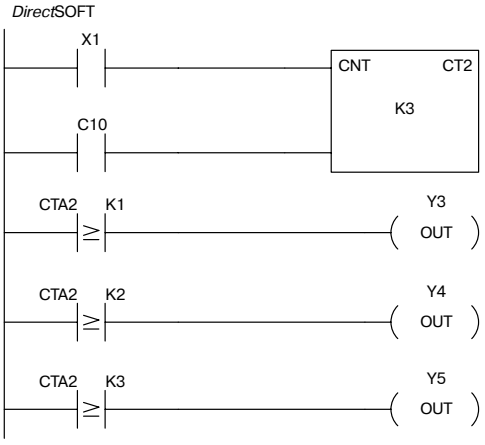


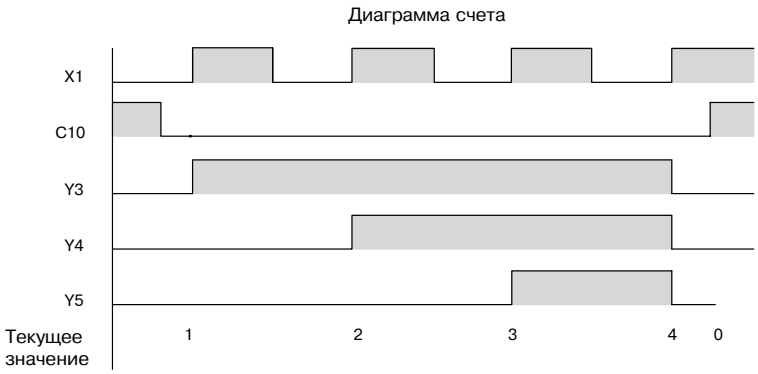
Набор на ручном программаторе

STR	X(IN)	1	↵		
STR	C(CR)	1	0	↵	
CNT	CNT	2	K(CON)	3	↵
STR	CNT	2	↵		
OUT	Y(OUT)	1	0	↵	

Пример счетчика, использующего контакты сравнения

В следующем примере, когда X1 переходит из положения ВЫКЛ. в положение ВКЛ., счетчик CT2 увеличится на 1. Контакты сравнения используются, чтобы включить Y3, Y4 и Y5 для разных подсчетов. Контакты сравнения выключаются, когда счетчик сбрасывается. Когда сброс C10 включится, бит состояния счетчика выключится, текущее значение счетчика станет 0.





Набор на ручном программаторе

STR	X(IN)	1	↵		
STR	C(CR)	1	0	↵	
CNT	CNT	2	K(CON)	3	↵
STR	CNT	2	K(CON)	1	↵
OUT	Y(OUT)	3	↵		

Набор на ручном программаторе

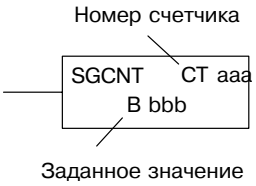
STR	CNT	2	K(CON)	2	↵
OUT	Y(OUT)	4	↵		
STR	CNT	2	K(CON)	3	↵
OUT	Y(OUT)	5	↵		

ПРИМЕЧАНИЕ. Когда этот пример показывается из DirectSOFT, Вы должны использовать альтернативное имя CTA2 (или V1002) вместо CT2, которое требуется для эквивалентной цепи при вводе с ручного программатора.

Stage Counter (SGCNT)

√	√	√
430	440	450

Stage Counter - это счетчик с одним входом, который увеличивается при логическом переходе входа из состояния ВЫКЛ. в состояние ВКЛ. Этот счетчик отличается от других счетчиков, так как он хранит текущее значение до тех пор, пока оно не будет сброшено командой RST. Stage Counter разработан для использования в программах RLL^{PLUS}, но может использоваться в программах релейной логики. Когда текущее значение равно заданному значению, бит состояния счетчика включается, а счетчик продолжает считать до максимального значения 9999. Максимальное значение будет сохраняться до тех пор, пока счетчик не будет сброшен.



Бит дискретного состояния счетчика и текущее значение не определяются в команде счетчика.

Спецификации команды

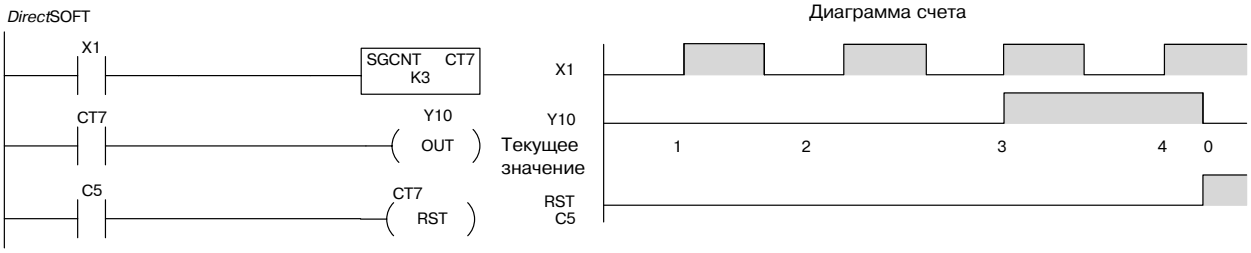
Метка счетчика (CTaaa): Указывает номер счетчика.
Заданное значение (Bbbb): Константа (K) или ячейка V-памяти. (V-ячейки являются 16-битовыми словами).
Текущие значения: Текущие значения счетчика доступны при обращении к соответствующим ячейкам V- или CT-памяти*. Ячейка V-памяти - ячейка счетчика + 1000. Например, текущее значение счетчика для CT3 находится в ячейке V-памяти V1003. (V-ячейки являются 16-битовыми словами).
Бит дискретного состояния: Бит состояния доступен при обращении к соответствующей ячейке CT-памяти. Он будет включен, если текущее значение равно или больше заданного значения. Например, бит состояния для счетчика 2 был бы CT2.

Тип данных операнда		Диапазон DL430		Диапазон DL440		Диапазон DL450	
	B	**aaa**	**bbb**	**aaa**	**bbb**	**aaa**	**bbb**
Таймеры	T	0-177	––	0-377	––	0-377	––
V-память для заданных значений	V	––	1400-7377	––	1400–7377 10000–17777	––	1400–7377 10000–17777
Пойнтеры-указатели (только заданные)	P	––	––	––	1400–7377 10000–17777	––	1400–7377 10000–17777
Константы (только заданные)	K	––	0–9999	––	0–9999	––	0–9999
Дискретные биты состояния таймера	T/V	0-177		0-377		0–377	
Текущие значения таймера	V /T*	1000-1177		1000-1177		1000-1377	

ПРИМЕЧАНИЕ. * На ручном программаторе доступны как биты дискретных состояний счетчика стадий, так и текущие значения с одним типом данных (пример CT2). Способ использования типа данных зависит от того, является ли это бит состояния или текущее значение. Для любой команды сравнения, использующей CT2, будет доступно текущее значение, для всех других команд, использующих CT2, доступен бит состояния. Текущие значения доступны также через ячейку V-памяти. В DirectSOFT использование CT2 относится к биту дискретного состояния счетчика. Вы должны использовать V1002 (или альтернативное имя CTA2), чтобы обратиться к текущему значению.

Пример счетчика стадий (Stage), использующего биты дискретного состояния

В следующем примере, когда X1 переходит из положения ВЫКЛ. в положение ВКЛ., счетчик CT7 увеличится на 1. Когда текущее значение достигнет 3, бит состояния счетчика CT7 включится и включит Y10. Бит состояния счетчика CT7 будет оставаться включенным, пока счетчик не будет сброшен командой RST. Когда счетчик сброшен, бит состояния счетчика выключится, а текущее значение счетчика станет 0. Текущее значение для счетчика CT7 будет храниться в ячейке V-памяти V1007.

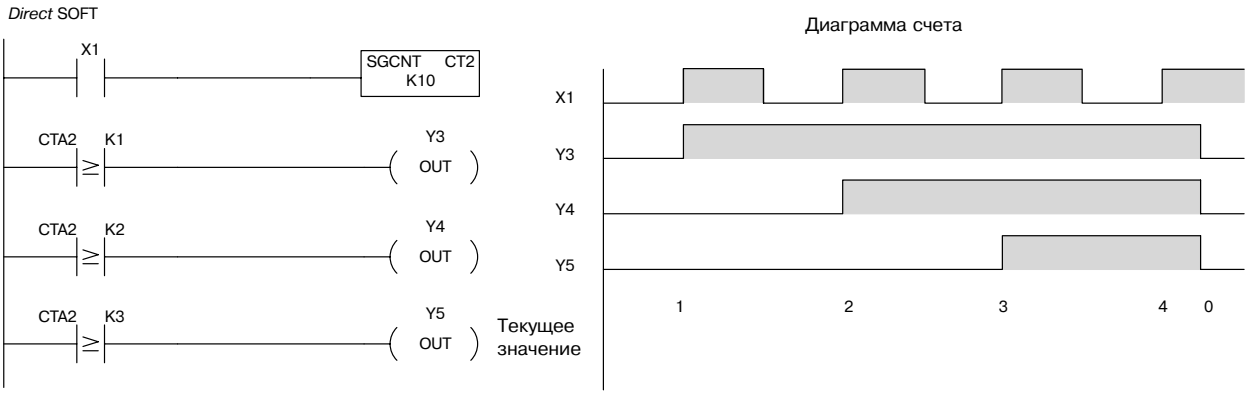


Набор на ручном программаторе

STR	X(IN)	1	↵			
SG	CNT	CNT	7	K(CON)	3	↵
STR	CNT	7	↵			
OUT	Y(OUT)	1	0	↵		
STR	C(CR)	5	↵			
RST	CNT	7	↵			

Пример счетчика стадий (Stage), использующего контакты сравнения

В следующем примере когда X1 переходит из положения ВЫКЛ. в положение ВКЛ., счетчик CT2 увеличится на 1. Контакты сравнения используются, чтобы включить Y3, Y4 и Y5 для разных подсчетов. Хотя это и не показано в примере, когда счетчик сбрасывается командой Reset, бит состояния счетчика выключится и текущее значение станет 0. Текущее значение для счетчика CT2 хранится в ячейке V-памяти V1002.



Набор на ручном программаторе

STR	X(IN)	1	↵				
SG	CNT	CNT	2	K(CON)	1	0	↵
STR	CNT	2	K(CON)	1	↵		
OUT	Y(OUT)	3	↵				

Набор на ручном программаторе

STR	CNT	2	K(CON)	2	↵
OUT	Y(OUT)	4	↵		
STR	CNT	2	K(CON)	3	↵
OUT	Y(OUT)	5	↵		

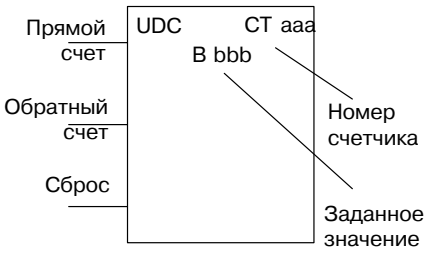
ПРИМЕЧАНИЕ. Когда этот пример показывается из DirectSOFT, Вы должны использовать альтернативное имя CTA2 (или V1002) вместо CT2, которое требуется для эквивалентной цепи при вводе с ручного программатора.

Up Down Counter (UDC)

√	√	√
430	440	450

Счетчик Up/Down Counter (Реверсивный счетчик) производит прямой отсчет при

каждом переходе входа Up из состояния ВЫКЛ. в состояние ВКЛ. И обратный отсчет при каждом переходе входа Down из состояния ВЫКЛ. в состояние ВКЛ. Счетчик сбрасывается в 0, когда вход сброса включается. Диапазон счета: 0-99999999. Чтобы функционировал активный вход счета, неиспользуемый вход счета должен быть выключен.



Бит дискретного состояния счетчика и текущее значение не определяются в команде счетчика.

Предостережение: UDC использует две последовательных ячейки, так как заданное значение может состоять из 8 цифр, которые требуют две ячейки V-памяти. Например, если в программе используется UDC CT0, то следующим доступным счетчиком будет CT2. Или, если CT0 был обычный счетчик, и CT1 был реверсивным счетчиком, то следующим доступным счетчиком будет CT3.

Спецификации команды

Метка счетчика (CTaaa): Указывает номер счетчика.
Заданное значение (Bbbb): Константа (K) или две последовательные ячейки V-памяти. (V-ячейки являются 16-битовыми словами).
Текущие значения: Текущие значение счетчика является двойным словом, доступным при обращении к соответствующим ячейкам V- или CT-памяти*. Ячейка V-памяти - это ячейка счетчика + 1000. Например, текущее значение счетчика для CT5 находится в ячейках V-памяти V1005 и V1006. (V-ячейки являются 16-битовыми словами).
Бит дискретного состояния: Бит состояния доступен при обращении к соответствующей ячейке CT-памяти. Он будет включен, если текущее значение равно или больше заданного значения. Например, бит состояния для счетчика 2 был бы CT2.

Тип данных операнда		Диапазон DL430		Диапазон DL440		Диапазон DL450	
	B	**aaa**	**bbb**	**aaa**	**bbb**	**aaa**	**bbb**
Таймеры	T	0-177	--	0-377	--	0-377	--
V-память для заданных значений	V	--	1400-7377	--	1400–7377 10000–17777	--	1400–7377 10000–17777
Пойнтеры-указатели (только заданные)	P	--	--	--	1400–7377 10000–17777	--	1400–7377 10000–17777
Константы (только заданные)	K	--	0-99999999	--	0-99999999	--	0-99999999
Дискретные биты состояния таймера	T/V	0-177		0-377		0–377	
Текущие значения таймера	V /T*	1000-1177		1000-1177		1000-1177	

ПРИМЕЧАНИЕ. * На ручном программаторе доступны как биты дискретных состояний счетчика стадий, так и текущие значения с одним типом данных (пример CT2). Способ использования типа данных зависит от того, является ли это бит состояния или текущее значение. Для любой команды сравнения, использующей CT2, будет доступно текущее значение, для всех других команд, использующих CT2, доступен бит состояния. Текущие значения доступны также через ячейку V-памяти. В DirectSOFT использование CT2 относится к биту дискретного состояния счетчика. Вы должны использовать V1002 (или альтернативное имя CTA2), чтобы обратиться к текущему значению.

Пример реверсивного (Up/Down) счетчика, использующего дискретные биты состояния

В следующем примере, если X2 и X3 выключены, когда X1 переключается из состояния ВЫКЛ. в состояние ВКЛ., счетчик увеличится на 1. Если X1 и X3 выключены, счетчик уменьшится на 1, когда X2 переключается из состояния ВЫКЛ. в состояние ВКЛ. Когда текущее значение достигнет заданного значения 3, бит состояния счетчика включится. Когда сброс X3 включится, бит состояния счетчика выключится, а текущее значение счетчика станет 0.

DirectSOFT

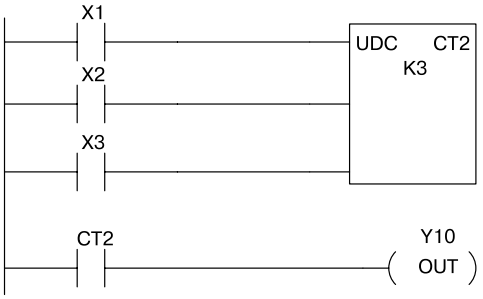
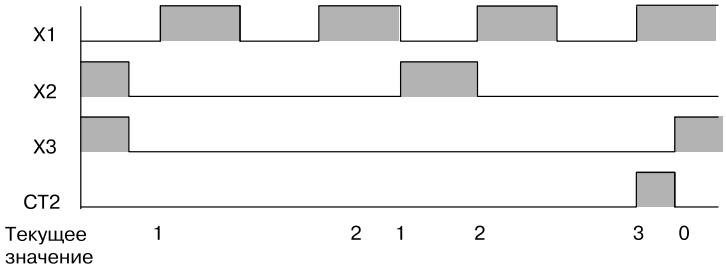


Диаграмма работы



Набор на ручном программаторе

STR	X(IN)	1	↵			
STR	X(IN)	2	↵			
STR	X(IN)	3	↵			
SHFT	U	D	C	SHFT	CNT	2

Набор на ручном программаторе

K(CON)	3	↵		
STR	CNT	2	↵	
OUT	Y(OUT)	1	0	↵

Пример реверсивного (Up/Down) счетчика, использующего контакты сравнения

В следующем примере, когда X1 переходит из состояния ВЫКЛ. в состояние ВКЛ., счетчик CT2 увеличится на 1. Контакты сравнения используются, чтобы включить Y3 и Y4 с разными отсчетами. Контакты сравнения выключаются, когда счетчик сбрасывается. Когда сброс X3 включится, бит состояния счетчика выключится, текущее значение станет 0.

DirectSOFT

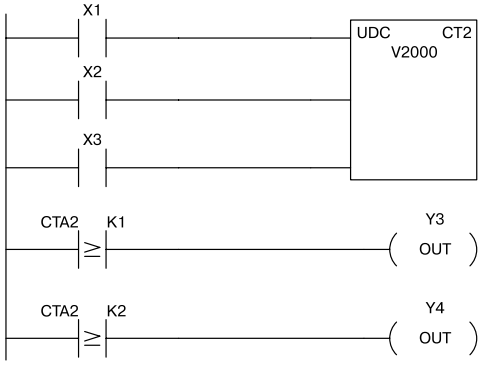
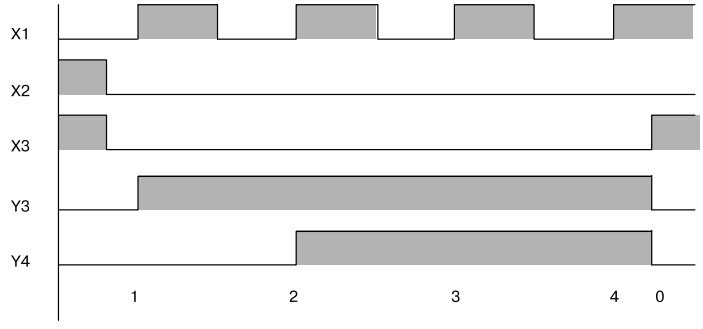


Диаграмма работы



Набор на ручном программаторе

STR	X(IN)	1	↵			
STR	X(IN)	2	↵			
STR	X(IN)	3	↵			
SHFT	U	D	C	SHFT	CNT	2
V	1	4	0	0	↵	
STR	CNT	2	K(CON)	1	↵	

Набор на ручном программаторе

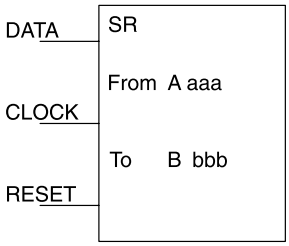
OUT	Y(OUT)	3	↵		
STR	CNT	2	K(CON)	2	↵
OUT	Y(OUT)	4	↵		

ПРИМЕЧАНИЕ. Когда этот пример показывается из DirectSOFT, Вы должны использовать альтернативное имя CTA2 (или V1002) вместо CT2, которое требуется для эквивалентной цепи при вводе с ручного программатора.

Shift Register (SR)

√	√	√
430	440	450

Команда Shift Register (Регистр сдвига) сдвигает данные на заранее определенное число управляющих реле. Диапазон управления в блоке регистров сдвига должен начинаться в начале 8-битовой границы и заканчиваться в конце 8-битовой границы реле управления.

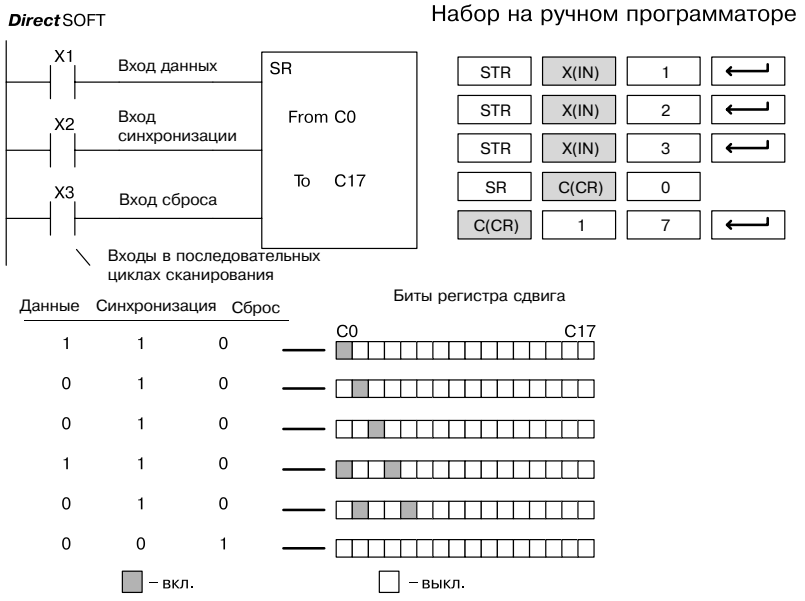


Сдвиговый регистр имеет три контакта.

- Данные (Data) - определяет значение (1 или 0), которое будет введено в регистр,
- Синхронизация (Clock) - сдвигает биты на один разряд при каждом изменении от низкого уровня к высокому.
- Сброс (Reset) - устанавливает регистр сдвига в нулевое состояние.

При каждом переходе входа синхронизации (Clock) из состояния ВЫКЛ. в состояние ВКЛ., биты, составляющие блок регистра сдвига, сдвигаются на один битовый разряд и состояние входа данных помещается в начальную позицию регистра сдвига. Направление сдвига зависит от ввода ОТ и ДО полей. Ввод от C0 до C17 определяет блок из шестнадцати битов, которые будут сдвинуты с меньшего адреса в более высокий адрес. Ввод от C17 до C0 определяет блок из шестнадцати битов, которые будут сдвинуты с более высокого на более низкий адрес. Максимальный размер блока регистра сдвига зависит от количества доступных управляющих реле. Минимальный размер блока - 8 управляющих реле.

Тип данных операнда		Диапазон DL430		Диапазон DL440		Диапазон DL450	
	B	**aaa**	**bbb**	**aaa**	**bbb**	**aaa**	**bbb**
Реле управления	C	0-737	0-737	0–1777	0–1777	0-3777	0-3777



Команды загрузки и вывода аккумулятора/стека данных

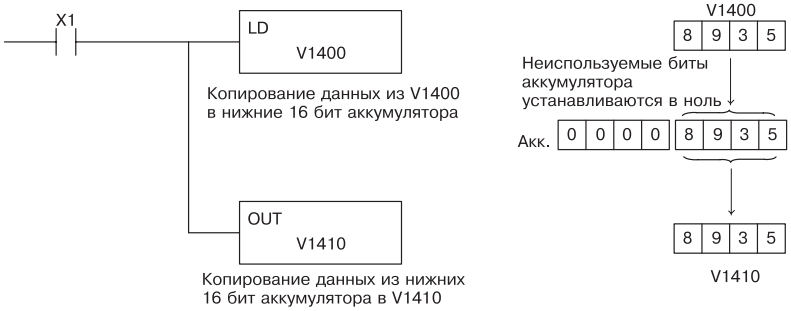
Использование аккумулятора

Аккумулятор в процессорах серии DL405 - это 32-битовый регистр, используемый как временная ячейка хранения данных, которые копируются и управляются несколькими способами. Например, Вы должны использовать аккумулятор для выполнения математических операций, таких как сложение, вычитание, умножение и т.п. Так как имеется 32 бита, Вы можете использовать 8-значные двоично-десятичные числа или 32-битовые числа в дополнительном коде. Аккумулятор сбрасывается в 0 в конце каждого цикла сканирования процессора.

Копирование данных в аккумулятор

Команды Load и Out и их вариации используются для копирования данных из ячейки V-памяти в аккумулятор, или для копирования данных из аккумулятора в V-память.

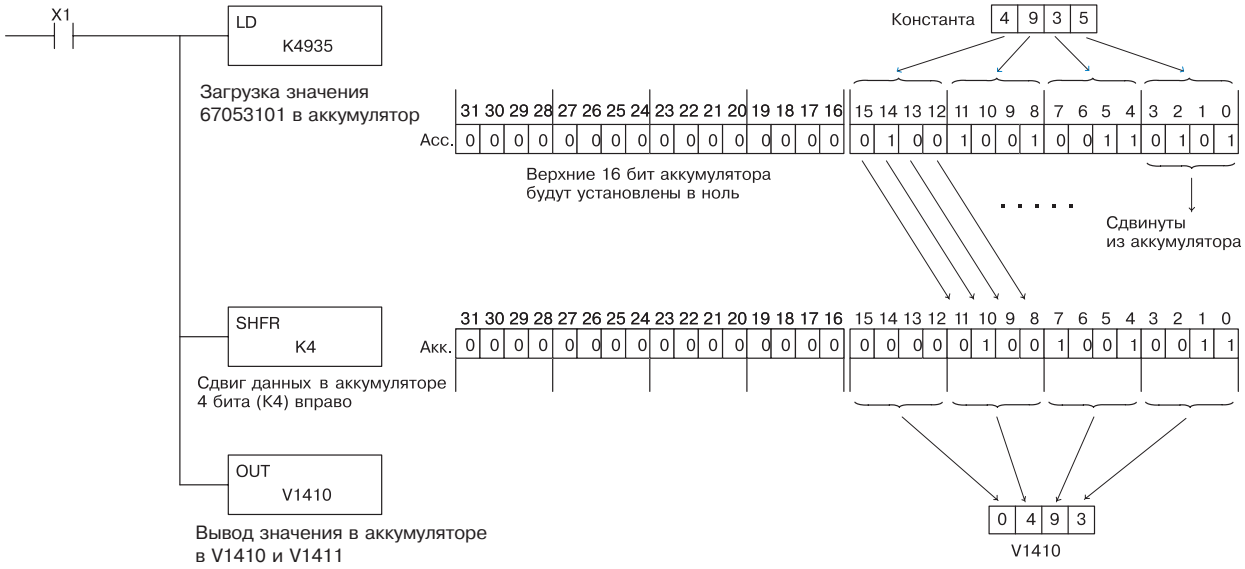
Пример копирования данные из ячейки V-памяти V1400 в ячейку V-памяти V1410.



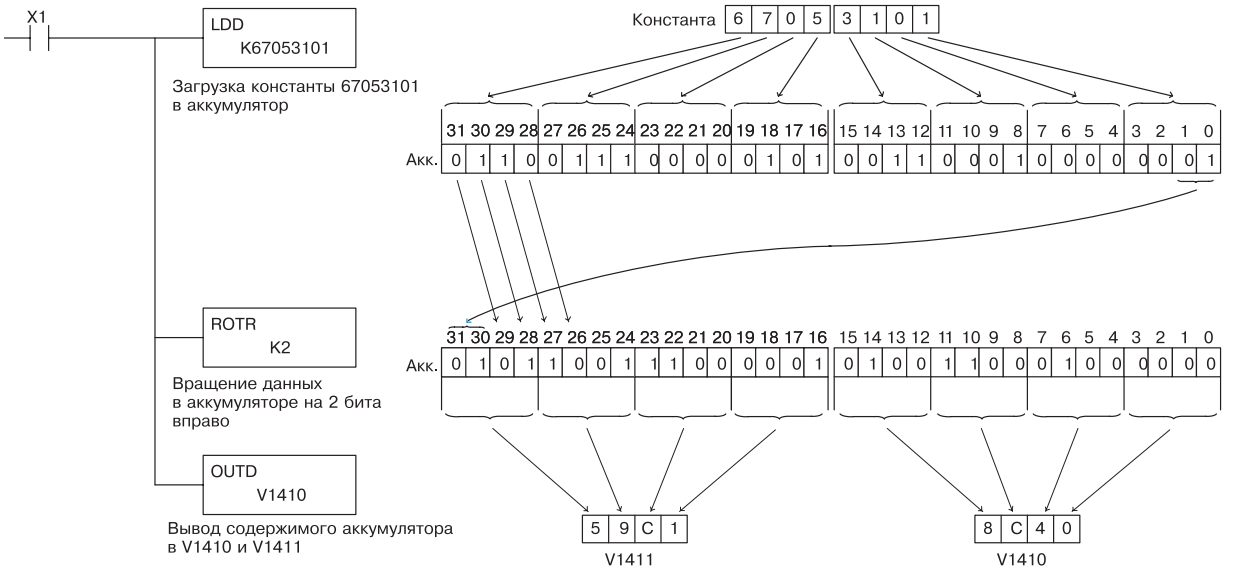
Так как аккумулятор 32-битовый, а ячейки V-памяти 16-битовые, то команды Load Double и Out Double (или их вариации) используют две последовательные ячейки V-памяти или 8-значные двоично-десятичные константы для копирования данных из аккумулятора в адрес V-памяти или из адреса V-памяти в аккумулятор. Например, если Вы хотите копировать данные из ячеек V-памяти V1400 и V1401 в ячейки V-памяти V1410 и V1411, то самый эффективный путь выполнить эту операцию был бы следующим:

Изменение данных аккумулятора

Команды, которые оперируют с данными, также используют аккумулятор. Результат операций с данными находится в аккумуляторе. Данные, которые были обработаны, стираются из аккумулятора. Следующий пример: загрузка двоично-десятичной константы 4935 в аккумулятор, сдвиг данных вправо на 4 бита и вывод результата в V1410.

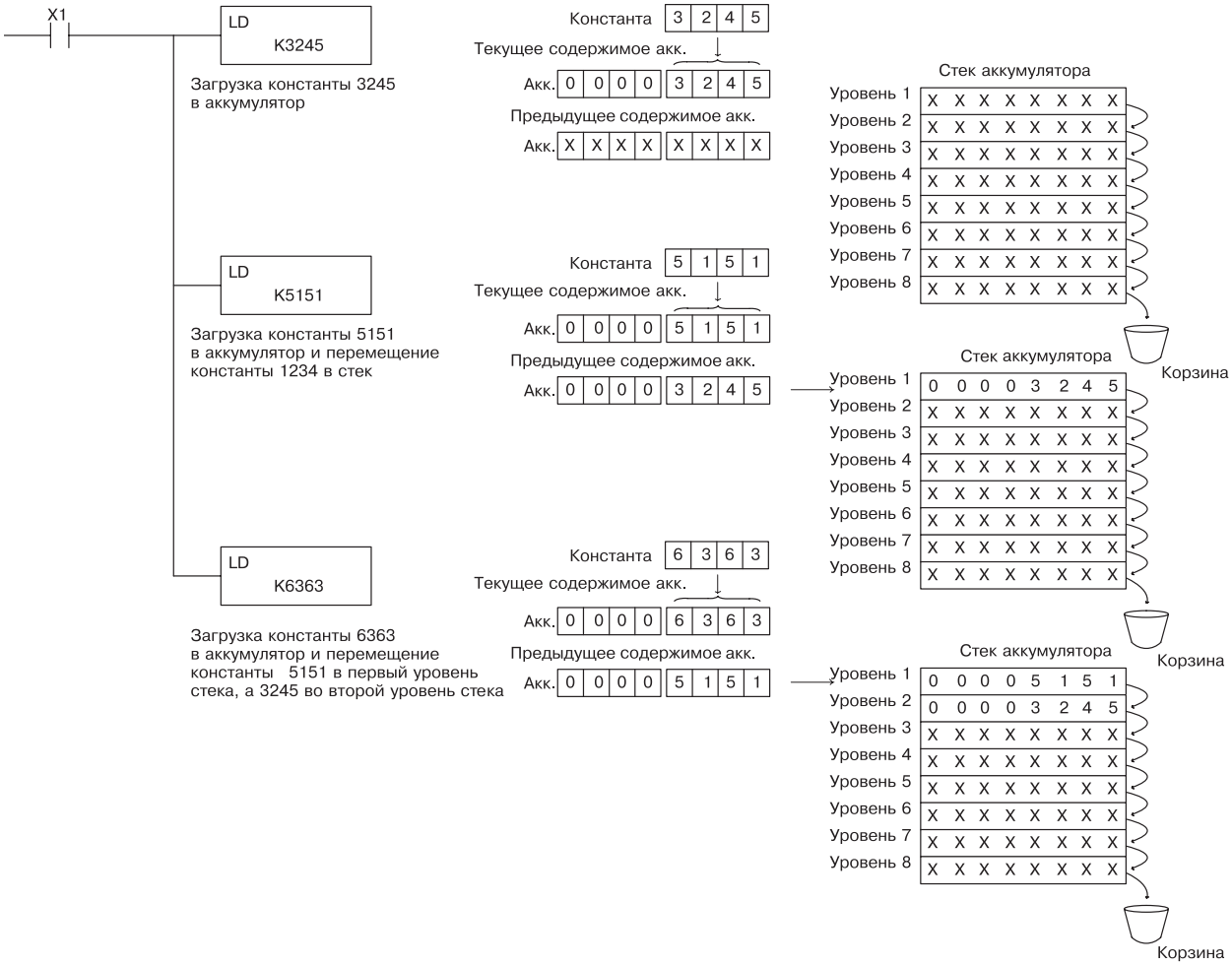


Некоторые из команд, которые оперируют с данными, используют 32 бита. Они используют две последовательные ячейки V-памяти или 8-значные двоично-десятичные константы для работы с данными в аккумуляторе. В следующем примере сдвигается значение 67053101 на два бита вправо и выводит значение в V1410 и в V1411.

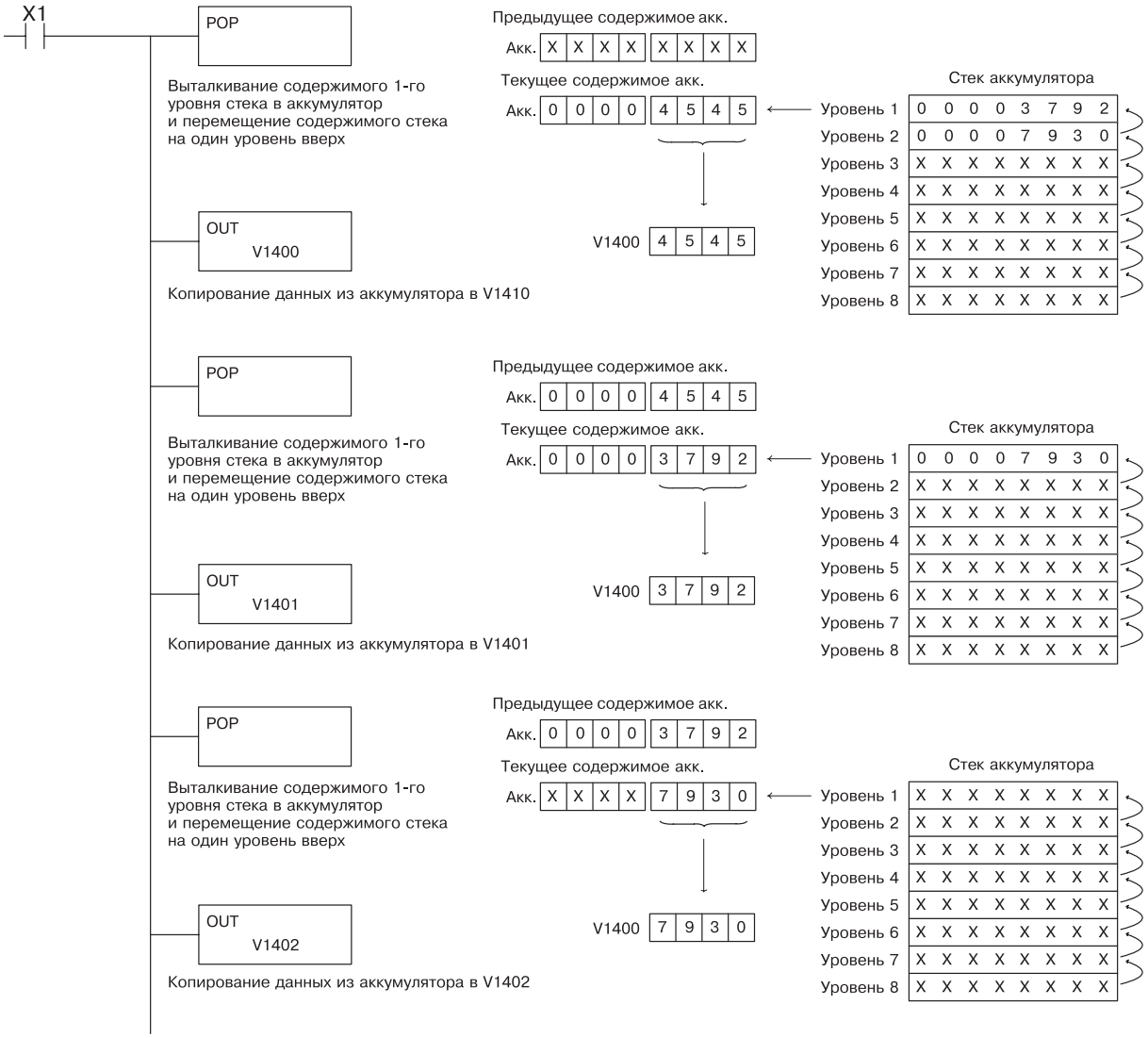


Использование стека аккумулятора

Стек аккумулятора используется для команд, которые требуют больше одного параметра для выполнения функции или для выполнения набора функций, определяемого пользователем. Стек аккумулятора используется тогда, когда более одной команды типа Load выполняется без использования команды типа Out. Первая команда Load в цикле сканирования помещает значение в аккумулятор. После этого любая команда Load без использования команды Out помещает значение в аккумулятор, а значение, которое было в аккумуляторе, помещается в стек аккумулятора. Команда Out аннулирует предыдущую команду Load и не помещает значение из аккумулятора в стек аккумулятора, когда следующая команда Load выполняется. Каждый раз, когда значение помещается в стек аккумулятора, другие значения в стеке опускаются на один уровень. Аккумулятор имеет 8 уровней (восемь 32-битовых регистров). Если на восьмом уровне находится какое-либо значение, то при помещении нового значения в стек, значение на восьмом уровне исчезает из стека и не может быть восстановлено.



Команда POP сдвигает значения вверх через стек в аккумулятор. Когда выполняется команда POP, значение, которое было в аккумуляторе, стирается, а значение, которое было на вершине стека, попадает в аккумулятор. Значения в стеке сдвигаются вверх на один уровень.



Ячейки памяти аккумулятора и стека аккумулятора

X	√	√
430	440	450

Иногда вам необходимо прочитать значение, которое помещено в стек аккумулятора без выталкивания первого уровня стека. Как аккумулятор, так и стек аккумулятора имеют соответствующие ячейки V-памяти, которые доступны программам.

Вы не можете записывать в эти ячейки, но Вы можете читать их и использовать их в булевых командах сравнения и т. д.

Аккумулятор

0 0 0 0	3 7 9 2
V701	V700

Стек аккумулятора

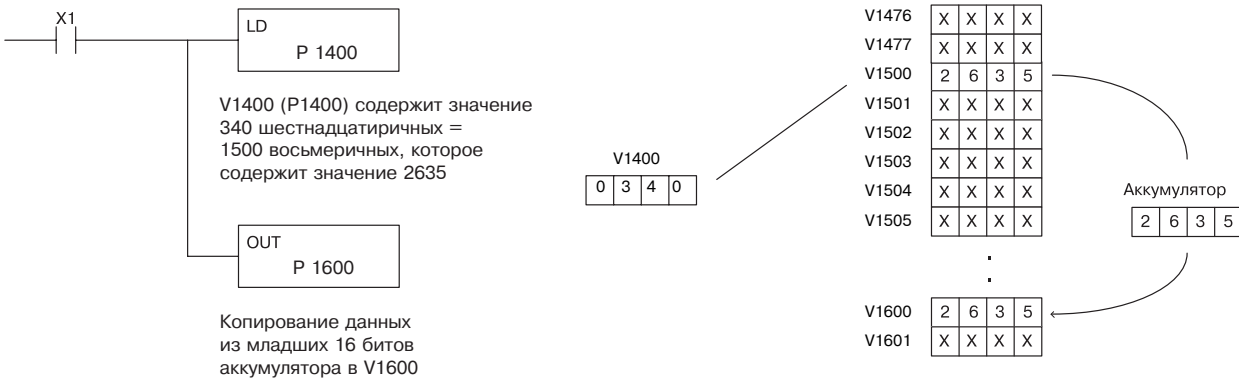
Уровень 1	0 0 0 0 3 7 9 2	V703 - V702
Уровень 2	0 0 0 0 7 9 3 0	V705 - V704
Уровень 3	0 0 0 0 0 0 0 0	V707 - V706
Уровень 4	0 0 0 0 0 0 0 0	V711 - V710
Уровень 5	0 0 0 0 0 0 0 0	V713 - V712
Уровень 6	0 0 0 0 0 0 0 0	V715 - V714
Уровень 7	0 0 0 0 0 0 0 0	V717 - V716
Уровень 8	0 0 0 0 0 0 0 0	V721 - V720

Использование указателей (Pointers)

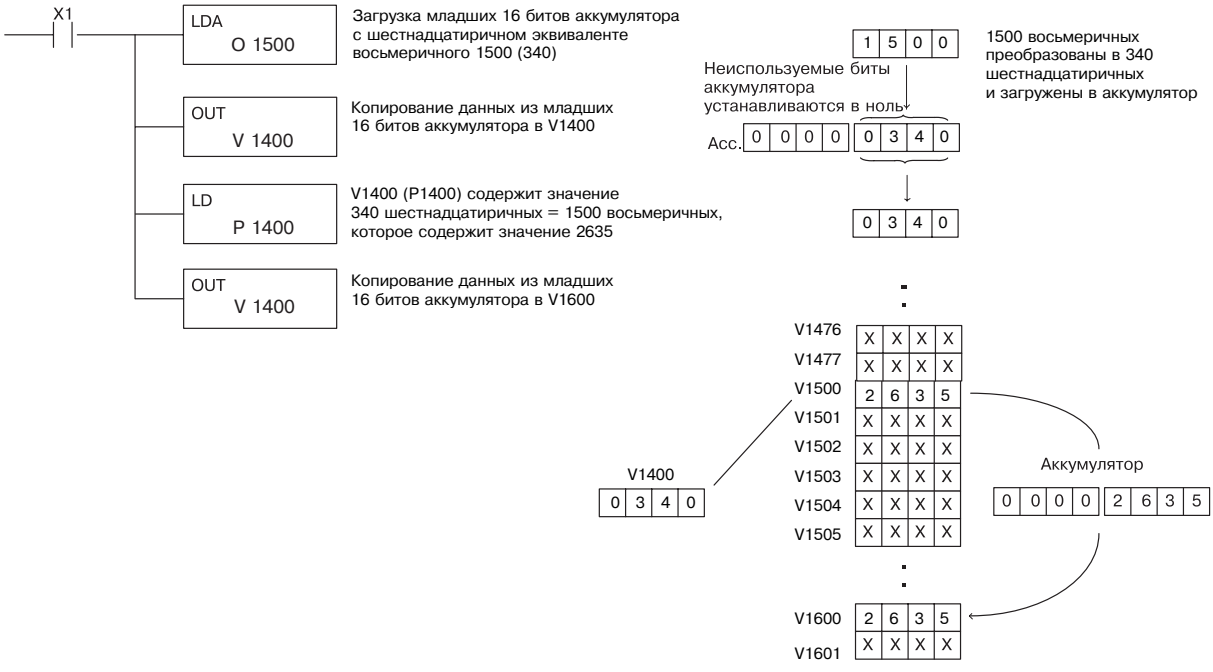
Многие из команд процессоров серии DL405 допускают использование указателей V-памяти в качестве операндов. Указатели могут быть полезны в программировании на языке релейной логики, но трудно использовать их в Вашей задаче, если Вы не имели опыта работы с указателями (известными как непрямая адресация). Указатели позволяют командам получать данные из ячеек V-памяти, относящихся к значению указателя.

ПРИМЕЧАНИЕ. В процессорах серии DL405 используется восьмеричная адресация V-памяти. Однако значение в ячейке указателя, которое будет относиться к ячейке V-памяти, показывается в шестнадцатиричном формате. Используйте команду Load Address для передачи адреса в ячейку указателя. Эта команда выполняет преобразование из восьмеричного формата в шестнадцатиричный.

Следующий пример использует операнд указателя в команде Load. Ячейка V-памяти V1400 является ячейкой указателя. V1400 содержит значение 340, которое является шестнадцатиричным эквивалентом восьмеричного адреса ячейки V-памяти V1500. Процессор копирует данные из ячейки V1500 (содержащей 2635) в младшее слово аккумулятора.



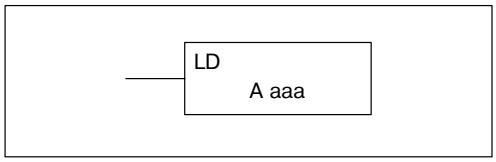
Следующий пример похож на пример, приведенный выше, если бы не команда LDA (загрузка адреса - load address), которая автоматически преобразует восьмеричный адрес в шестнадцатиричный эквивалент.



Load (LD)

√	√	√
430	440	450

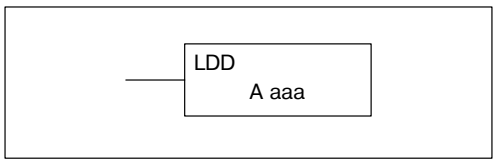
Команда Load - это 16-битовая команда, которая загружает значение (Аааа) (либо ячейку V-памяти, либо 4-знаковую константу) в младшие 16 бит аккумулятора. Старшие 16 бит аккумулятора устанавливаются в 0.



Load Double (LDD)

√	√	√
430	440	450

Команда Load Double - это 32-битовая команда, которая загружает значение (Аааа), являющееся либо двумя последовательными ячейками V-памяти, либо 8-знаковой константой, в аккумулятора.



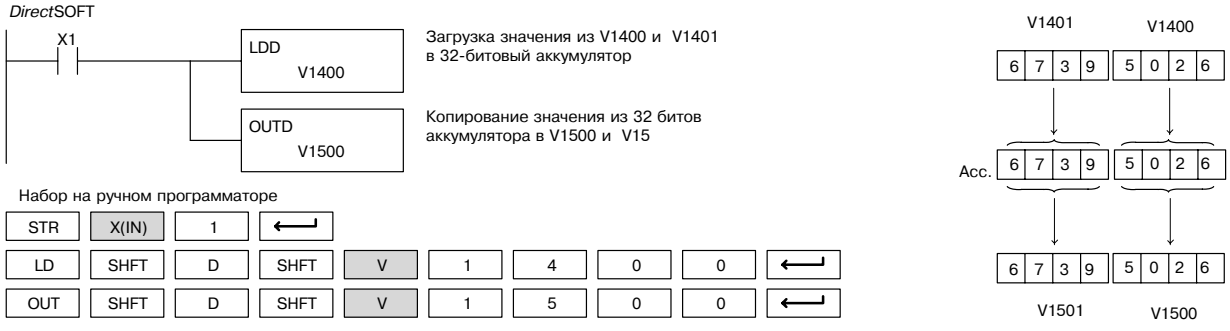
Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	aaa	aaa	aaa
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Вся V-память (см. стр. 3-40)	Вся V-память (см. стр. 3-41)	Вся V-память (см. стр. 3-42)
Константа	K	0-FFFF	0-FFFF	0-FFFF

Флаги дискретных разрядов	Описание
SP76	Включено, когда значение, загружаемое в аккумулятор любой командой, является нулем.

ПРИМЕЧАНИЕ. Две последовательных команды Load или Load Double поместят значение первой команды Load в стек аккумулятора.

В следующем примере, когда X1 включен, значение в V1400 будет загружено в аккумулятор и выведено в V1500.

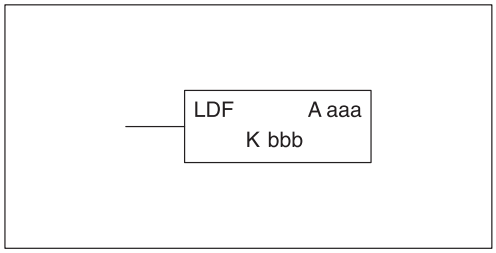
В следующем примере, когда X1 включен, 32-битовое значение в V1400 и V1401 будет загружено в аккумулятор и выведено в V1500 и V1501.



Load Formatted (LDF)

X	X	√
430	440	450

Команда Load Formatted загружает 1-32 последовательных бита из дискретных ячеек памяти в аккумулятор. Для загрузки команды требуется определить стартовую ячейку (Aaaa) и количество битов (Kbbb). Неиспользуемые ячейки битов аккумулятора устанавливаются в нуль.

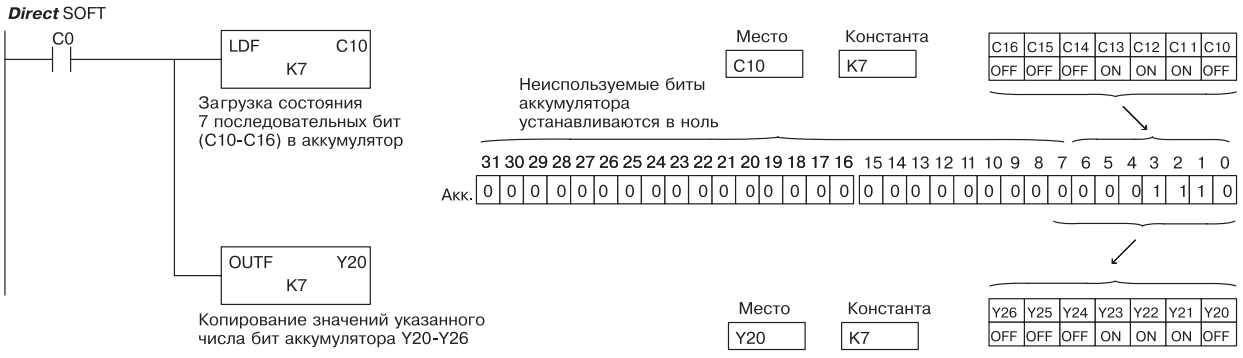


Тип данных операнда		Диапазон DL440		Диапазон DL450	
	A	**aaa**	**bbb**	**aaa**	**bbb**
Входы	X	0-477	--	0-1777	--
Выходы	Y	0-477	--	0-1777	--
Реле управления	C	0-1777	--	0-3777	--
Биты стадии	S	0-1777	--	0-1777	--
Биты таймера	T	0-377	--	0-377	--
Биты счетчика	CT	0-177	--	0-377	--
Специальные реле	SP	0-137 320-717	--	0-137 320-717	--
Глобальный ввод/вывод	GX	0-1777	--	0-2777	--
Константа	K	--	1-32	--	1-32

Флаги дискретных разрядов	Описание
SP76	Включено, когда значение, загружаемое в аккумулятор любой командой, □ является нулем.

ПРИМЕЧАНИЕ. Две последовательных команды Load поместят значение первой команды Load в стек аккумулятора.

В следующем примере, когда C0 включен, двоичный набор C10-C16 (7 бит) будет загружен в аккумулятор с использованием команды Load Formatted. Младшие 6 бит аккумулятора выводятся на Y20-Y26, используя команду Out Formatted.



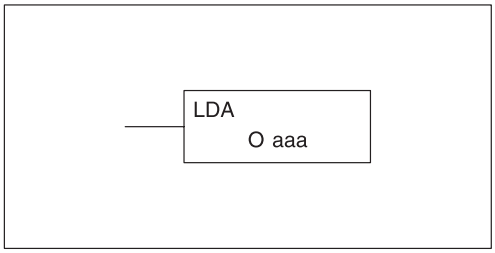
Набор на ручном программаторе

STR	C(CR)	0	↵						
LD	SHFT	F	SHFT	C(CR)	1	0	K(CON)	7	↵
OUT	SHFT	F	SHFT	Y(OUT)	2	0	K(CON)	7	↵

Load Address (LDA)

√	√	√
430	440	450

Команда Load Address является 16-битовой командой. Она преобразует любое восьмеричное значение или адрес в шестнадцатеричный эквивалент и загружает шестнадцатиричное значение в аккумулятор. Эта команда полезна, когда требуется в качестве параметра адрес, так как все адреса для системы DL405 находятся в восьмеричном виде.

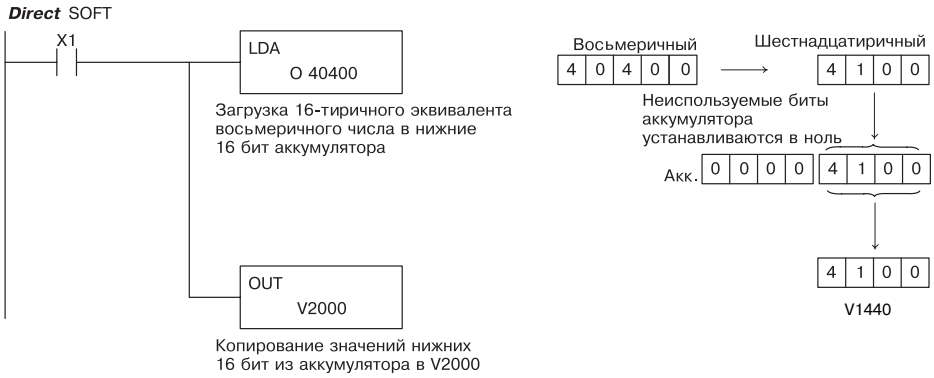


Тип данных операнда	Диапазон DL430	Диапазон DL440	Диапазон DL450
	aaa	**aaa**	**aaa**
Восьмеричный O адрес	0 - 77777	0 - 77777	0 - 177777

Флаги дискретных разрядов	Описание
SP76	Включено, когда значение, загружаемое в аккумулятор любой командой, □ является нулем.

ПРИМЕЧАНИЕ. Две последовательных команды Load поместят значение первой команды Load в стек аккумулятора.

В следующем примере, когда X1 включен, восьмеричное значение 40400 будет преобразовано в шестнадцатиричное значение 4100 и загружено в аккумулятор, используя команду Load Address. Значение младших 16 битов аккумулятора будет скопировано в V1440, используя команду Out.



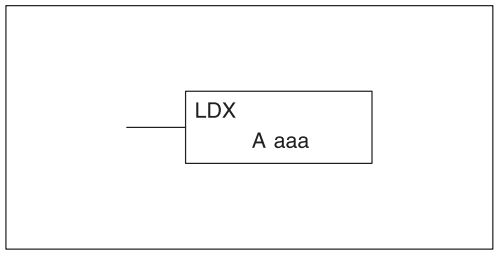
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	A	OCT	4	0	4	0	0	↵
OUT	V	1	4	4	0	↵			

Load Accumulator Indexed (LDX)

√	√	√
430	440	450

Команда Load Accumulator Indexed - это 16-битовая команда, которая указывает исходный адрес (V-памяти), который будет смещен на значение из первой ячейки стека. Эта команда интерпретирует значение в первой ячейке стека как шестнадцатиричное значение. Значение в адресе смещения (исходный адрес + смещение) загружается в младшие 16 битов аккумулятора. Старшие 16 битов аккумулятора устанавливаются в 0.



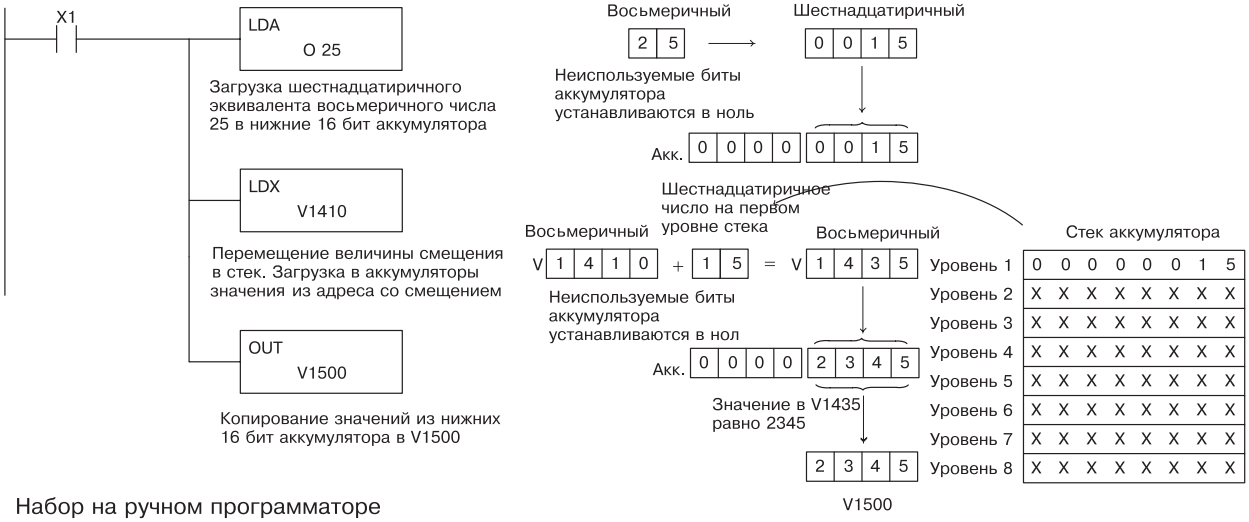
Полезная подсказка: Команда Load Address может использоваться для преобразования восьмеричного адреса в шестнадцатиричный адрес и для загрузки значения в аккумулятор.

Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	aaa	aaa	aaa
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	—	Все (см. стр. 3-41)	Все (см. стр. 3-42)

Флаги дискретных разрядов	Описание
SP76	Включено, когда значение, загружаемое в аккумулятор любой командой, является нулем.

ПРИМЕЧАНИЕ. Две последовательных команды Load поместят значение первой команды Load в стек аккумулятора.

В следующем примере, когда X1 включен, шестнадцатиричный эквивалент восьмеричного значения 25 будет загружен в аккумулятор (это значение помещается в стек когда команда Load Accumulator Indexed выполняется). Ячейка V-памяти V1410 будет добавлена к значению на первом уровне стека, и значение в этой ячейке (V1435 = 2345) будет загружено в младшие 16 бит аккумулятора, используя команду Load Accumulator Indexed. Значение из младших 16 битов аккумулятора записывается в V1500, используя команду Out.

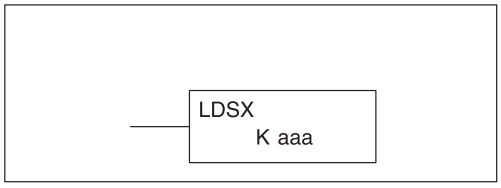


Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	A	OCT	2	5	↵			
LD	SHFT	X	SHFT	V	1	4	1	0	↵
OUT	V	1	5	0	0	↵			

Load Accumulator Indexed from Indexed from Data Constants (LDSX)

X	√	√
430	440	450



Команда Load Accumulator Indexed from Data Constants - это 16-битовая команда. Эта команда определяет область меток данных Data Label Area (DLBL), где хранятся числа или ASCII константы. Это значение будет загружено в младшие 16 битов.

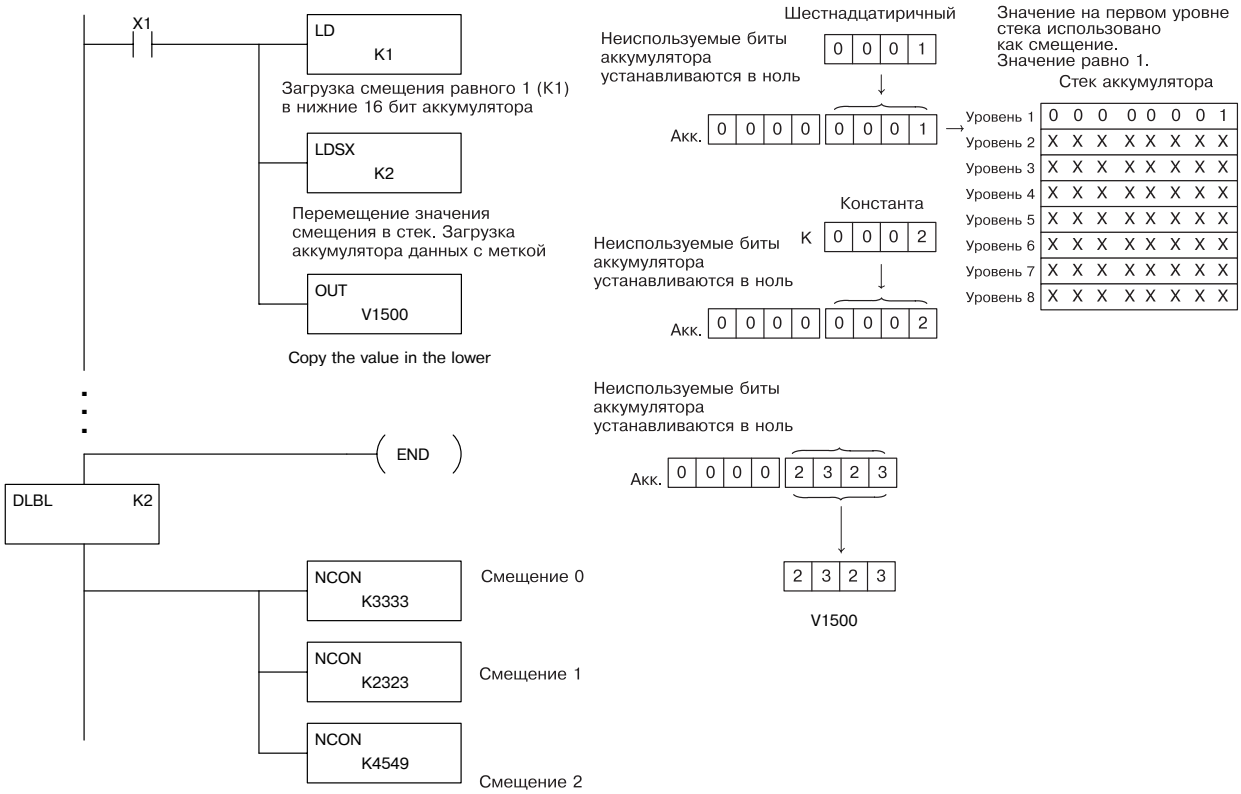
Команда LDSX использует значение в первом уровне стека аккумулятора в качестве смещения, чтобы определить какое число или константа ASCII из области меток данных (DLBL) будет загружено в аккумулятор. Команда LDSX интерпретирует значение в первом уровне стека аккумулятора как шестнадцатеричное значение.

Тип данных операнда		Диапазон DL440	Диапазон DL450
		aaa	aaa
Константа	K	1-FFFF	1-FFFF

Полезная подсказка: Команда Load Address может использоваться для преобразования восьмеричного адреса в шестнадцатеричный адрес и для загрузки значения в аккумулятор.

ПРИМЕЧАНИЕ. Две последовательных команды Load поместят значение первой команды Load в стек аккумулятора.

В следующем примере, когда X1 включен, команда Load загружает смещение 1 в аккумулятор. Это значение будет помещено в первый уровень стека аккумулятора после выполнения команды LDSX. Команда LDSX определяет метку данных (DLBL K2), где в программе размещена числовая константа (константы). Она загружает значение константы в соответствии со значением смещения в младшие 16 битов аккумулятора.

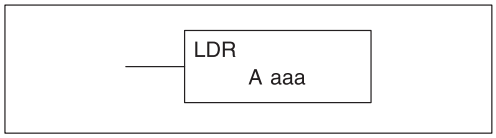


STR	X(IN)	1	↵								
LD	K(CON)	1	↵								
LD	SHFT	S	X	K(CON)	2	↵					
OUT	V	1	5	0	0	↵					
END	↵										
D	L	B	L	K(CON)	2	↵					
SHFT	N	C	O	N	SHFT	K(CON)	2	2	2	2	↵
SHFT	N	C	O	N	SHFT	K(CON)	2	3	2	3	↵
SHFT	N	C	O	N	SHFT	K(CON)	4	4	4	4	↵

Load Real Number (LDR)

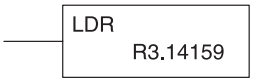
X	X	√
430	440	450

Команда Load Real Number загружает вещественное число, содержащееся в двух последовательных ячейках V-памяти, или 8-знаковую константу в аккумулятор.

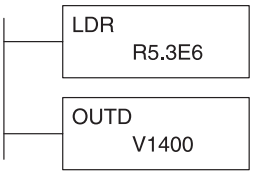


Тип данных операнда		Диапазон DL450
	A	**aaa**
V-память	V	Вся V-память (см. стр. 3-42)
Указатель	P	Вся V-память (см. стр. 3-42)
Вещественная константа	R	Полный 32-битный IEEE диапазон

DirectSOFT позволяет Вам прямо вводить вещественные числа, используя символ "R" перед числом для показа ввода вещественного числа. Вы можете вводить константу, как, например, Pi, показанную на примере внизу. Чтобы вводить отрицательные числа, используйте знак "минус" (-) после "R".

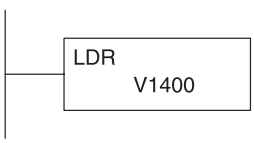


Для очень больших или очень маленьких чисел, Вы можете использовать экспоненциальный вид записи. Число, показанное ниже, - 5,3 миллиона. Команда OUTD хранит его в V1400 и V1401.



Эти вещественные числа представлены в 32-битовом IEEE формате с плавающей точкой, поэтому они занимают две ячейки V-памяти, безотносительно к тому, большие это или маленькие числа! Если Вы смотрите вещественное число, хранящееся в восьмеричном, шестнадцатеричном или даже в двоично-десятичном форматах, то это число будет трудно дешифровать. Просто подобно всем другим типам чисел, Вы должны отслеживать ячейки памяти с вещественными числами, так что бы они позже могли быть прочтены соответствующими командами.

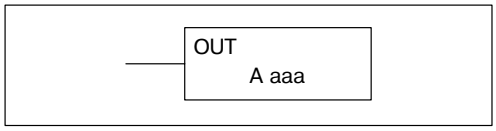
В предыдущем примере хранится вещественное число в V1400 и V1401. Предположим, что сейчас мы хотим извлечь это число. Для этого просто используйте команду Load Real с типом данных V, как показано ниже. Теперь мы можем выполнять математические операции над ним или преобразовать его в двоичное число.



Out (OUT)

√	√	√
430	440	450

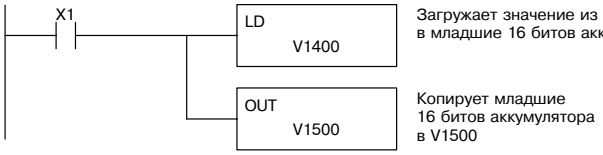
Команда Out - это 16-битовая команда, которая копирует значение из младших 16 битов аккумулятора в заданную ячейку V-памяти (Aaaa).



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)

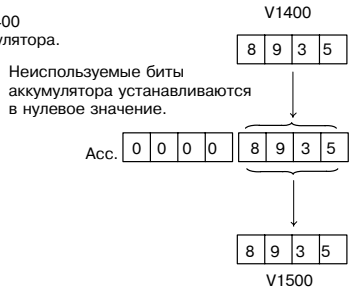
В следующем примере, когда X1 включен, значение в V1400 будет загружено в младшие 16 битов аккумулятора при использовании команды Load. Значение из младших 16 битов аккумулятора копируются в V1500 командой Out.

*Direct*SOFT



Загружает значение из V1400 в младшие 16 битов аккумулятора.

Копирует младшие 16 битов аккумулятора в V1500



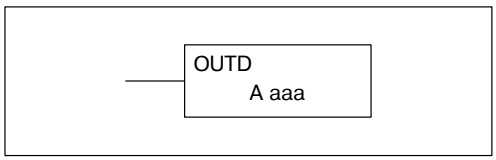
Набор на ручном программаторе

STR	X(IN)	1	↵			
LD	V	1	4	0	0	↵
OUT	V	1	5	0	0	↵

Out DOUBLE (OUTD)

√	√	√
430	440	450

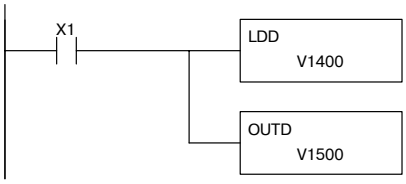
Команда Out Double - это 32-битовая команда, которая копирует значение из аккумулятора в две последовательных ячейки V-памяти с указанной стартовой ячейки (Aaaa).



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)

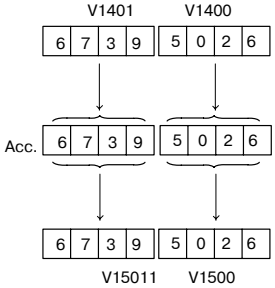
В следующем примере, когда X1 включен, 32-битовое значение в V1400 и V1401 будет загружено в аккумулятор, используя команду Load Double. Значение в аккумуляторе выводится в V1500 и V1501 командой Out Double.

*Direct*SOFT



Загружает значение из V1400 и V1401 в аккумулятора.

Копирует младшие 16 битов аккумулятора в V1500 и V1501



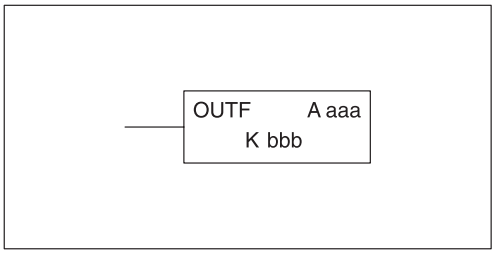
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Out Formatted (OUTF)

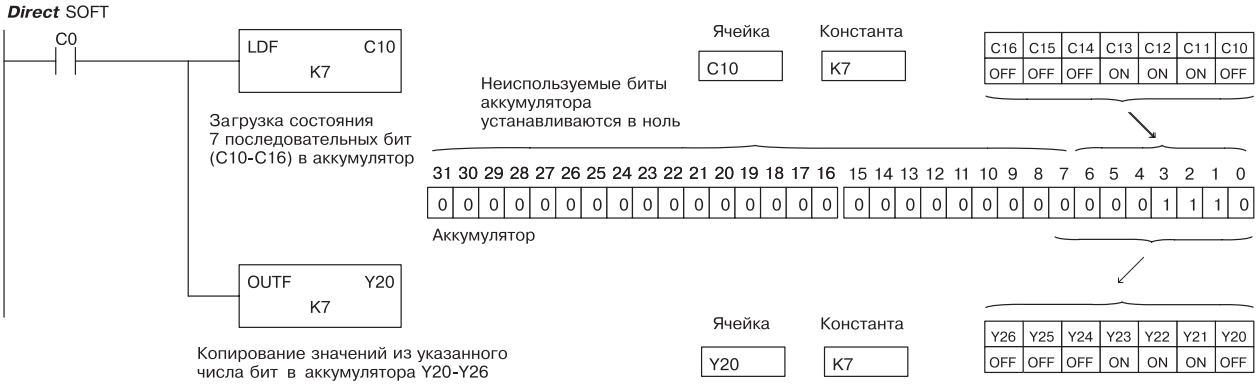
X	√	√
430	440	450

Команда Out Formatted выводит биты 1-32 из аккумулятора в указанные дискретные ячейки памяти. Для вывода команды требуется определить стартовую ячейку (Aaaa) для адреса и количество битов (Kbbb).



Тип данных операнда		Диапазон DL440		Диапазон DL450	
	A	**aaa**	**bbb**	**aaa**	**bbb**
Входы	X	0-477	--	0-1777	--
Выходы	Y	0-477	--	0-1777	--
Реле управления	C	0-1777	--	0-3777	--
Глобальный ввод/вывод	GX	0-1777	--	0-2777	--
Константа	K	--	1-32	--	1-32

В следующем примере, когда C0 включен, двоичный набор C10-C16 (7 бит) будет загружен в аккумулятор с использованием команды Load Formatted. Младшие 6 бит аккумулятора выводятся на Y20-Y26, используя команду Out Formatted.



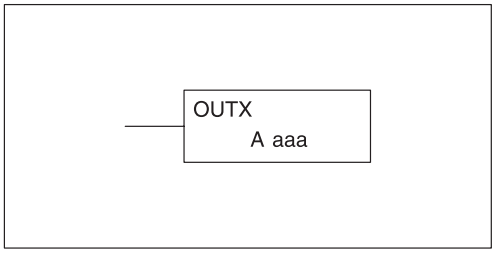
Набор на ручном программаторе

STR	C(CR)	0	↵						
LD	SHFT	F	SHFT	C(CR)	1	0	K(CON)	7	↵
OUT	SHFT	F	SHFT	Y(OUT)	2	0	K(CON)	7	↵

Out Indexed (OUTX)

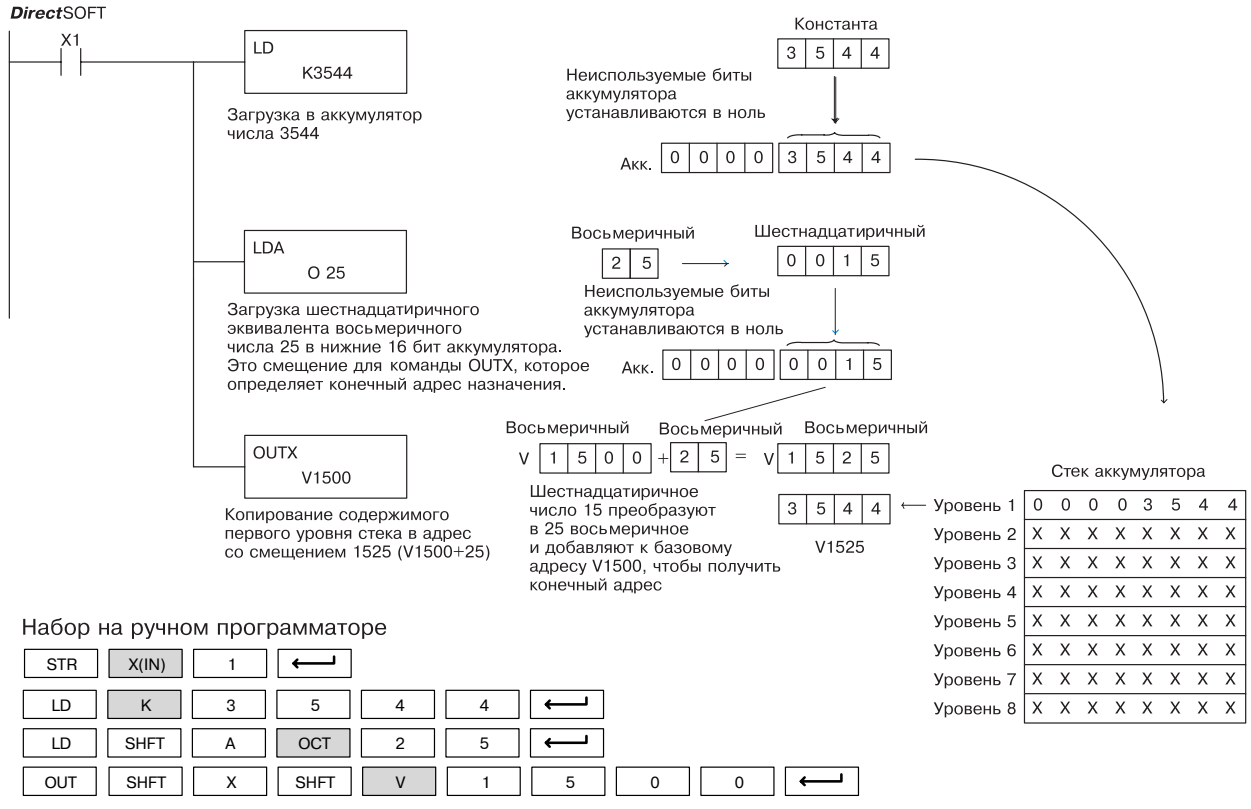
√	√	√
430	440	450

Команда Out Indexed является 16-битовой командой. Она копирует 16 бит или 4-значное число из первого уровня стека аккумулятора в ячейку с адресом, равным заданной ячейке V-памяти плюс смещение, находящееся в аккумуляторе. Эта команда интерпретирует значение смещения как шестнадцатеричное число. Старшие 16 битов аккумулятора устанавливаются в нуль.



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	aaa	aaa	aaa
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)

В следующем примере, когда X1 включен, константа 3544 загружается в аккумулятор. Это число является значением, которое будет выведено в ячейку V-памяти с определенным смещением (V1525). Когда выполнится команда Load Address, значение 3544 будет помещено в стек. Помните, что две последовательных команды Load поместят значение первой команды в стек. Команда Load Address преобразовывает восьмеричное число 25 в шестнадцатеричное число 15 и помещает значение в аккумулятор. Команда Out Indexed выводит значение 3544, которое находится на первом уровне стека аккумулятора, в V1525.



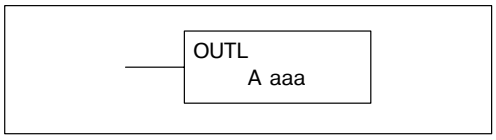
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	K	3	5	4	4	↵			
LD	SHFT	A	OCT	2	5	↵			
OUT	SHFT	X	SHFT	V	1	5	0	0	↵

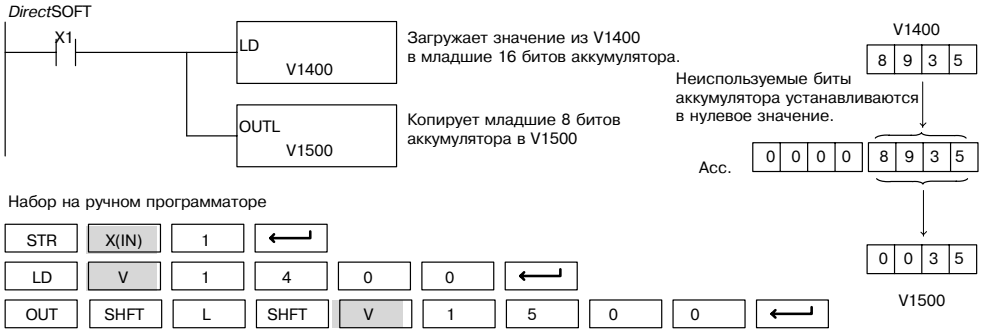
Out Least (OUTL)

X	X	√
430	440	450

Команда Out Least копирует значение в младших восьми битах аккумулятора в младшие восемь битов заданной ячейки V-памяти (то есть она копирует младший байт младшего слова аккумулятора)



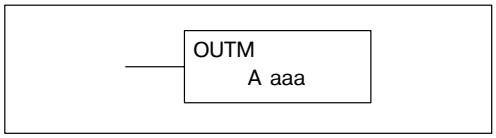
В следующем примере, когда X1 включен, значение V1400 будет загружаться в младшие 16 битов аккумулятора при использовании команды Load. Значение в младших 8 битах аккумулятора копируются в V1500 командой Out Least.



Out Most (OUTM)

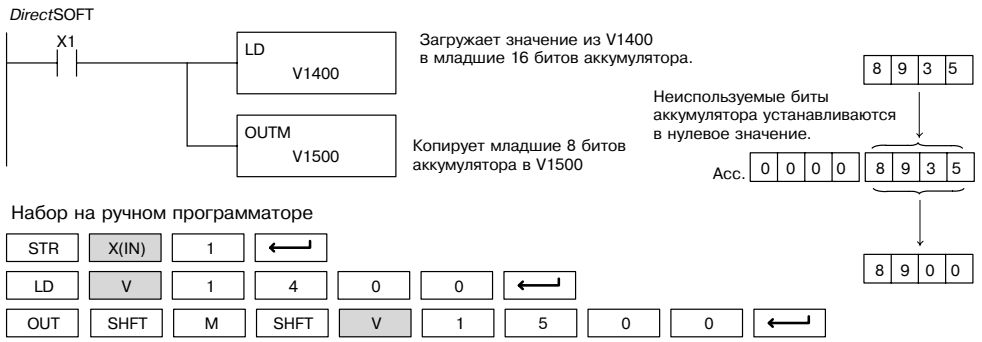
X	X	√
430	440	450

Команда Out Most копирует значение в старших восьми битах младших 16 битах аккумулятора в старшие восемь битов заданной ячейки V-памяти (то есть она копирует старший байт младшего слова аккумулятора).



Тип данных операнда		Диапазон DL450
	A	**aaa**
V-память	V	Вся V-память (см. стр. 3-42)
Указатель	P	Вся V-память (см. стр. 3-42)

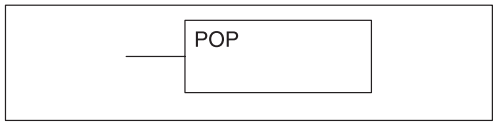
В следующем примере, когда X1 включено, значение V1400 будет загружаться в младшие 16 битов аккумулятора при использовании команды Load. Значение в старших 8 битах младших 16 битов аккумулятора копируются в V1500 командой Out Most.



Pop (POP)

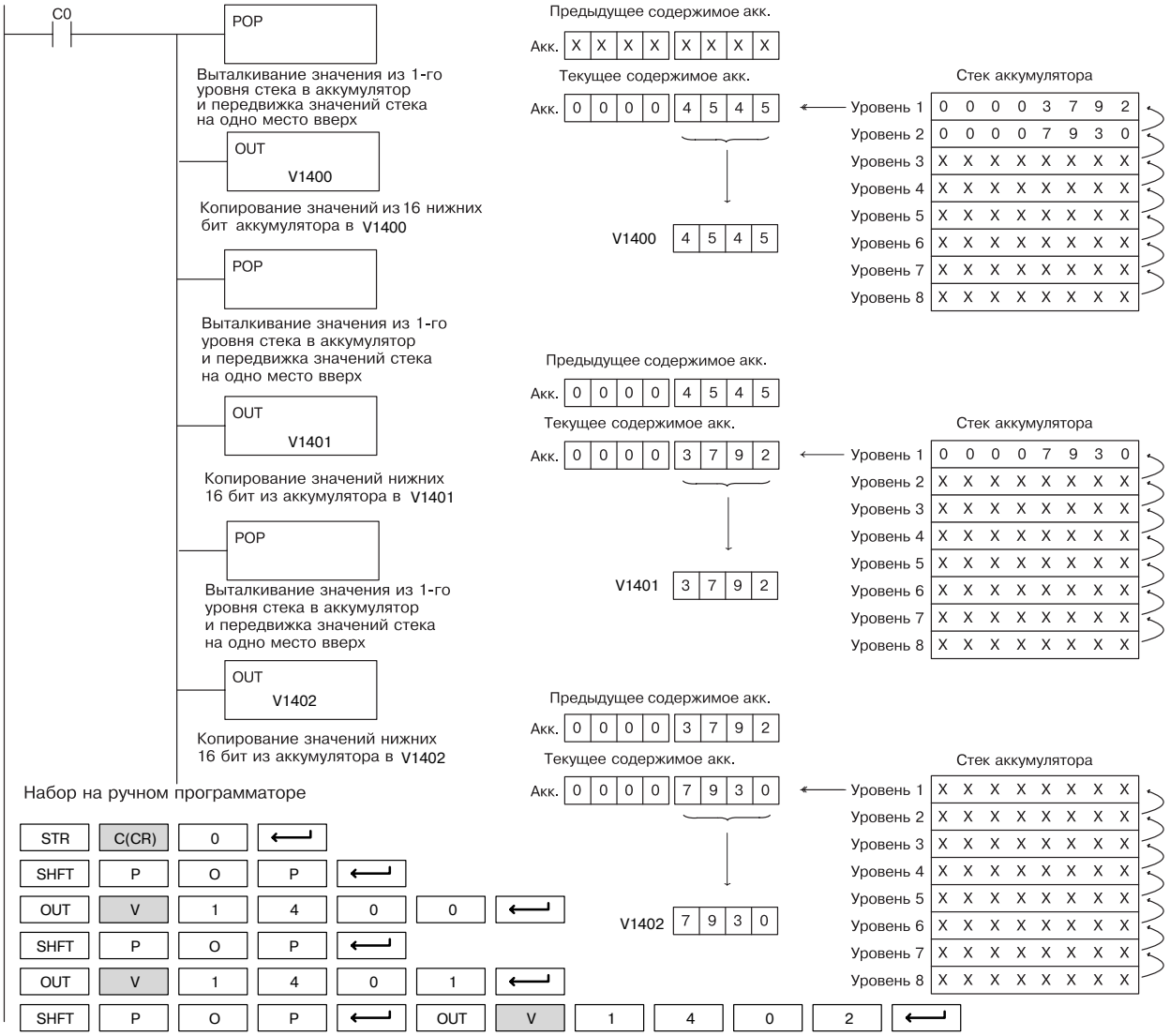
√	√	√
430	440	450

Команда Pop перемещает значение с первого уровня стека аккумулятора (32 бита) в аккумулятор и сдвигает каждое значение в стеке вверх на один уровень.



В примере ниже, когда C0 включен, значение 4545, которое было на вершине стека, перемещается в аккумулятор с использованием команды Pop. Значение выводится в V1400, используя команду Out. Следующая команда Pop перемещает значение 3792 в аккумулятор и выводит значение в V1401. Последняя команда Pop перемещает значение 7930 в аккумулятор и выводит значение в V1402. Обращаем внимание на то, что если значение в стеке было бы больше, чем 16 битов (4 знака), то использовалась бы команды Out Double и 2 ячейки V-памяти для каждой команды Out Double.

Флаги дискретных разрядов	Описание
SP63	Включено, когда значение, загруженное в аккумулятор любой командой, является нулем.

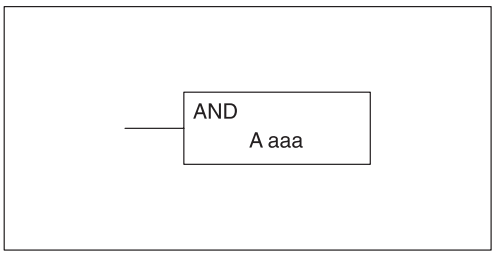


Логические команды аккумулятора

And (AND)

√	√	√
430	440	450

Команда And - это 16-битовая команда, которая выполняет операцию логического И над значением, расположенным в младших 16 битах аккумулятора, и значением в заданной ячейке V-памяти (Аааа). Результат находится в аккумуляторе. Дискретный флаг состояния указывает, является ли результат команды And нулем.

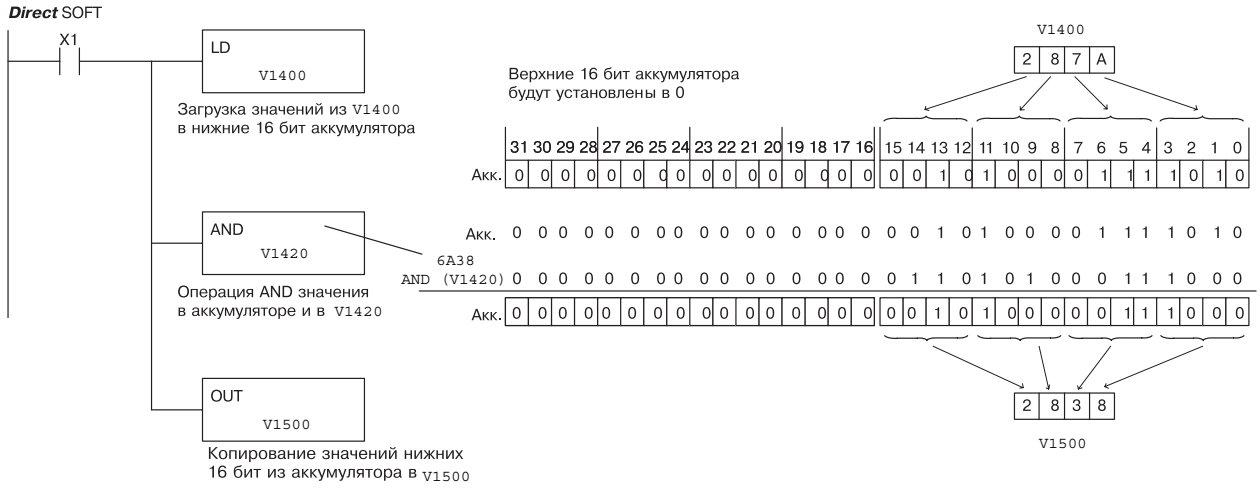


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	–	Все (см. стр. 3-41)	Все (см. стр. 3-42)

Флаги дискретных разрядов	Описание
SP63	Будет включено, если результат в аккумуляторе является нулем

ПРИМЕЧАНИЕ. Флаги состояния доступны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 будет загружено в аккумулятор, используя команду Load. Выполняется операция логического И над значением из аккумулятора и значением из V1420. Значение из младших 16 бит аккумулятора выводится в V1500, используя команду Out.



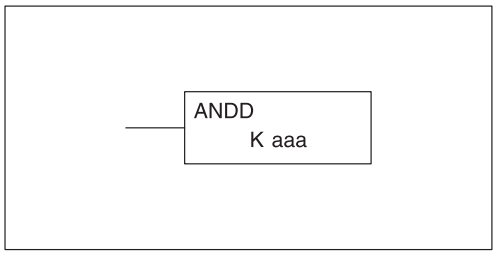
Набор на ручном программаторе

STR	X(IN)	1	↵			
LD	V	1	4	0	0	↵
AND	V	1	4	2	0	↵
OUT	V	1	5	0	0	↵

And Double (ANDD)

√	√	√
430	440	450

Команда And Double - это 32-битовая команда, которая выполняет операцию логического И над значением из аккумулятора с двумя последовательными ячейками V-памяти или со значением константы (Аааа, максимум 8 знаков). Результат находится в аккумуляторе. Дискретные флаги состояния указывают, является ли результат команды And Double нулем или отрицательным числом (наиболее значимый бит включен).

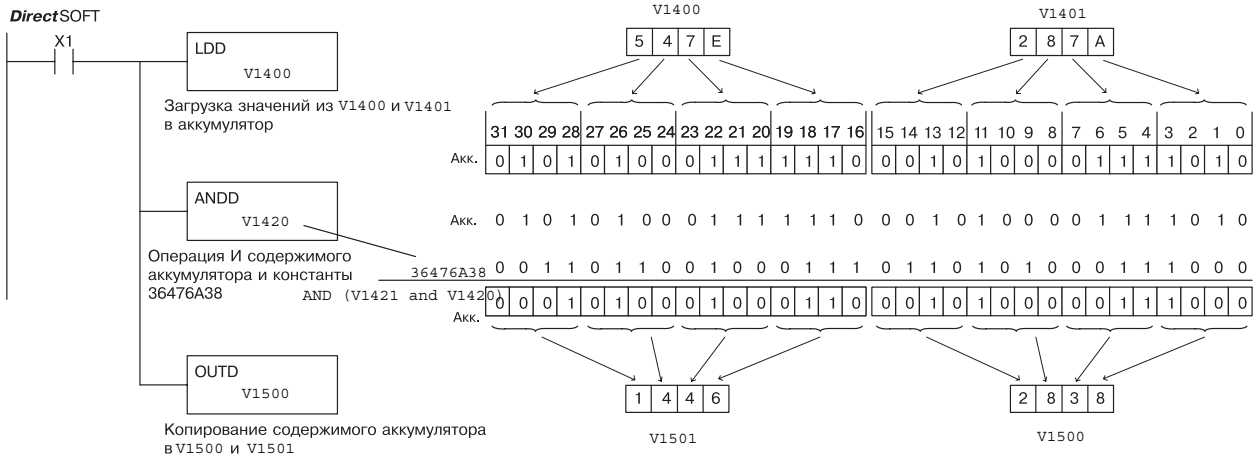


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**bb**	**bb**
V-память	V	—	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	—	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	0-FFFF	0-FFFF	0-FFFF

Флаги дискретных разрядов	Описание
SP63	Будет включен, если результат в аккумуляторе является нулем
SP70	Будет включен, если результат в аккумуляторе является отрицательным

ПРИМЕЧАНИЕ. Флаги состояния доступны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор при использовании команды Load Double. Значение в аккумуляторе является результатом операции логического И над значениями V1400 и V1401 при использовании команды And Double. Значение из аккумулятора выводится в V1500 и V1501, используя команду Out Double.



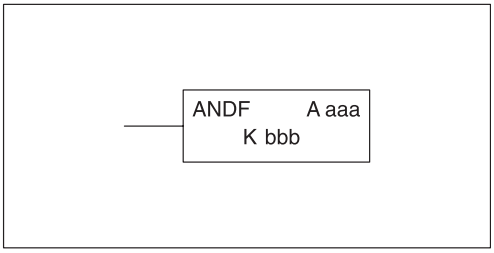
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
AND	SHFT	D	SHFT	V	1	4	2	0	↵
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

And Formatted (ANDF)

X	√	√
430	440	450

Команда And Formatted выполняет операцию логического И над двоичным значением из аккумулятора и определенным диапазоном дискретных битов памяти (1-32). В команде должны быть определены стартовая ячейка (Aaaa) и количество битов (Kbbb). Флаги дискретного состояния указывают, является ли результат нулем или отрицательным числом (старший бит = 1).

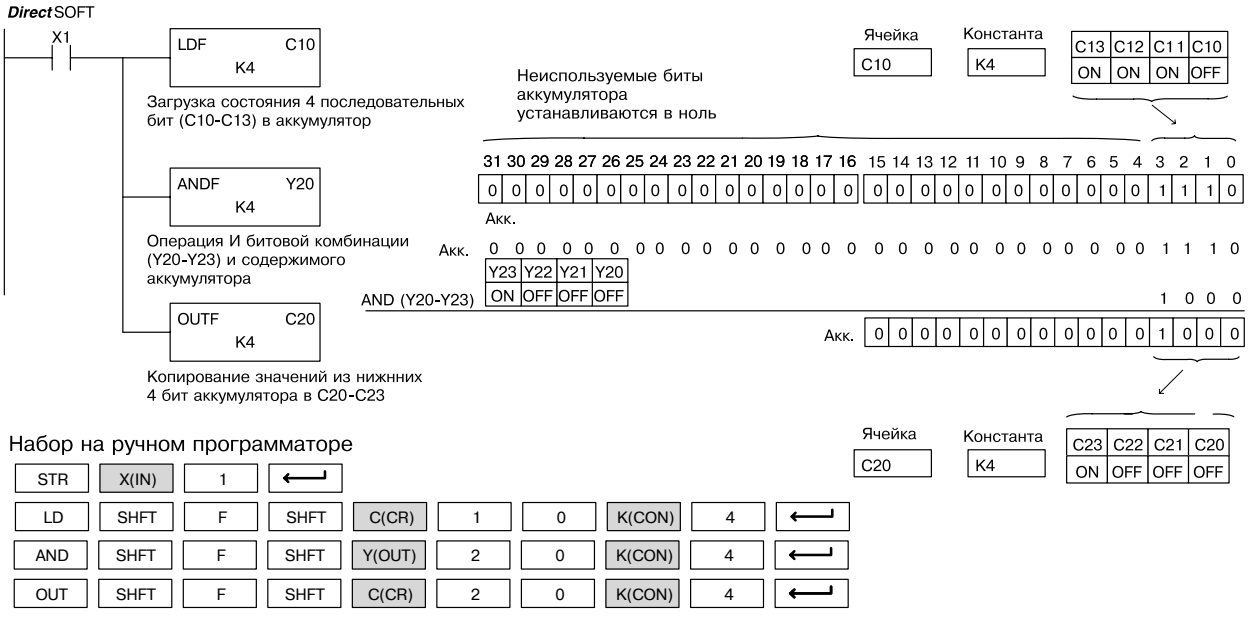


Тип данных операнда		Диапазон DL440		Диапазон DL450	
	A/B	aaa	bbb	aaa	bbb
Входы	X	0-477	--	0-1777	--
Выходы	Y	0-477	--	0-1777	--
Реле управления	C	0-1777	--	0-3777	--
Биты стадии	S	0-1777	--	0-1777	--
Биты таймера	T	0-377	--	0-377	--
Биты счетчика	CT	0-177	--	0-377	--
Специальные реле	SP	0-137 320-717	--	0-137 320-717	--
Глобальный ввод/вывод	GX	0-1777	--	0-2777	--
Константа	K	--	1-32	--	1-32

Флаги дискретных разрядов	Описание
SP63	Будет включен, если результат в аккумуляторе является нулем
SP70	Будет включен, если результат в аккумуляторе является отрицательным

ПРИМЕЧАНИЕ. Флаги состояния доступны только до того момента, когда будет выполнена другая команда, использующая тот же самый флаг.

В следующем примере, когда X1 включен, команда Load Formatted загружает C10-C13 (4 двоичных бита) в аккумулятор. Выполняется операция логического И над содержимым аккумулятора и битовым набором из Y20-Y23 при использовании команды And Formatted. Команда Out Formatted выводит младшие четыре бита аккумулятора в C20-C23.



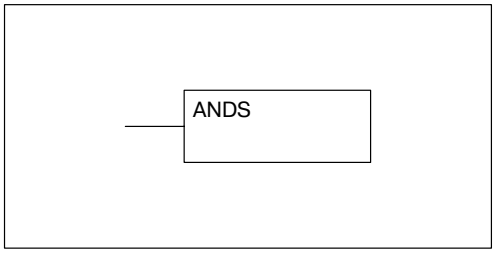
Набор на ручном программаторе

STR	X(IN)	1	↵					
LD	SHFT	F	SHFT	C(CR)	1	0	K(CON)	4	↵
AND	SHFT	F	SHFT	Y(OUT)	2	0	K(CON)	4	↵
OUT	SHFT	F	SHFT	C(CR)	2	0	K(CON)	4	↵

And with Stack (ANDS)

X	√	√
430	440	450

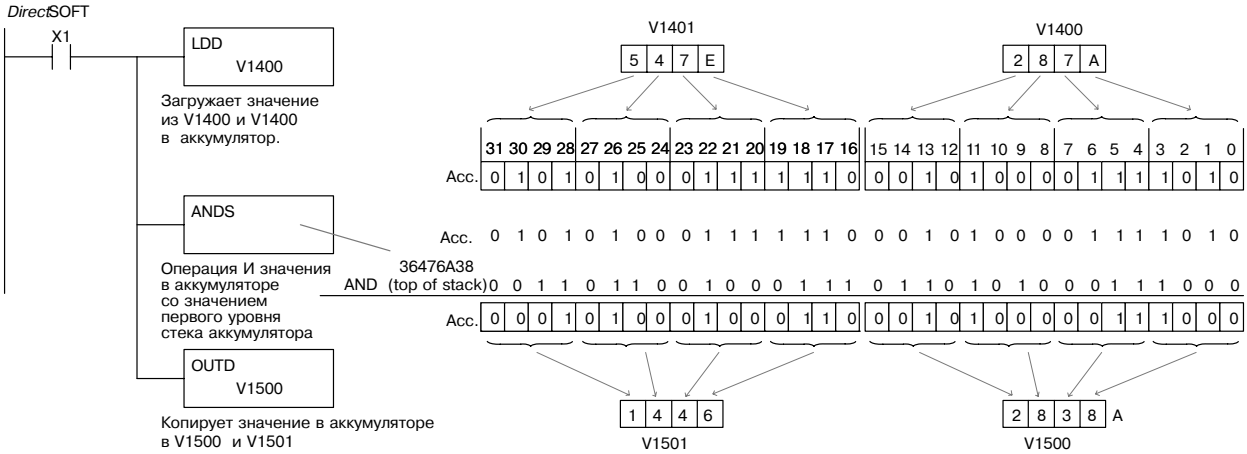
Команда And with Stack является 32-битовой командой, которая реализует логическое И значения в аккумуляторе с первым уровнем стека аккумулятора. Результат сохраняется в аккумуляторе. Значение в первом уровне стека аккумулятора удаляется из стека, а все значения перемещаются вверх на один уровень. Флаги дискретного состояния команды And with Stack указывают, является ли результат нулем или отрицательным числом (самый старший бит включен).



Флаги дискретных разрядов	Описание
SP63	Будет включен, если результат в аккумуляторе является нулем
SP70	Будет включен, если результат в аккумуляторе является отрицательным

ПРИМЕЧАНИЕ. Флаги состояния доступны только до того момента, когда будет выполнена другая команда, использующая тот же самый флаг.

В следующем примере, когда X1 включено, с двоичным значением в аккумуляторе и с двоичным значением в первом уровне стека аккумулятора будет проведена логическая операция И. Результат сохранится в аккумуляторе. 32-битовое значение затем будет помещено в ячейки в V1500 и V1501.



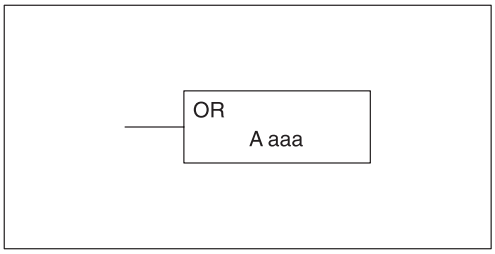
Набор на ручном программаторе

STR	X(IN)	1	⏎						
LD	SHFT	D	SHFT	V	1	4	0	0	⏎
AND	SHFT	S	⏎						
OUT	SHFT	D	SHFT	V	1	5	0	0	⏎

Or (OR)

√	√	√
430	440	450

Команда Or - это 16-битовая команда, которая выполняет операцию логического ИЛИ над значением, расположенным в младших 16 битах аккумулятора, и значением в заданной ячейке V-памяти (Аааа). Результат сохраняется в аккумуляторе. Флаг дискретного состояния указывает, является ли результат команды Or нулем.

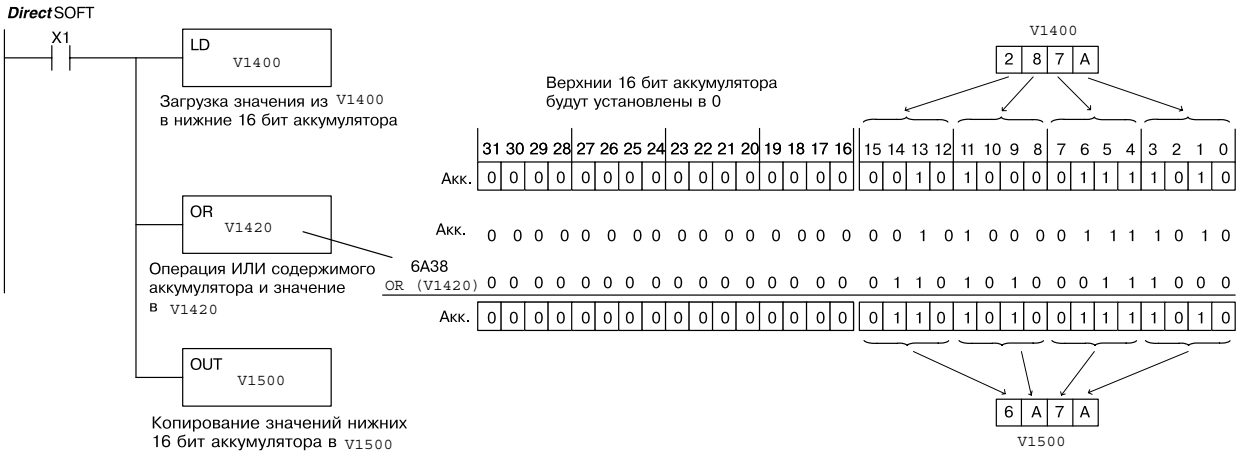


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	aaa	aaa	aaa
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	—	Все (см. стр. 3-41)	Все (см. стр. 3-42)

Флаги дискретных разрядов	Описание
SP63	Будет включено, если результат в аккумуляторе является нулем

ПРИМЕЧАНИЕ. Флаги состояния доступны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 будет загружено в аккумулятор командой Load. Выполняется операция логического ИЛИ над значением в аккумуляторе и значением в V1420 при использовании команды Or. Значение младших 16 битов аккумулятора выводится в V1500 командой Out.



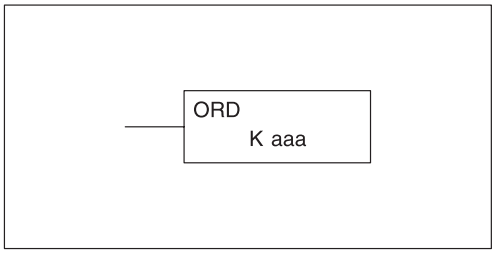
Набор на ручном программаторе

STR	X(IN)	1	↵			
LD	V	1	4	0	0	↵
OR	V	1	4	2	0	↵
OUT	V	1	5	0	0	↵

Or Double (ORD)

√	√	√
430	440	450

Команда Or Double - это 32-битовая команда, которая выполняет операцию логического ИЛИ над значением из аккумулятора и значением (Аааа), которое является либо двумя последовательными ячейками V-памяти или 8-значной (максимум) константой. Результат сохраняется в аккумуляторе. Флаги дискретного состояния указывают, является ли результат команды Or Double нулем или отрицательным числом (самый старший бит включен).

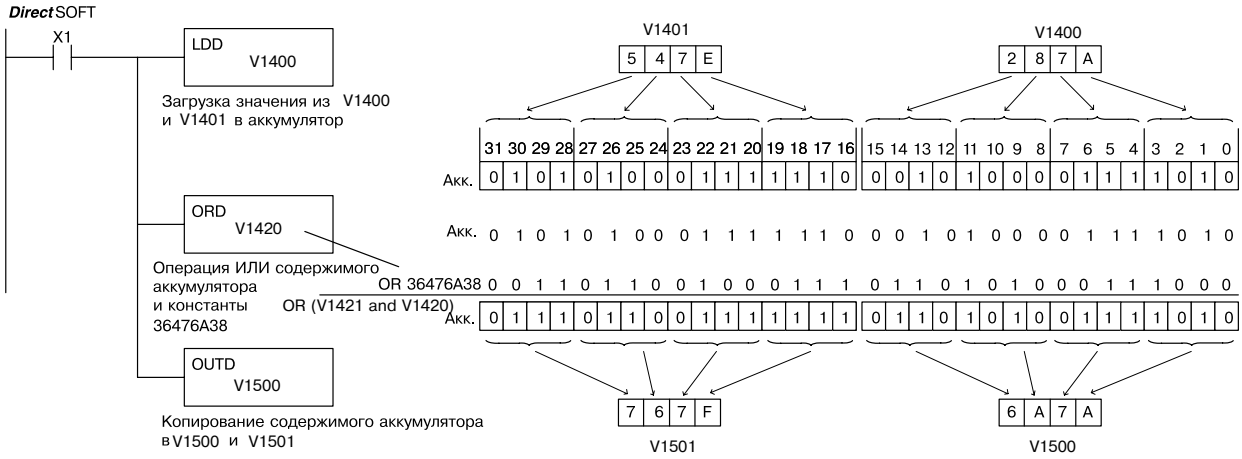


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**bb**	**bb**
V-память	V	—	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	—	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	0-FFFF	0-FFFF	0-FFFF

Флаги дискретных разрядов	Описание
SP63	Будет включен, если результат в аккумуляторе является нулем
SP70	Будет включен, если результат в аккумуляторе является отрицательным

ПРИМЕЧАНИЕ. Флаги состояния доступны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Выполняется операция логического ИЛИ над значением в аккумуляторе и значениями в V1420 и V1421 с использованием команды Or Double. Значение из аккумулятора выводится в V1500 и V1501 командой Out Double.



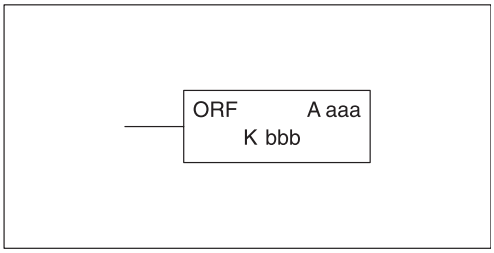
Набор на ручном программаторе

STR	X(IN)	1	⏎						
LD	SHFT	D	SHFT	V	1	4	0	0	⏎
OR	SHFT	D	SHFT	V	1	4	2	0	⏎
OUT	SHFT	D	SHFT	V	1	5	0	0	⏎

Or Formatted (ORF)

X	√	√
430	440	450

Команда Or Formatted выполняет операцию логического ИЛИ над двоичным значением в аккумуляторе и заданным диапазоном дискретных битов (1-32). В команде должны быть определены стартовая ячейка (Aaaa) и количество битов (Kbbb). Флаги дискретного состояния указывают, является ли результат нулем или отрицательным числом (старший бит = 1).

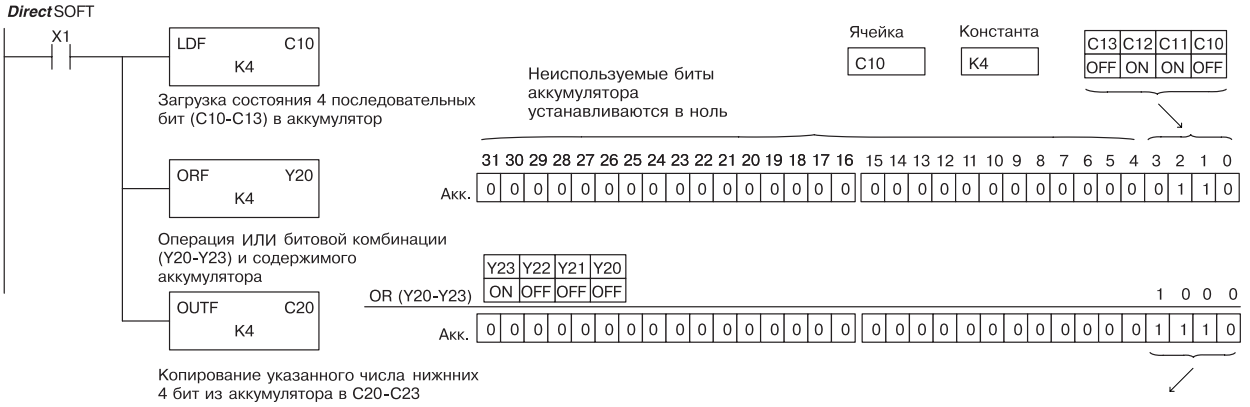


Тип данных операнда		Диапазон DL440		Диапазон DL450	
	A/B	aaa	bbb	aaa	bbb
Входы	X	0-477	--	0-1777	--
Выходы	Y	0-477	--	0-1777	--
Реле управления	C	0-1777	--	0-3777	--
Биты стадии	S	0-1777	--	0-1777	--
Биты таймера	T	0-377	--	0-377	--
Биты счетчика	CT	0-177	--	0-377	--
Специальные реле	SP	0-137 320-717	--	0-137 320-717	--
Глобальный ввод/вывод	GX	0-1777	--	0-2777	--
Константа	K	--	1-32	--	1-32

Флаги дискретных разрядов	Описание
SP63	Будет включен, если результат в аккумуляторе является нулем
SP70	Будет включен, если результат в аккумуляторе является отрицательным

ПРИМЕЧАНИЕ. Флаги состояния доступны только до того момента, когда будет выполнена другая команда, использующая тот же самый флаг.

В следующем примере, когда X1 включен, команда Load Formatted загружает C10-C13 (4 двоичных бита) в аккумулятор. Команда Or Formatted выполняет операцию логического ИЛИ над содержимым аккумулятора и битовой комбинацией из Y20-Y23. Команда Out Formatted выводит младшие четыре бита аккумулятора в C20-C23.



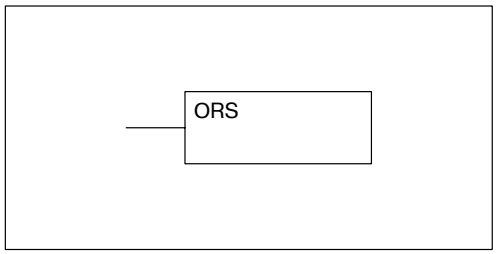
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	F	SHFT	C(CR)	1	0	K(CON)	4	↵
OR	SHFT	F	SHFT	Y(OUT)	2	0	K(CON)	4	↵
OUT	SHFT	F	SHFT	C(CR)	2	0	K(CON)	4	↵

Or with Stack (ORS)

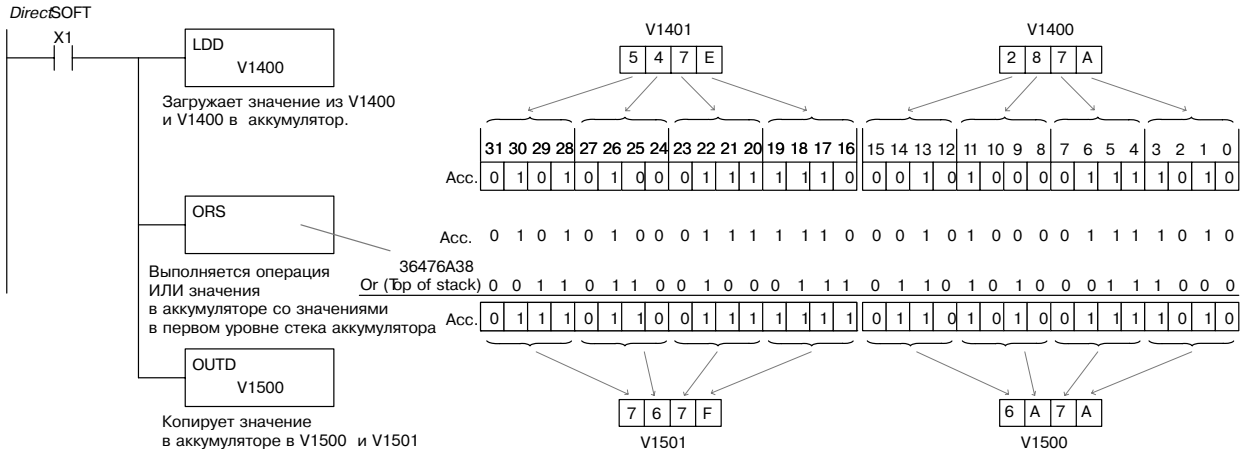
X	√	√
430	440	450

Команда Or with Stack является 32-битовой командой, которая реализует логическое ИЛИ значения в аккумуляторе с первым уровнем стека аккумулятора. Результат сохраняется в аккумуляторе. Значение в первом уровне стека аккумулятора удаляется из стека, а все значения перемещаются вверх на один уровень. Флаги дискретного состояния команды Or with Stack указывают, является ли результат нулем или отрицательным числом (самый старший бит включен).



Флаги дискретных разрядов	Описание
SP63	Будет включен, если результат в аккумуляторе является нулем
SP70	Будет включен, если результат в аккумуляторе является отрицательным

В следующем примере, когда X1 включен, с двоичным значением аккумулятора и двоичным значением первого уровня стека будет выполнена логическая операция ИЛИ. Результат сохранится в аккумуляторе.



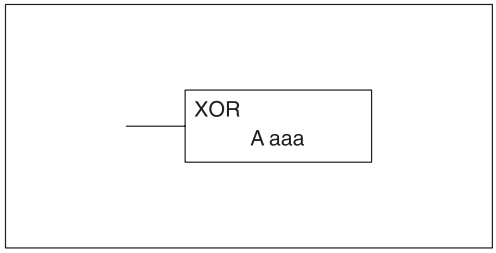
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
OR	SHFT	S	↵						
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Exclusive Or (XOR)

√	√	√
430	440	450

Команда Exclusive Or - это 16-битовая команда, которая выполняет операцию исключающего ИЛИ над значением, расположенным в младших 16 битах аккумулятора, и значением в заданной ячейке V-памяти (Аааа). Флаг дискретного состояния указывает, является ли результат команды XOR нулем.

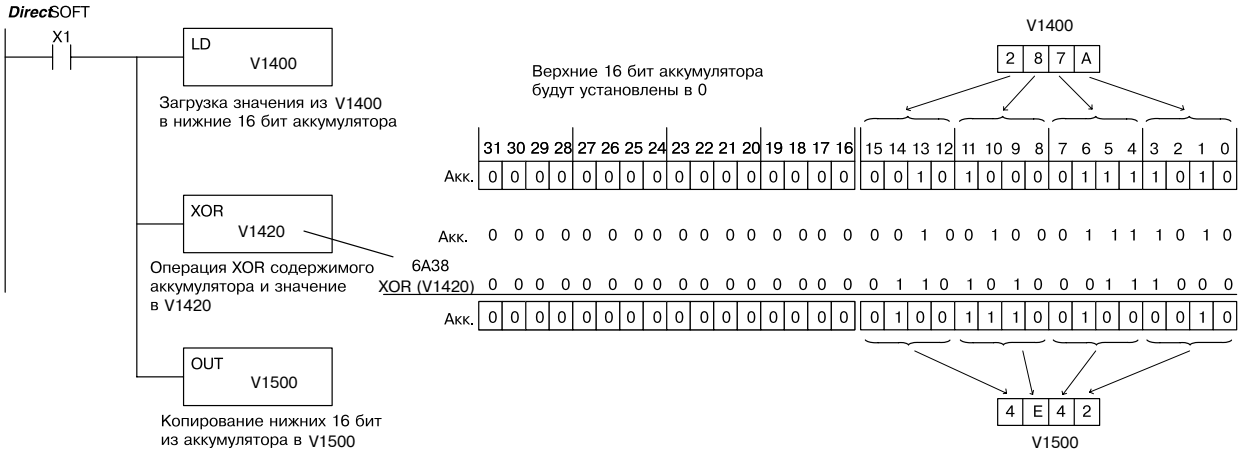


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	—	Все (см. стр. 3-41)	Все (см. стр. 3-42)

Флаги дискретных разрядов	Описание
SP63	Будет включено, если результат в аккумуляторе является нулем

ПРИМЕЧАНИЕ. Флаги состояния доступны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 будет загружено в аккумулятор командой Load. Выполняется операция исключающего ИЛИ над значением из аккумулятора и значением из V1420 с использованием команды Exclusive Or. Значение младших 16 битов аккумулятора выводится в V1500 командой Out.



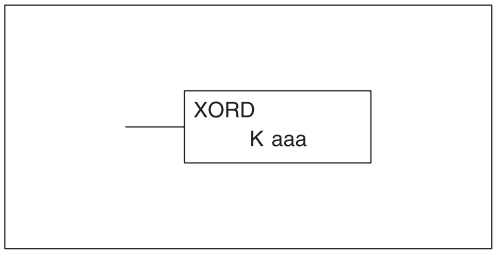
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	V	1	4	0	0	↵			
SHFT	X	SHFT	OR	V	1	4	2	0	↵
OUT	V	1	5	0	0	↵			

Exclusive Or Double (XORD)

√	√	√
430	440	450

Команда Exclusive Or Double - это 32-битовая команда, которая выполняет операцию исключающего ИЛИ над значением из аккумулятора и значения (Аaaa), которым являются либо две последовательные ячейки V-памяти или 8-значная (максимум) константа. Результат сохраняется в аккумуляторе. Флаги дискретного состояния указывают, является ли результат команды Exclusive Or Double нулем или отрицательным числом (старший бит включен).

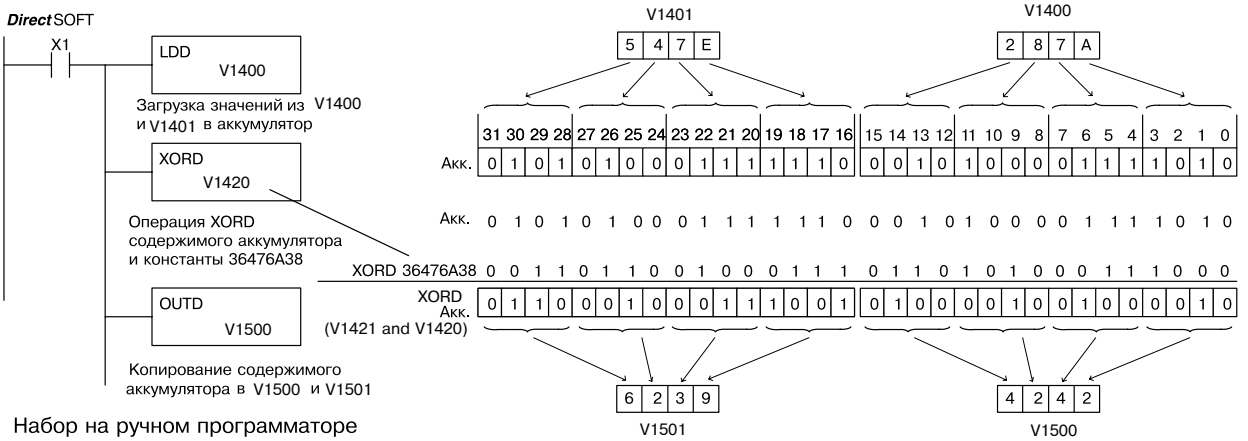


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**bb**	**bb**
V-память	V	—	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	—	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	0-FFFF	0-FFFF	0-FFFF

Флаги дискретных разрядов	Описание
SP63	Будет включен, если результат в аккумуляторе является нулем
SP70	Будет включен, если результат в аккумуляторе является отрицательным

ПРИМЕЧАНИЕ. Флаги состояния доступны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Выполняется операция исключающего ИЛИ над значением из аккумулятора и значениями в V1420 и V1421 с использованием команды Exclusive Or Double. Значение из аккумулятора выводится в V1500 и V1501 командой Out Double.



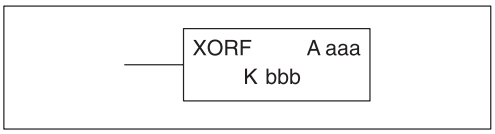
Набор на ручном программаторе

STR	X(IN)	1	↵									
LD	SHFT	D	SHFT	V	1	4	0	0	↵			
SHFT	X	SHFT	OR	SHFT	D	SHFT	V	1	4	2	0	↵
OUT	SHFT	D	SHFT	V	1	5	0	0	↵			

Exclusive Or Formatted (XORF)

X	√	√
430	440	450

Команда Exclusive Or Formatted выполняет операцию исключающего ИЛИ над двоичным значением из аккумулятора и определенным диапазоном дискретных битов памяти (1-32).



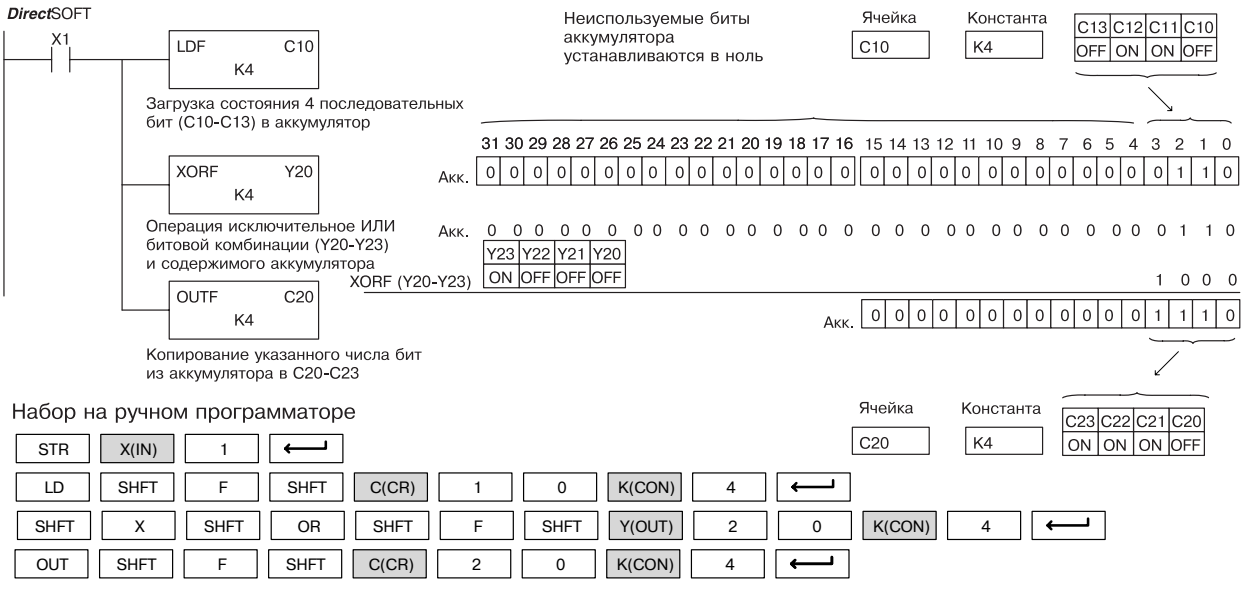
Тип данных операнда		Диапазон DL440		Диапазон DL450	
	A/B	aaa	bbb	aaa	bbb
Входы	X	0-477	--	0-1777	--
Выходы	Y	0-477	--	0-1777	--
Реле управления	C	0-1777	--	0-3777	--
Биты стадии	S	0-1777	--	0-1777	--
Биты таймера	T	0-377	--	0-377	--
Биты счетчика	CT	0-177	--	0-377	--
Специальные реле	SP	0-137 320-717	--	0-137 320-717	--
Глобальный ввод/вывод	GX	0-1777	--	0-2777	--
Константа	K	--	1-32	--	1-32

Флаги дискретных разрядов	Описание
SP63	Будет включен, если результат в аккумуляторе является нулем
SP70	Будет включен, если результат в аккумуляторе является отрицательным

В команде должны быть определены стартовая ячейка (Aaaa) и количество битов (Kbbb). Флаги дискретного состояния указывают, является ли результат команды Exclusive Or Formatted нулем или отрицательным числом (старший бит = 1).

ПРИМЕЧАНИЕ: Флаги состояния доступны только до того момента, когда будет выполнена другая команда, использующая тот же самый флаг.

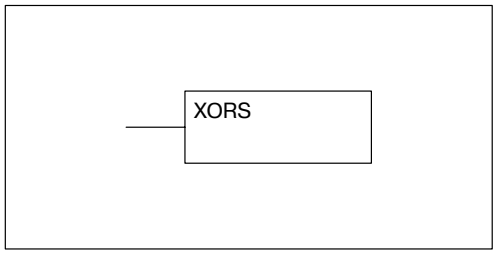
В следующем примере, когда X1 включен, двоичный набор C10-C13 (4 бита) будет загружен в аккумулятор командой Load Formatted. Команда Exclusive Or Formatted выполняет операцию исключающего ИЛИ над содержимым аккумулятора и битовым набором из Y20-Y23. Значение из младших четырех бит аккумулятора выводится в C20-C23 командой Out Formatted.



Exclusive Or with Stack (XORS)

√	√	√
430	440	450

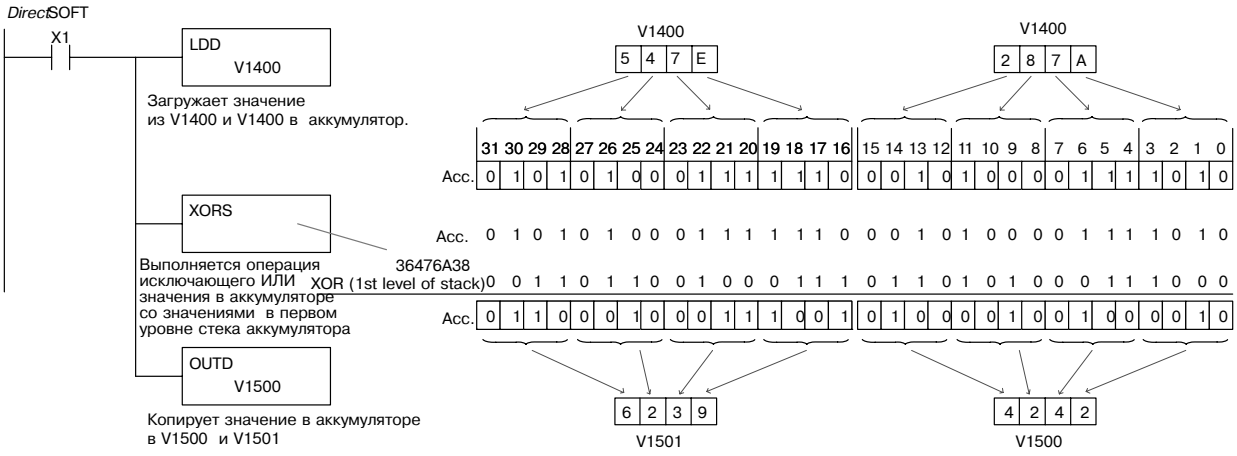
Команда Exclusive Or with Stack является 32-битовой командой, которая реализует исключающее ИЛИ значения в аккумуляторе и первого уровня стека аккумулятора. Результат сохраняется в аккумуляторе. Значение в первом уровне стека аккумулятора удаляется из стека, а все значения перемещаются вверх на один уровень. Флаги дискретного состояния команды Exclusive Or with Stack указывают, является ли результат нулем или отрицательным числом (самый старший бит включен).



Флаги дискретных разрядов	Описание
SP63	Будет включен, если результат в аккумуляторе является нулем
SP70	Будет включен, если результат в аккумуляторе является отрицательным

ПРИМЕЧАНИЕ. Флаги состояния доступны только до того момента, когда будет выполнена другая команда, использующая тот же самый флаг.

В следующем примере, когда X1 включен, с двоичным набором в аккумуляторе будет проведена операция исключающего ИЛИ с двоичным набором первого уровня стека аккумулятора. Результат сохраняется в аккумуляторе.



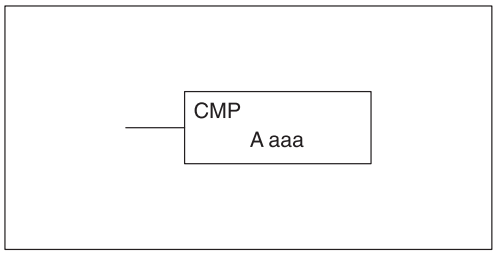
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SHFT	X	SHFT	OR	SHFT	S	↵			
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Compare (CMP)

√	√	√
430	440	450

Команда Compare - это 16-битовая команда, которая сравнивает значение из младших 16 битов аккумулятора со значением в определенной ячейке V-памяти (Aaaa). Соответствующий флаг состояния будет включен, показывая результат сравнения. Вы можете сравнивать как двоичные, так и двоично-десятичные числа, поскольку оба числа имеют один и тот же тип данных.

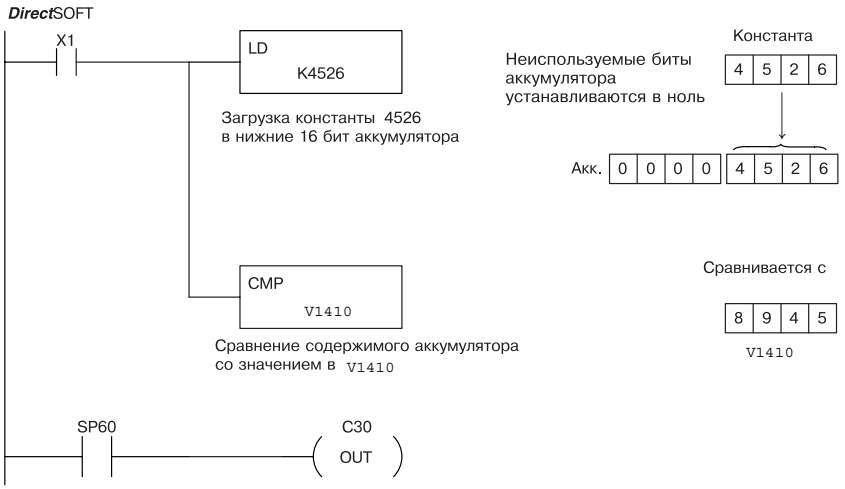


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	—	Все (см. стр. 3-41)	Все (см. стр. 3-42)

Флаги дискретных разрядов	Описание
SP60	Включено, когда значение в аккумуляторе меньше, чем значение команды
SP61	Включено, когда значение в аккумуляторе равно значению команды
SP62	Включено, когда значение в аккумуляторе больше, чем значение команды

ПРИМЕЧАНИЕ. Флаги состояния доступны только до того момента, когда будет выполнена другая команда, использующая тот же самый флаг.

В следующем примере, когда X1 включен, константа 4526 будет загружена в младшие 16 бит аккумулятора командой Load. Значение из аккумулятора сравнивается с значением в V1410, используя команду Compare. Соответствующий дискретный флаг состояния будет включен, показывая результат сравнения. В этом примере, если значение в аккумуляторе меньше, чем значение, определенное в команде Compare, то SP60 включит C30.



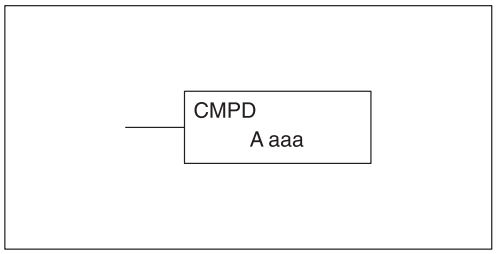
Набор на ручном программаторе

STR	X(IN)	1	↵			
LD	K(CON)	4	5	2	6	↵
CMP	V	1	4	1	0	↵
STR	SPCL	6	0	↵		
OUT	C(CR)	3	0	↵		

Compare Double (CMPD)

√	√	√
430	440	450

Команда Compare Double - это 32-битовая команда, которая сравнивает значение из аккумулятора со значением (Aaaa), которое является двумя последовательными ячейками V-памяти или константой (максимум 8 знаков). Соответствующий флаг состояния будет включен, показывая результат сравнения. Вы можете сравнивать как двоичные, так и двоично-десятичные числа, поскольку оба числа имеют один и тот же тип данных.

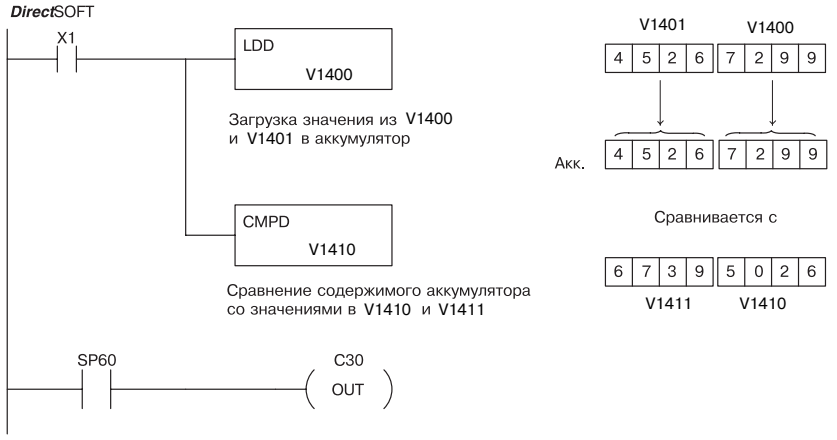


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**bb**	**bb**
V-память	V	—	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	—	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	0-FFFFFFFF	0-FFFFFFFF	0-FFFFFFFF

Флаги дискретных разрядов	Описание
SP60	Включено, когда значение в аккумуляторе меньше, чем значение команды
SP61	Включено, когда значение в аккумуляторе равно значению команды
SP62	Включено, когда значение в аккумуляторе больше, чем значение команды

ПРИМЕЧАНИЕ. Флаги состояния доступны только до того момента, когда будет выполнена другая команда, использующая тот же самый флаг.

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор с использованием команды Load Double. Значение в аккумуляторе сравнивается со значениями в V1400 и V1401 командой Compare Double. Соответствующий дискретный флаг состояния будет включен, показывая результат сравнения. В этом примере, если значение в аккумуляторе меньше, чем значение, определенное в команде Compare, то SP60 включит C30.



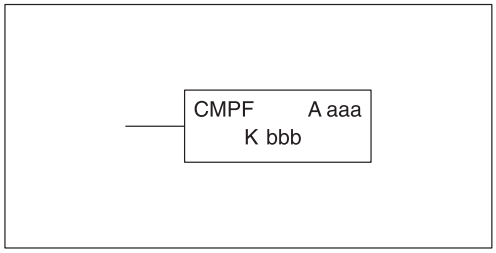
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
CMP	SHFT	D	SHFT	V	1	4	1	0	↵
STR	SPCL	6	0	↵					
OUT	C(CR)	3	0	↵					

Compare Formatted (CMPF)

X	√	√
430	440	450

Команда Compare Formatted сравнивает значение из аккумулятора с определенным числом дискретных разрядов (1-32). В команде должны быть определены стартовая ячейка (Aaaa) и количество сравниваемых битов (Kbbb). Соответствующий флаг состояния будет включен, показывая результат сравнения.

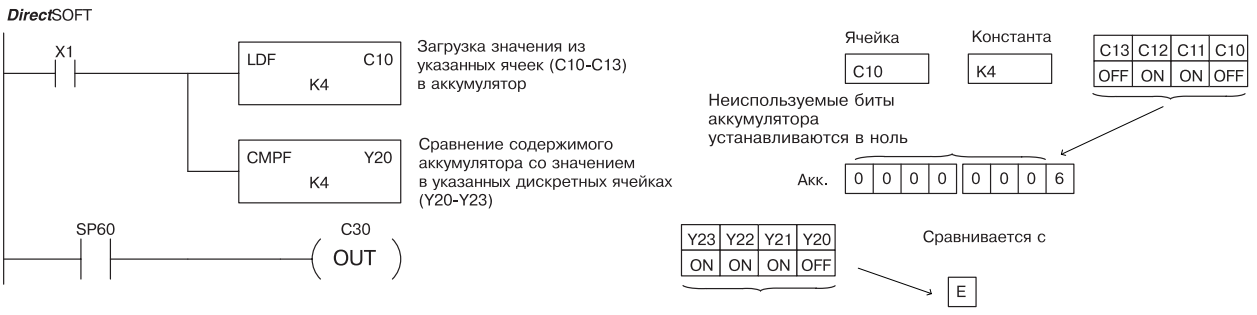


Тип данных операнда		Диапазон DL440		Диапазон DL450	
	A/B	aaa	bbb	aaa	bbb
Входы	X	0-477	--	0-1777	--
Выходы	Y	0-477	--	0-1777	--
Реле управления	C	0-1777	--	0-3777	--
Биты стадии	S	0-1777	--	0-1777	--
Биты таймера	T	0-377	--	0-377	--
Биты счетчика	CT	0-177	--	0-377	--
Специальные реле	SP	0-137 320-717	--	0-137 320-717	--
Глобальный ввод/вывод	GX	0-1777	--	0-2777	--
Константа	K	--	1-32	--	1-32

Флаги дискретных разрядов	Описание
SP60	Включено, когда значение в аккумуляторе меньше, чем значение команды
SP61	Включено, когда значение в аккумуляторе равно значению команды
SP62	Включено, когда значение в аккумуляторе больше, чем значение команды

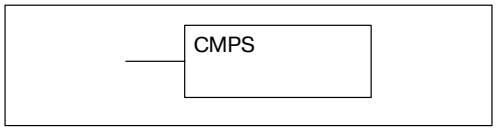
ПРИМЕЧАНИЕ. Флаги состояния доступны только до того момента, когда будет выполнена другая команда, использующая тот же самый флаг.

В следующем примере, когда X1 включено, команда Load Formatted загружает двоичное значение (6) из C10-C13 в аккумулятор. Команда Compare Formatted сравнивает значение в аккумуляторе со значением в Y20-Y23 (E шестнадцатеричное). Соответствующий дискретный флаг состояния будет включен, показывая результат сравнения. В этом примере, если значение в аккумуляторе меньше, чем значение, определенное в команде Compare, то SP60 включит C30.



Compare with Stack (CMPS)

√	√	√
430	440	450

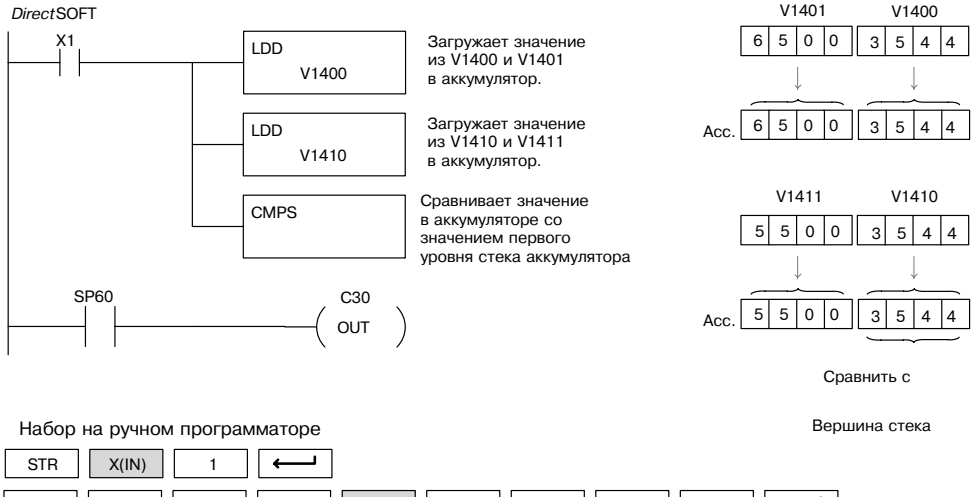


Команда Compare with Stack (Сравнить со стеком) является 32-битовой командой, которая сравнивает значение в аккумуляторе со значением в первом уровне стека.

Соответствующий флаг состояния будет включен, показывая результат сравнения. Это не влияет на значение в аккумуляторе.

Флаги дискретных разрядов	Описание
SP60	Включено, когда значение в аккумуляторе меньше, чем значение команды
SP61	Включено, когда значение в аккумуляторе равно значению команды
SP62	Включено, когда значение в аккумуляторе больше, чем значение команды

В следующем примере, когда X1 включен, значение в V1400 и V1401 загружено в аккумулятор с использованием команды Load Double. Значение в V1410 и V1411 загружено в аккумулятор командой Load Double. Значение, которое было загружено в аккумулятор из V1400 и V1401 находится на вершине стека, когда выполняется вторая команда Load. Значение в аккумуляторе сравнивается со значением в первом уровне стека командой Compare with Stack. Соответствующий дискретный флаг состояния будет включен, показывая результат сравнения. В этом примере, если значение в аккумуляторе меньше, чем значение в стеке, то SP60 включится, запуская C30.



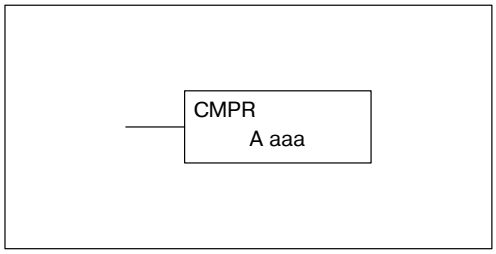
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
LD	SHFT	D	SHFT	V	1	4	1	0	↵
CMP	SHFT	S	↵						
STR	SPCL	6	0	↵					
OUT	C(CR)	3	0	↵					

Compare Real Number (CMPR)

X	X	√
430	440	450

Команда Compare Real Number сравнивает вещественное число из аккумулятора с двумя последовательными ячейками V-памяти, содержащими вещественное число. Соответствующий флаг состояния будет включен, показывая результат сравнения. Оба сравниваемых числа имеют величину 32 бита.



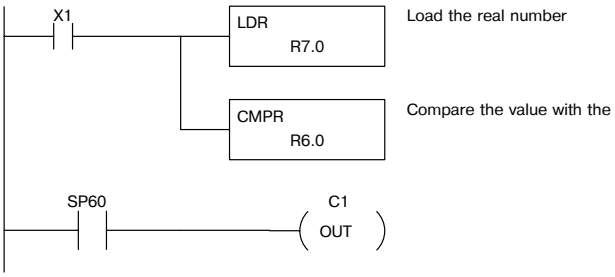
Тип данных операнда		Диапазон DL450
	A	**aaa**
V-память	V	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-42)
Константа	K	От -3.402823E+038 до + 3.402823E+038

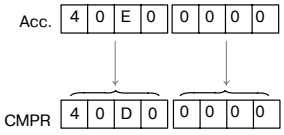
Флаги дискретных разрядов	Описание
SP60	Включен, когда значение в аккумуляторе меньше, чем значение в команде
SP61	Включен, когда значение в аккумуляторе равно значению в команде
SP62	Включен, когда значение в аккумуляторе больше, чем значение в команде
SP71	Включен, когда V-память, определенная как указатель (P), недоступна
SP75	Включен, когда выполняется команда с вещественным числом, но встретилось невещественное число.

ПРИМЕЧАНИЕ. Флаги состояния доступны только до того момента, когда будет выполнена другая команда, использующая тот же самый флаг.

В следующем примере, когда X1 включен, команда LDR загружает вещественное представление десятичного числа 7 в аккумулятор. Команда CMPR сравнивает содержимое аккумулятора с вещественным представлением десятичного числа 6. Так как 7 > 6, соответствующий дискретный флаг состояния включен (специальное реле SP60).

*Direct*SOFT





Набор на ручном программаторе

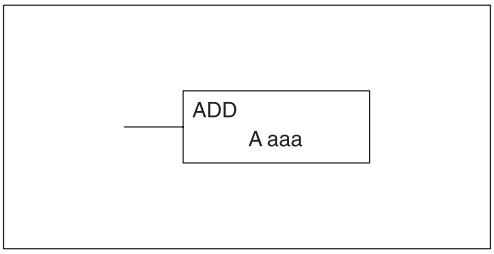
STR	X(IN)	1	LD	SHFT	D	SHFT	↵		
K(CON)	4	0	E	0	0	0	0	0	↵
CMP	SHFT	R	SHFT						
K(CON)	4	0	D	0	0	0	0	0	↵
STR	SPCL	6	0	↵					
OUT	C(CR)	3	0	↵					

Математические команды

Add (ADD)

√	√	√
430	440	450

Add - 16-битовая команда, которая складывает двоично-десятичное значение в аккумуляторе с двоично-десятичным значением в ячейке V-памяти (Aaaa). Результат находится в аккумуляторе.

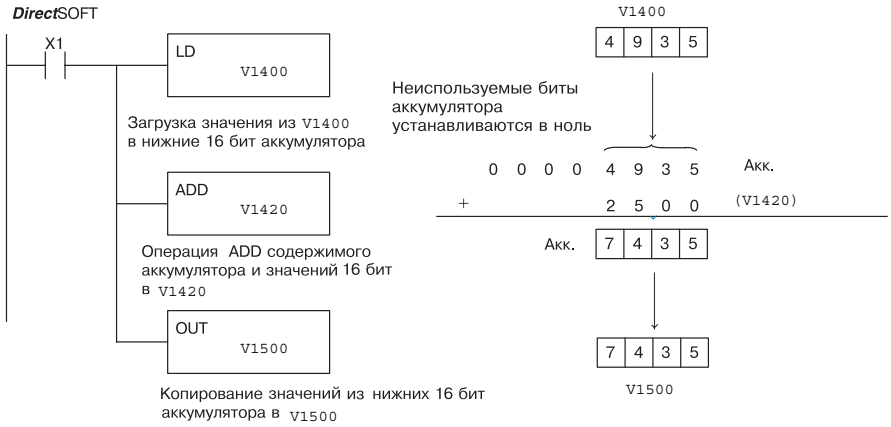


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP66	Включен, когда 16-битовая команда сложения приводит к переносу
SP67	Включен, когда 32-битовая команда сложения приводит к переносу
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75	Включен, когда выполняется двоично-десятичная команда не с двоично-десятичным числом

ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 будет загружено в аккумулятор командой Load. Значение в младших 16 битах аккумулятора складывается со значением в V1420 командой Add. Значение в аккумуляторе копируется в V1500 командой Out.



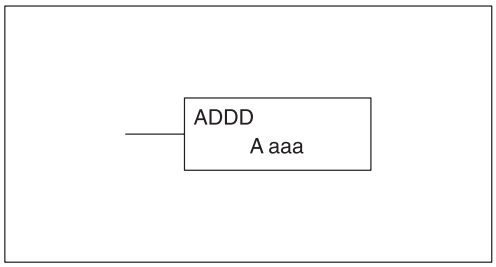
Набор на ручном программаторе

STR	X(IN)	1	↵			
LD	V	1	4	0	0	↵
ADD	V	1	4	2	0	↵
OUT	V	1	5	0	0	↵

Add Double (ADDD)

√	√	√
430	440	450

Add Double - 32-битовая команда, которая складывает двоично-десятичное значение в аккумуляторе с двоично-десятичным значением (Aaaa), которое является двумя последовательными ячейками V-памяти или двоично-десятичной константой (8 знаков максимум). Результат находится в аккумуляторе.

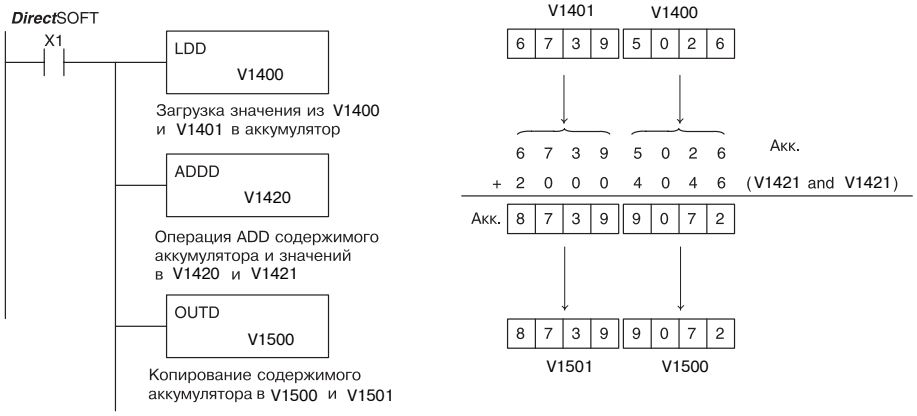


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	0-99999999	0-99999999	0-99999999

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP66	Включен, когда 16-битовая команда сложения приводит к переносу
SP67	Включен, когда 32-битовая команда сложения приводит к переносу
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75	Включен, когда выполняется двоично-десятичная команда не с двоично-десятичным числом

ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Значение в аккумуляторе складывается со значением в V1420 и V1421 командой Add Double. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.



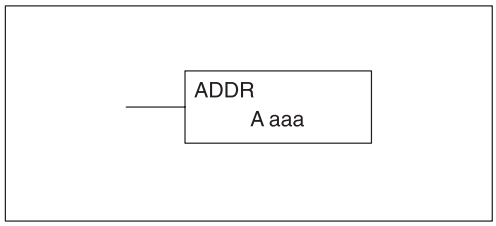
Набор на ручном программаторе

STR	X(IN)	1	⟵						
LD	SHFT	D	SHFT	V	1	4	0	0	⟵
ADD	SHFT	D	SHFT	V	1	4	2	0	⟵
OUT	SHFT	D	SHFT	V	1	5	0	0	⟵

Add Real (ADDR)

X	X	√
430	440	450

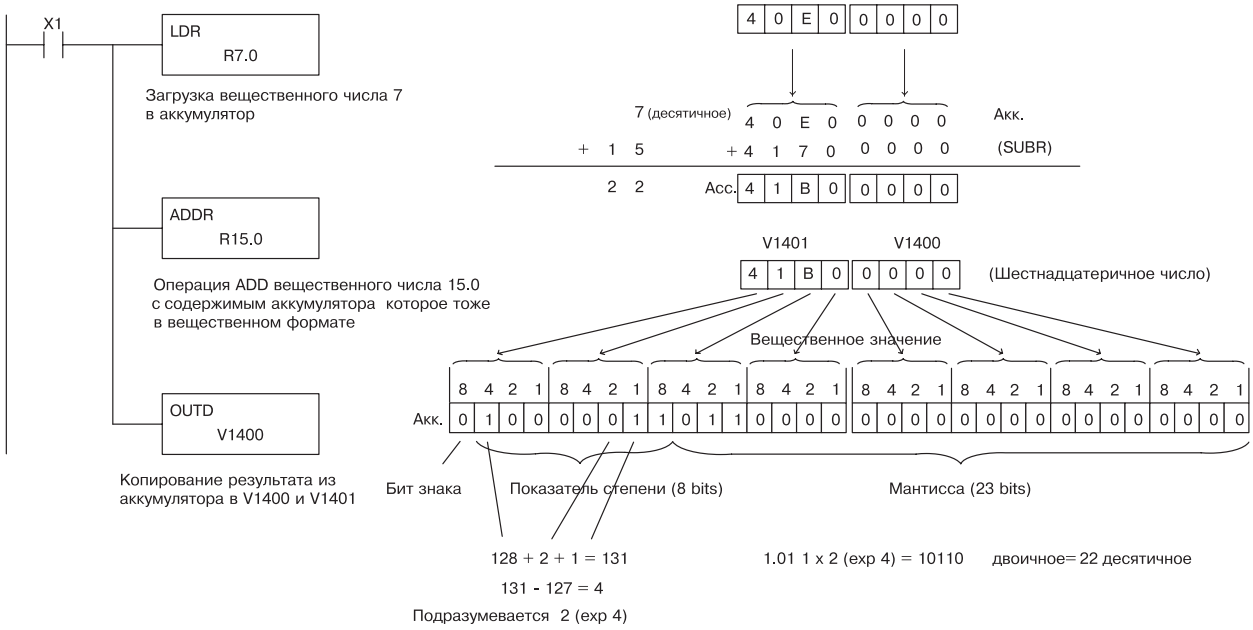
Команда Add Real складывает вещественное число в аккумуляторе с вещественной константой или с вещественным числом, занимающим две последовательные ячейки V-памяти. Результат находится в аккумуляторе. Оба числа должны соответствовать формату IEEE с плавающей точкой.



Тип данных операнда		Диапазон DL450
	A	**aaa**
V-память	V	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-42)
Константа	K	От -3.402823E+038 до + 3.402823E+038

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP71	Включен в любое время, когда V-память, определенная как указатель (P) не действительна
SP72	Включен в любое время, когда значение в аккумуляторе является числом с плавающей точкой
SP73	Включен, когда команды сложения или вычитания приводят к неправильному знаковому биту
SP74	Включен в любое время, когда математическая операция с плавающей точкой приводит к исчезновению разрядов (сигнал ошибки)
SP75	Включен, когда выполняется команда для вещественного числа с невещественным числом

ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

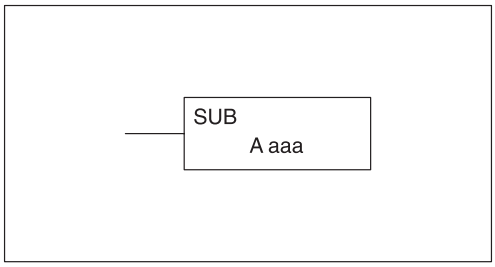


ПРИМЕЧАНИЕ. Ручной программатор не поддерживает ввод вещественных чисел с автоматическим преобразованием в 32-битовый IEEE формат. Вы должны использовать для этого DirectSOFT.

Subtract (SUB)

√	√	√
430	440	450

Subtract - 16-битовая команда, которая вычитает BCD значение (Aaaa) в ячейке V-памяти из BCD значения в младших 16 битах аккумулятора. Результат находится в аккумуляторе.

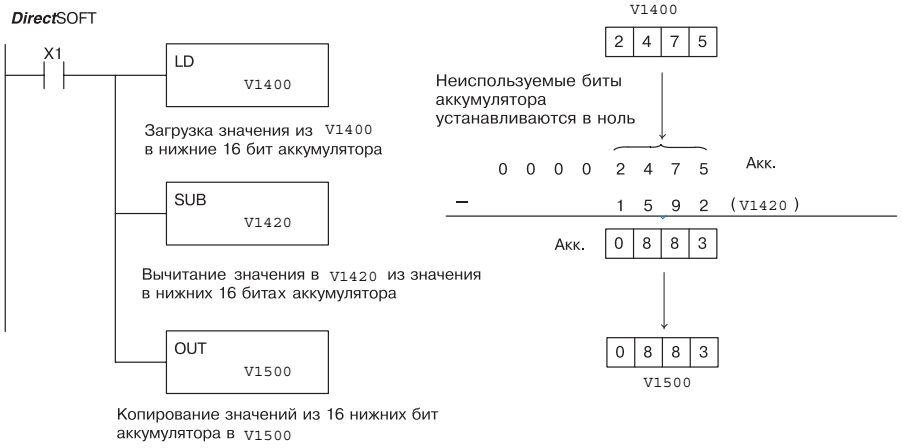


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	0-99999999	0-99999999	0-99999999

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP64	Включен, когда 16-битовая команда вычитания приводит к заимствованию
SP65	Включен, когда 32-битовая команда вычитания приводит к заимствованию
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75	Включен, когда выполняется двоично-десятичная команда не с двоично-десятичным числом

ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 будет загружено в аккумулятор командой Load. Значение в V1420 вычитается из значения в аккумуляторе командой Subtract. Значение в аккумуляторе копируется в V1500 командой Out.



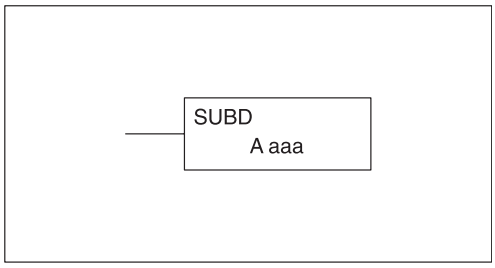
Набор на ручном программаторе

STR	X(IN)	1	↵			
LD	V	1	4	0	0	↵
SUB	V	1	4	2	0	↵
OUT	V	1	5	0	0	↵

Subtract Double (SUBD)

√	√	√
430	440	450

Subtract Double - 32-битовая команда, которая вычитает двоично-десятичное значение (Aaaa), которое является двумя последовательными ячейками V-памяти или константой (8 знаков максимум), из двоично-десятичного значения в аккумуляторе. Результат находится в аккумуляторе.

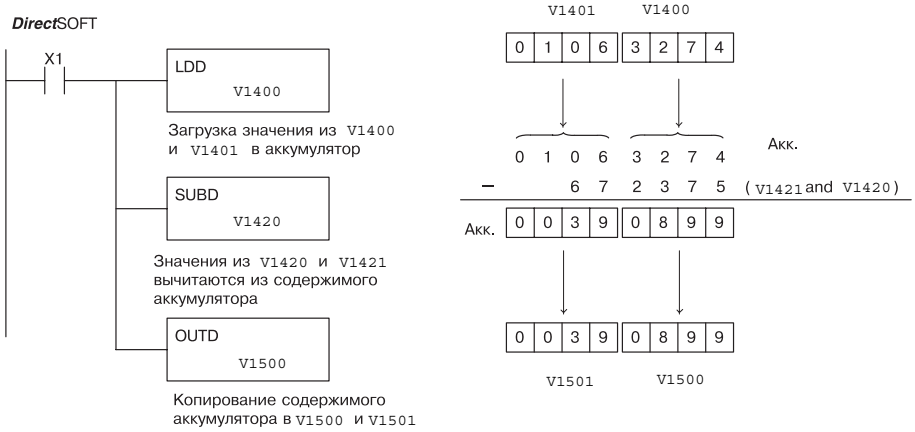


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	0-99999999	0-99999999	0-99999999

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP64	Включен, когда 16-битовая команда вычитания приводит к заимствованию
SP65	Включен, когда 32-битовая команда вычитания приводит к заимствованию
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75	Включен, когда выполняется двоично-десятичная команда не с двоично-десятичным числом

ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Значение в V1420 и V1421 вычитается из значения в аккумуляторе. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.



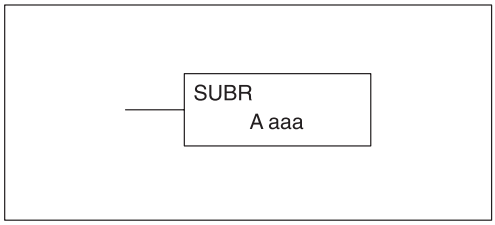
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SUB	SHFT	D	SHFT	V	1	4	2	0	↵
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Subtract Real (SUBR)

X	X	√
430	440	450

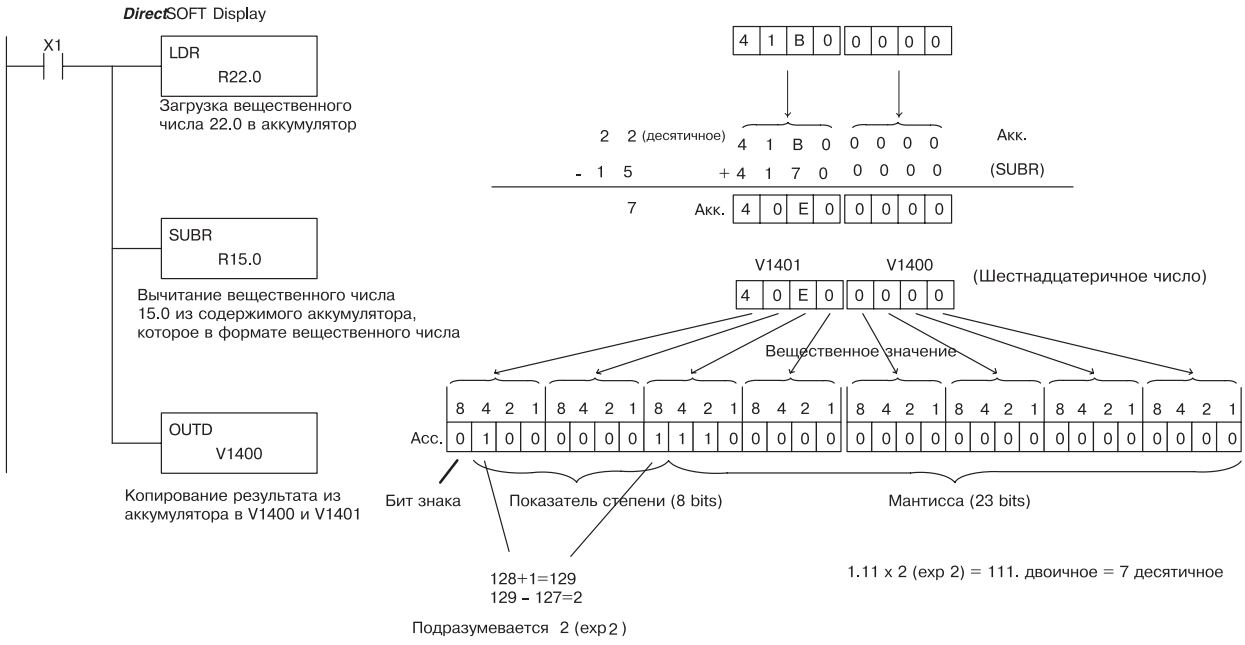
Команда Subtract Real вычитает вещественное число в аккумуляторе из вещественной константы или из вещественного числа, занимающего две последовательные ячейки V-памяти. Результат находится в аккумуляторе. Оба числа должны соответствовать формату IEEE с плавающей точкой.



Тип данных операнда		Диапазон DL450
	A	**aaa**
V-память	V	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-42)
Константа	K	От -3.402823E+038 до + 3.402823E+038

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP71	Включен в любое время, когда V-память, определенная как указатель (P) не действительна
SP72	Включен в любое время, когда значение в аккумуляторе является числом с плавающей точкой
SP73	Включен, когда команды сложения или вычитания приводят к неправильному знаковому биту
SP74	Включен в любое время, когда математическая операция с плавающей точкой приводит к исчезновению разрядов (сигнал ошибки)
SP75	Включен, когда выполняется команда для вещественного числа с невещественным числом

ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

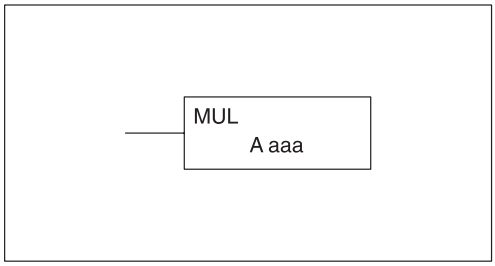


ПРИМЕЧАНИЕ. Ручной программатор не поддерживает ввод вещественных чисел с автоматическим преобразованием в 32-битовый IEEE формат. Вы должны использовать для этого DirectSOFT.

Multiply (MUL)

√	√	√
430	440	450

Multiply - 16-битовая команда, которая умножает двоично-десятичное значение (Аааа), которое является или ячейкой V-памяти или константой (4 знака максимум), на двоично-десятичное значение в младших 16 битах аккумулятора. Результат может иметь до 8 знаков и находится в аккумуляторе.

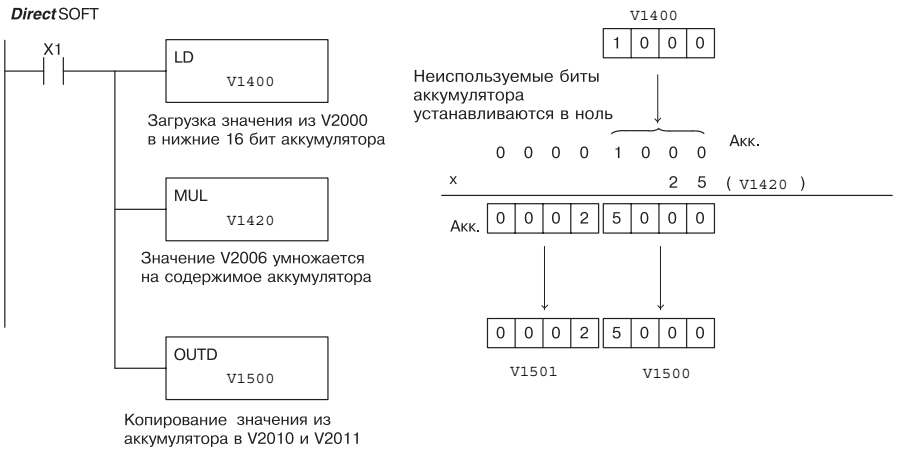


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	0-9999	0-9999	0-9999

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75	Включен, когда выполняется двоично-десятичная команда с числом не в двоично-десятичном формате

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 будет загружено в аккумулятор командой Load. Значение в V1420 умножается на значение в аккумуляторе. Значение аккумулятора копируется в V1500 и V1501 командой Out Double.



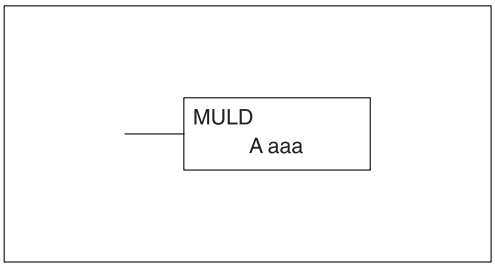
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	V	1	4	0	0	↵			
MUL	V	1	4	2	0	↵			
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Multiply Double (MULD)

X	√	√
430	440	450

Multiply Double - 32-битовая команда, которая умножает 8-значное двоично-десятичное значение в аккумуляторе на 8-значное двоично-десятичное значение в двух последовательных ячейках V-памяти, указанных в команде. Младшие 8 знаков результата находятся в аккумуляторе. Верхние знаки результата хранятся в стеке аккумулятора.

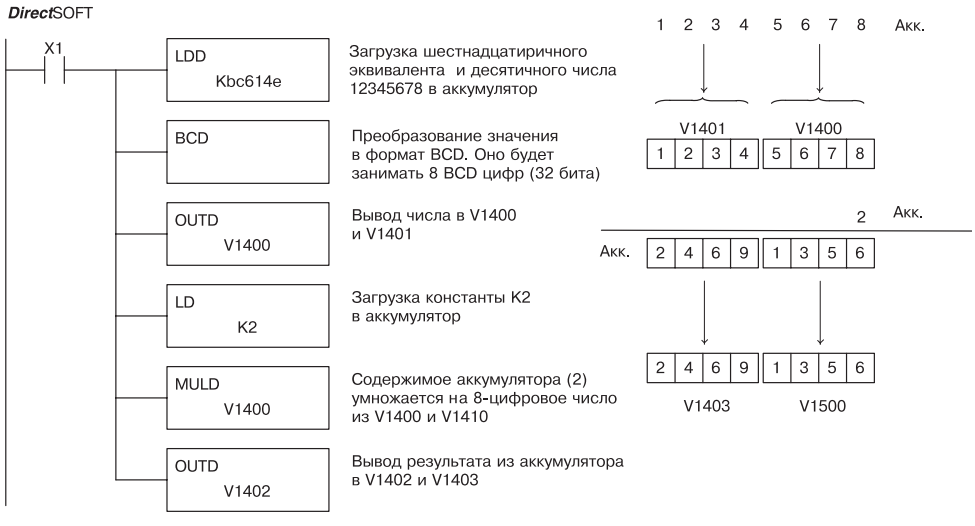


Тип данных операнда		Диапазон DL450
	A	**aaa**
V-память	V	Все (см. стр. 3-42)
Указатель	P	—

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75	Включен, когда выполняется двоично-десятичная команда с числом не в двоично-десятичном формате

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, шестнадцатиричная константа Kbc614e будет загружена в аккумулятор. Это число, преобразованное в двоично-десятичный формат, - ”12345678”. Эти числа хранятся в V1400 и V1401. После загрузки константы K2 в аккумулятор, мы умножаем ее на 12345678 и получаем 24691356.



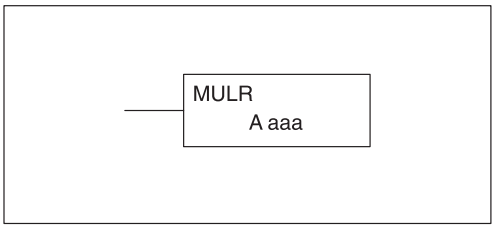
Набор на ручном программаторе

STR	X(IN)	1	↵	LD	SHFT	D	SHFT			
K(CON)	SHFT	B(H)	C(H)	6	1	4	E(H)	↵	BCD	↵
OUT	SHFT	D	SHFT	V	1	4	0	0	↵	
LD	K(CON)	2	↵	MUL	SHFT	D				
V	1	4	0	0	↵					

Multiply Real (MULR)

X	X	√
430	440	450

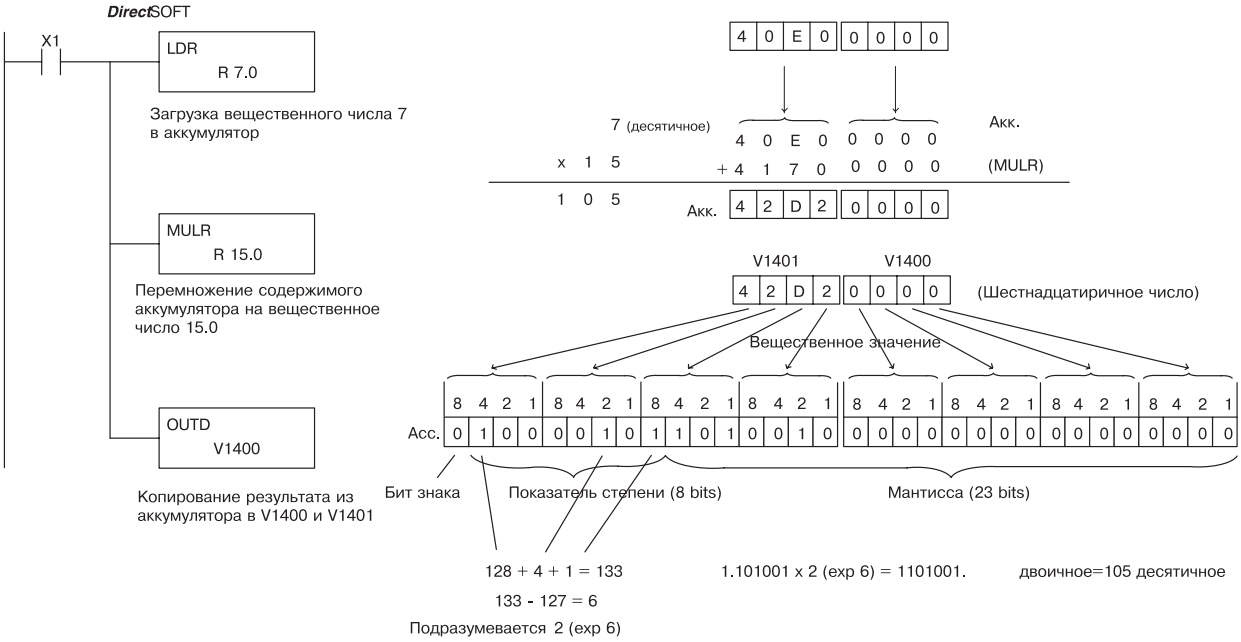
Команда Multiply Real умножает вещественное число в аккумуляторе на вещественную константу или на вещественное число, занимающее две последовательные ячейки V-памяти. Результат находится в аккумуляторе. Оба числа должны соответствовать формату IEEE с плавающей точкой.



Тип данных операнда		Диапазон DL450
	A	**aaa**
V-память	V	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-42)
Константа	K	От -3.402823E+038 до + 3.402823E+038

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP71	Включен в любое время, когда V-память, определенная как указатель (P) не действительна
SP72	Включен в любое время, когда значение в аккумуляторе является числом с плавающей точкой
SP73	Включен, когда команды сложения или вычитания приводят к неправильному знаковому биту
SP74	Включен в любое время, когда математическая операция с плавающей точкой приводит к исчезновению разрядов (сигнал ошибки)
SP75	Включен, когда выполняется команда для вещественного числа с невещественным числом

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

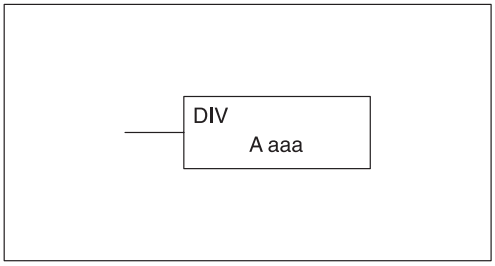


ПРИМЕЧАНИЕ: Ручной программатор не поддерживает ввод вещественных чисел с автоматическим преобразованием в 32-битовый IEEE формат. Вы должны использовать DirectSOFT для этого.

Divide (DIV)

√	√	√
430	440	450

Divide - 16-битовая команда, которая делит двоично-десятичное значение в аккумуляторе на двоично-десятичное значение (Aaaa), которое является или ячейкой V-памяти или константой (4 знака максимум). Первая часть результата находится в аккумуляторе, а остаток - в первой ячейке стека.

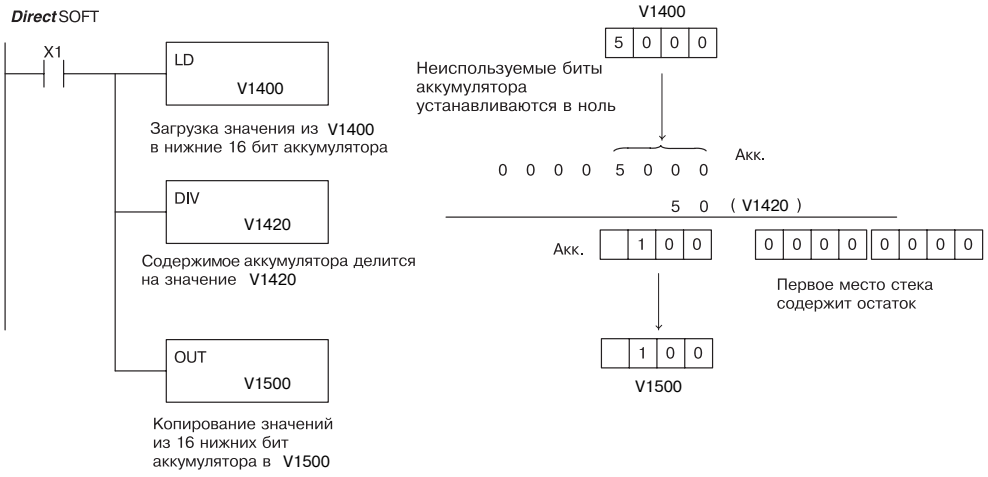


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	0-9999	0-9999	0-9999

Флаги дискретных разрядов	Описание
SP53	Включен, когда значение операнда больше, чем значение, с которым может работать аккумулятор
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75	Включен, когда выполняется двоично-десятичная команда с числом не в двоично-десятичном формате

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 будет загружено в аккумулятор командой Load. Значение в аккумуляторе будет поделено на значение в V1420 командой Divide. Значение в аккумуляторе копируется в V1500 командой Out.



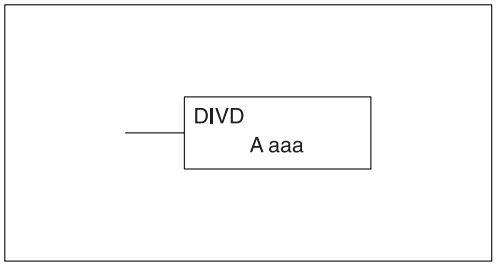
Набор на ручном программаторе

STR	X(IN)	1	↵			
LD	V	1	4	0	0	↵
DIV	V	1	4	2	0	↵
OUT	V	1	5	0	0	↵

Divide Double (DIVD)

X	√	√
430	440	450

Divide Double - 32-битовая команда, которая делит двоично-десятичное значение в аккумуляторе на двоично-десятичное значение (Aaaa), которое должно быть взято из двух последовательных ячеек V-памяти. (Вы не можете использовать константу как параметр в команде). Первая часть результата находится в аккумуляторе, а остаток - в первой ячейке стека.

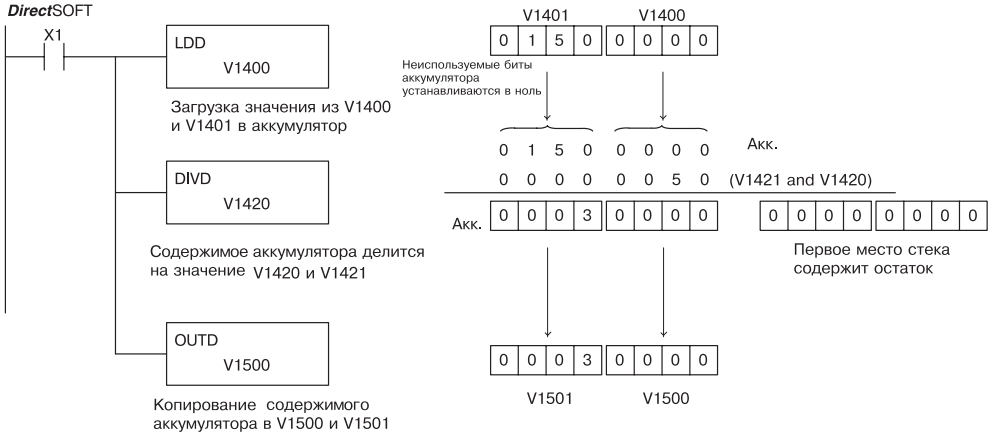


Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-41)	Все (см. стр. 3-42)

Флаги дискретных разрядов	Описание
SP53	Включен, когда значение операнда больше, чем значение, с которым может работать аккумулятор
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75	Включен, когда выполняется двоично-десятичная команда с числом не в двоично-десятичном формате

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Значение в аккумуляторе делится на значение в V1420 и V1421 командой Divide Double. Первая часть результата находится в аккумуляторе, а остаток находится в первой ячейке стека. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.



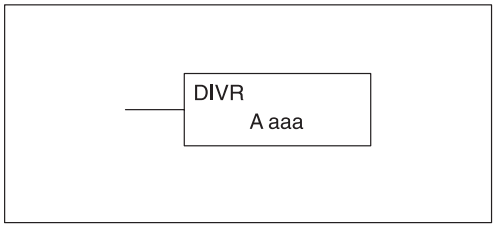
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
DIV	SHFT	D	SHFT	V	1	4	2	0	↵
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Divide Real (DIVR)

X	X	√
430	440	450

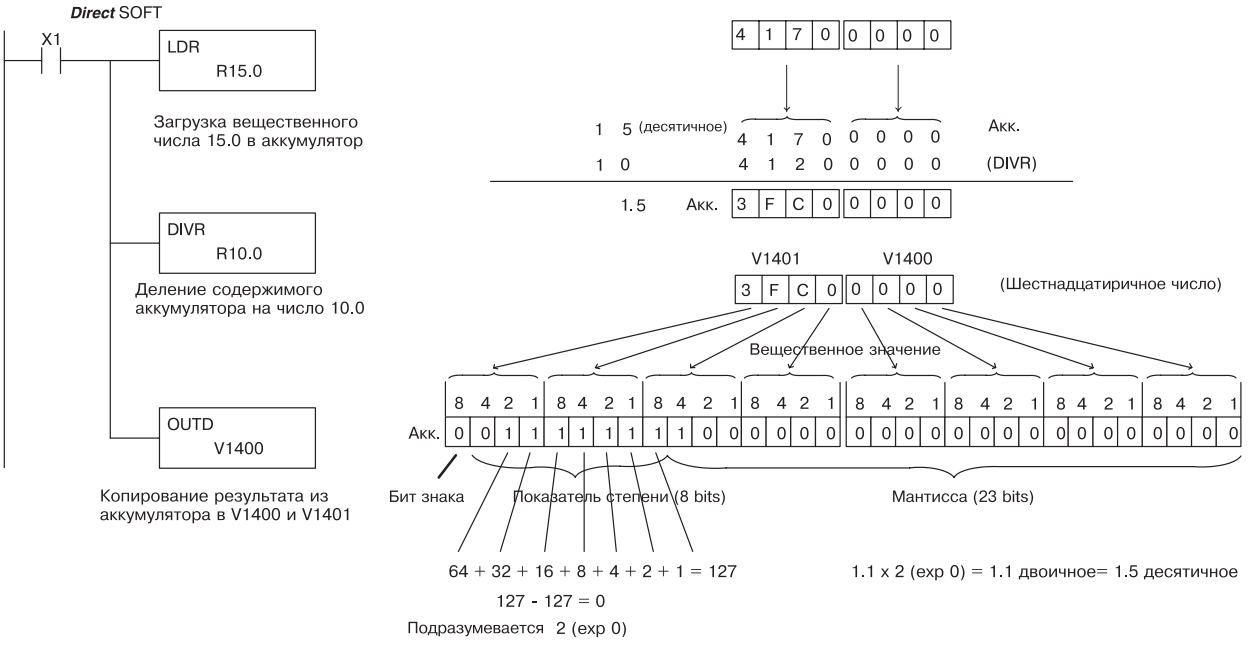
Команда Divide Real делит вещественное число в аккумуляторе на вещественную константу или на вещественное число, занимающее две последовательные ячейки V-памяти. Результат находится в аккумуляторе. Оба числа должны соответствовать формату IEEE с плавающей точкой.



Тип данных операнда		Диапазон DL450
	A	**aaa**
V-память	V	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-42)
Константа	K	От -3.402823E+038 до + 3.402823E+038

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP71	Включен в любое время, когда V-память, определенная как указатель (P) не действительна
SP72	Включен в любое время, когда значение в аккумуляторе является числом с плавающей точкой
SP73	Включен, когда команды сложения или вычитания приводят к неправильному знаковому биту
SP74	Включен в любое время, когда математическая операция с плавающей точкой приводит к исчезновению разрядов (сигнал ошибки)
SP75	Включен, когда выполняется команда для вещественного числа с невещественным числом

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

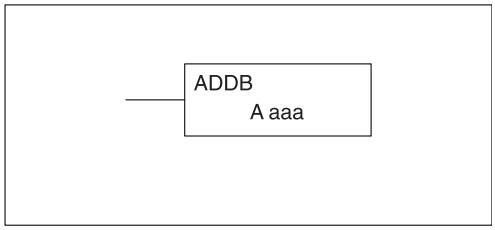


ПРИМЕЧАНИЕ: Ручной программатор не поддерживает ввод вещественных чисел с автоматическим преобразованием в 32-битовый IEEE формат. Вы для этого должны использовать DirectSOFT.

Add Binary (ADDB)

√	√	√
430	440	450

Add Binary - 16-битовая команда, которая складывает двоичное значение в формате дополнения до 2-х без знака в младших 16 битах аккумулятора с двоичным значением в формате дополнения до 2-х без знака (Аааа), которое является или ячейкой V-памяти или 16-битовой константой. Результат может быть до 32 бит (в формате дополнения до 2-х без знака) и остаетсяя в аккумуляторе.

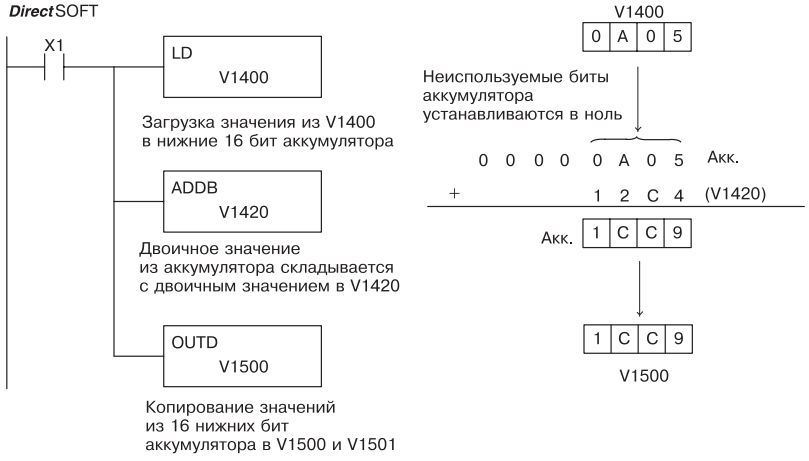


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-40)	Вся V-память (см. стр. 3-41)	Вся V-память (см. стр. 3-42)
Константа	K	0-FFFF	0-FFFF	0-FFFF

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP66	Включен, когда 16-битовая команда сложения приводит к переносу
SP67	Включен, когда 32-битовая команда сложения приводит к переносу
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP73	Включен, когда команды сложения или вычитания приводят к неправильному знаковому биту

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 будет загружено в аккумулятор командой Load. Двоичное значение в аккумуляторе будет сложено с двоичным значением в V1420 командой Add Binary.



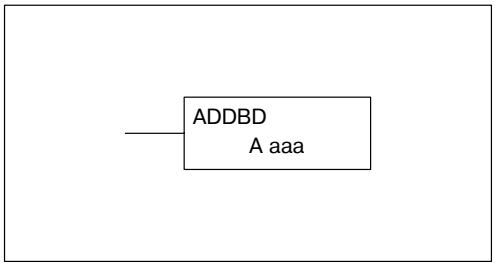
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	V	1	4	0	0	↵			
ADD	SHFT	B	SHFT	V	1	4	2	0	↵
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Add Binary Double (ADDBD)

X	√	√
430	440	450

Add Binary Double является 32-битовой командой, которая складывает двоичное значение в формате дополнения до 2-х без знака в аккумуляторе с двоичным значением в формате дополнения до 2-х без знака (Аааа), которое является двумя последовательными ячейками V-памяти или 32-битовой константой в формате дополнения до 2-х без знака. Результат остается в аккумуляторе.

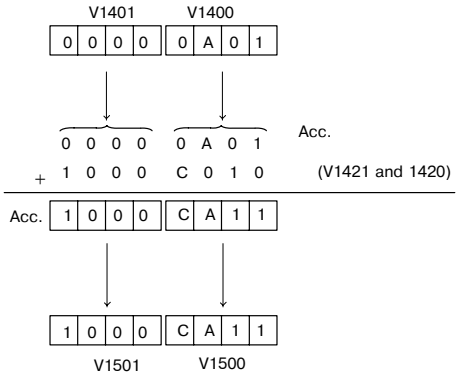


Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-41)	Вся V-память (см. стр. 3-42)
Константа	K	0-FFFFFFFF	0-FFFFFFFF

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP66	Включен, когда 16-битовая команда сложения приводит к переносу
SP67	Включен, когда 32-битовая команда сложения приводит к переносу
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP73	Включен, когда команды сложения или вычитания приводят к неправильному знаковому биту

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Двоичное значение в аккумуляторе будет сложено с двоичным значением в V1420 и V1421 командой Add Binary Double. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.



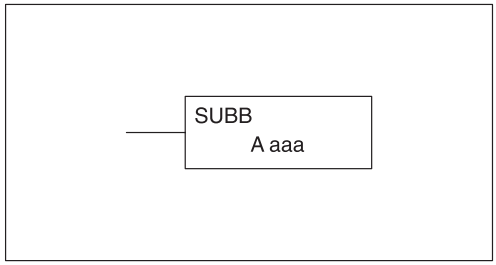
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
ADD	SHFT	B	D	SHFT	V	1	4	2	0	↵
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Subtract Binary (SUBB)

√	√	√
430	440	450

Subtract Binary - 16-битовая команда, которая вычитает двоичное значение в формате дополнения до 2-х без знака (Аааа), которое является или ячейкой V-памяти или 16-битовым двоичным значением в формате дополнения до 2-х, из двоичного значения в аккумуляторе. Результат находится в аккумуляторе.

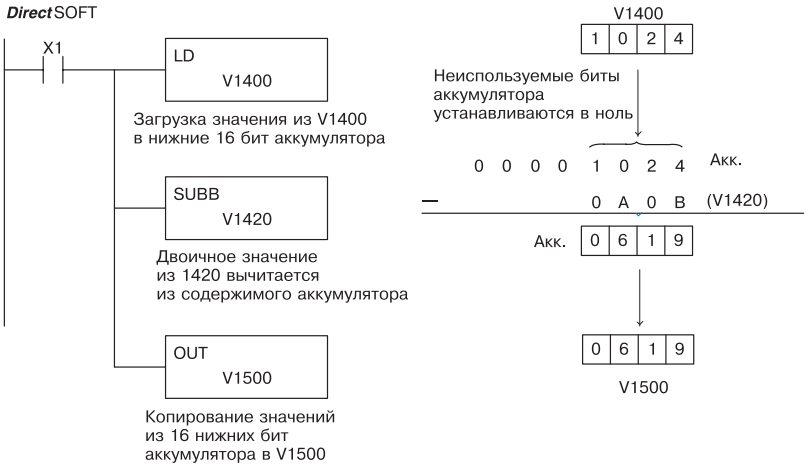


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	0-FFFF	0-FFFF	0-FFFF

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP64	Включен, когда 16-битовая команда вычитания приводит к заимствованию
SP65	Включен, когда 32-битовая команда вычитания приводит к заимствованию
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 будет загружено в аккумулятор командой Load. Двоичное значение в V1420 вычитается из двоичного значения в аккумуляторе командой Subtract Binary. Значение в аккумуляторе копируется в V1500 командой Out.



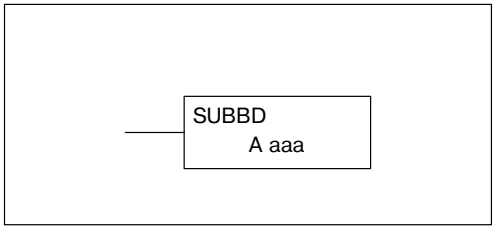
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	V	1	4	0	0	↵			
SUB	SHFT	B	SHFT	V	1	4	2	0	↵
OUT	V	1	5	0	0	↵			

Subtract Binary Double (SUBBD)

X	√	√
430	440	450

Subtract Binary Double является 32-битовой командой, которая вычитает двоичное значение в формате дополнения до 2 без знака (Aaaa), которым могут быть или две последовательные ячейки V-памяти или 32-битовая константа в формате дополнения до 2 без знака, из двоичного значения в аккумуляторе. Результат остается в аккумуляторе.

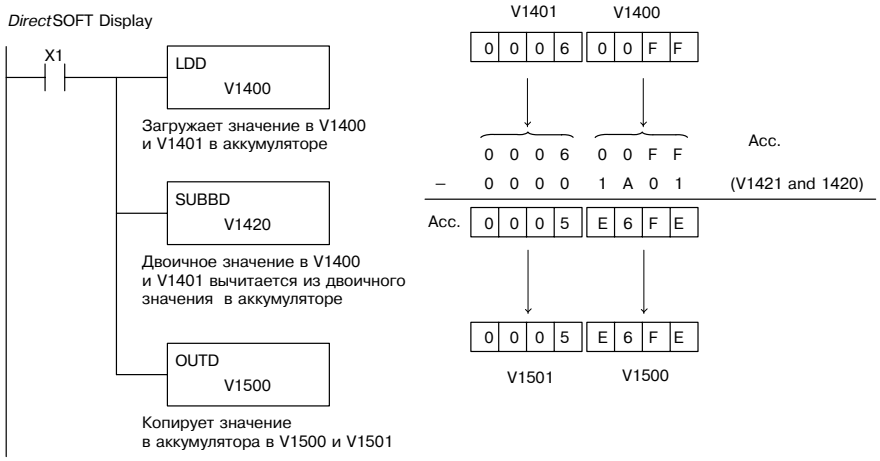


Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	0-FFFFFFFF	0-FFFFFFFF

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP64	Включен, когда 16-битовая команда вычитания приводит к заимствованию
SP65	Включен, когда 32-битовая команда вычитания приводит к заимствованию
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Двоичное значение в V1400 и V1401 вычитается из двоичного значения в аккумуляторе командой Subtract Binary Double. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.



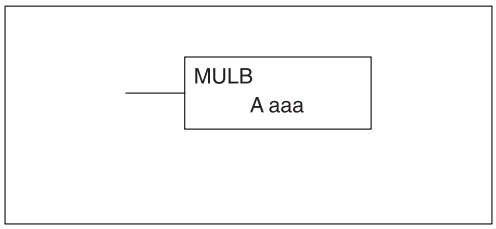
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SUB	SHFT	B	D	SHFT	V	1	4	2	0	↵
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Multiply Binary (MULB)

√	√	√
430	440	450

Multiply Binary - 16-битовая команда, которая умножает двоичное значение в формате дополнения до 2-х без знака (Aaaa), которым могут быть или ячейка V-памяти или 16-битовая двоичная константа в формате дополнения до 2-х без знака, на 16-битовое двоичное значение в аккумуляторе. Результат может быть до 32 бит и остается в аккумуляторе.

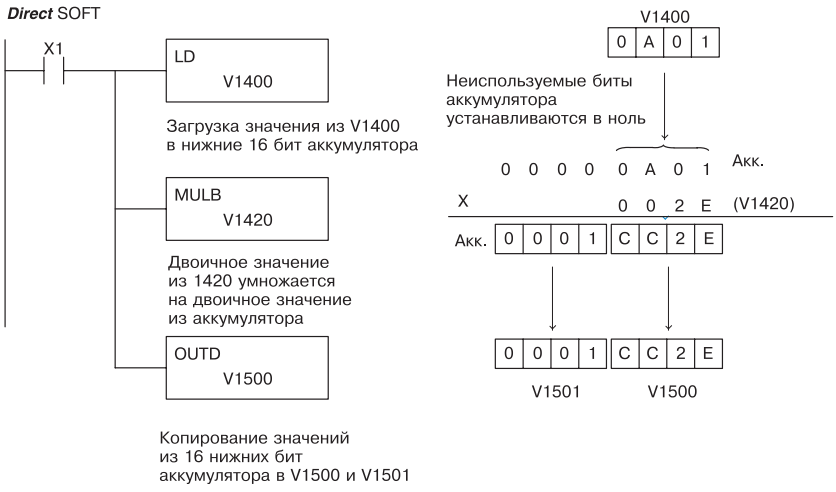


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	0-FFFF	0-FFFF	0-FFFF

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 будет загружено в аккумулятор командой Load. Двоичное значение в V1420 умножается на двоичное значение в аккумуляторе командой Multiply Binary. Значение в аккумуляторе копируется в V1500 и V1501 командой Out.



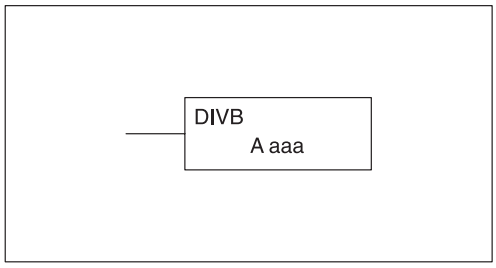
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	V	1	4	0	0	↵			
MUL	SHFT	B	SHFT	V	1	4	2	0	↵
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Divide Binary (DIVB)

√	√	√
430	440	450

Divide Binary - 16-битовая команда, которая делит двоичное значение в формате дополнения до 2-х без знака, находящееся в аккумуляторе, на двоичное значение (Aaaa), которым может быть ячейка V-памяти или 16-битовая двоичная константа в формате дополнения до 2-х без знака. Первая часть частного от деления находится в аккумуляторе, а остальная часть - в первой ячейке стека.

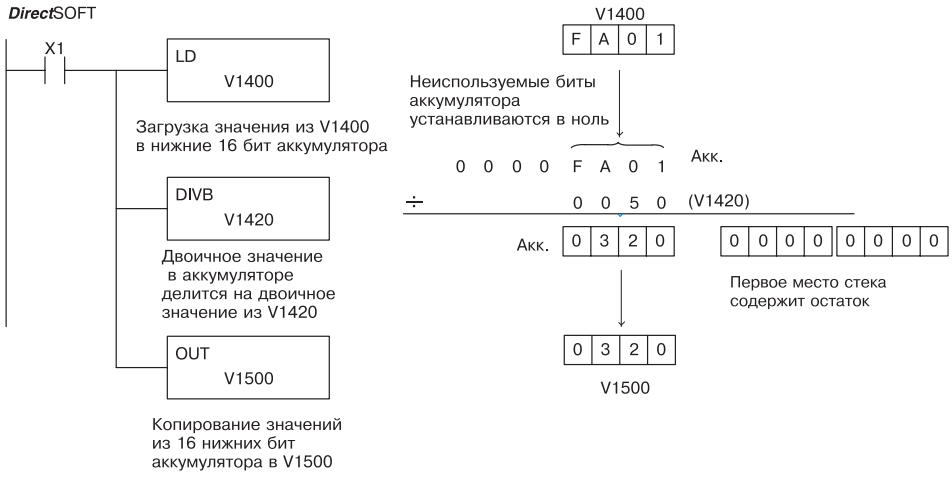


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	0-FFFF	0-FFFF	0-FFFF

Флаги дискретных разрядов	Описание
SP53	Включен, когда значение операнда больше, чем значение, с которым может работать аккумулятор
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 будет загружено в аккумулятор командой Load. Двоичное значение в аккумуляторе делится на двоичное значение в V1420 командой Divide Binary. Значение в аккумуляторе копируется в V1500 командой Out.



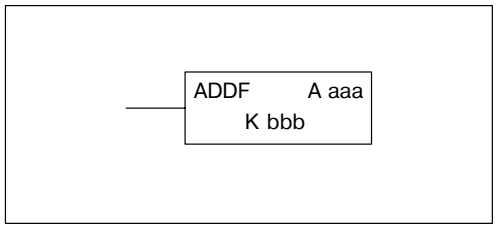
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	V	1	4	0	0	↵			
DIV	SHFT	B	SHFT	V	1	4	2	0	↵
OUT	V	1	5	0	0	↵			

Add Formatted (ADDF)

X	√	√
430	440	450

Add Formatted является 32-битовой командой, которая складывает двоично-десятичное значение в аккумуляторе с двоично-десятичным значением (Aaaa), которым является диапазон дискретных битов. Заданным диапазоном (Kbbb) могут быть последовательные биты с 1 по 32. Результат сохраняется в аккумуляторе.



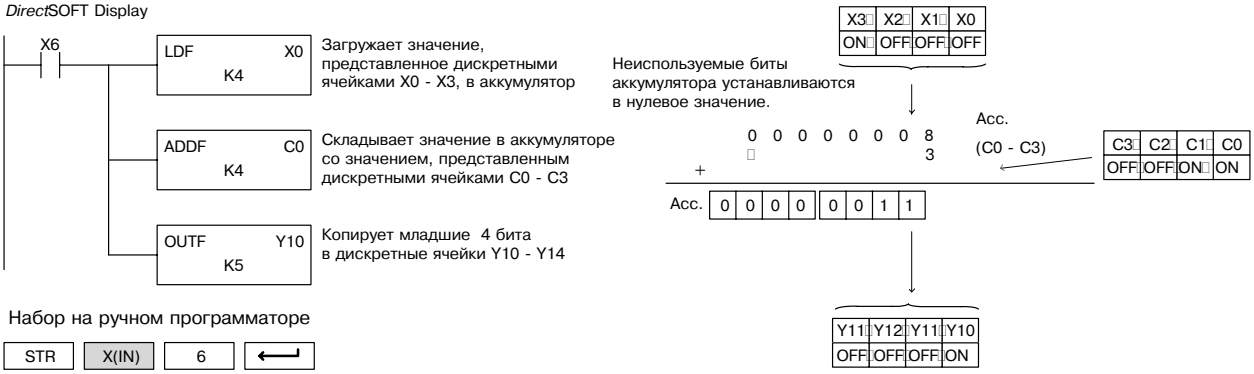
Тип данных операнда		Диапазон DL440		Диапазон DL450	
	A/B	**aaa**	**bbb**	**aaa**	**bbb**
Входы	X	0-477	--	0-1777	--
Выходы	Y	0-477	--	0-1777	--
Реле управления	C	0-1777	--	0-3777	--
Биты стадии	S	0-1777	--	0-1777	--
Биты таймера	T	0-377	--	0-377	--
Биты счетчика	CT	0-177	--	0-377	--
Специальные реле	SP	0-137 320-717	--	0-137 320-717	--
Глобальный ввод/вывод	GX	0-1777	--	0-2777	--
Константа	K	--	1-32	--	1-32

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP66	Включен, когда 16-битовая команда сложения приводит к переносу
SP67	Включен, когда 32-битовая команда сложения приводит к переносу
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75	Включен, когда выполняется двоично-десятичная команда не с двоично-десятичным числом

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X6 включен, сформированное дискретными ячейками X0 - X3 значение загружается командой Load Formatted в аккумулятор. Значение, сформированное дискретными ячейками C0 - C3, складывается со значением в аккумуляторе с использованием команды Add Formatted. Значение в младших четырех битах копируется в Y10 - Y13 командой Out Formatted.

*Direct*SOFT Display



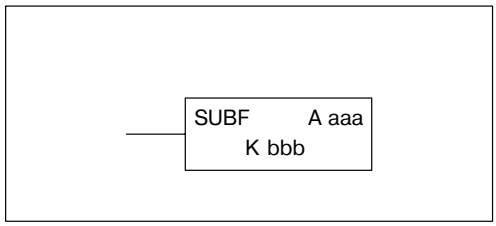
Набор на ручном программаторе

STR	X(IN)	6	↵					
LD	SHFT	F	SHFT	X(IN)	0	K(CON)	4	↵
ADD	SHFT	F	SHFT	C(CR)	0	K(CON)	4	↵
OUT	SHFT	F	SHFT	Y(OUT)	1	0	K(CON)	4	↵

Subtract Formatted (SUBF)

X	√	√
430	440	450

Subtract Formatted является 32-битовой командой, которая вычитает двоично-десятичное значение (Aaaa), которым является диапазон дискретных битов, из двоично-десятичного значения в аккумуляторе. Заданным диапазоном (Kbbb) может быть последовательные биты с 1 по 32. Результат сохраняется в аккумуляторе.



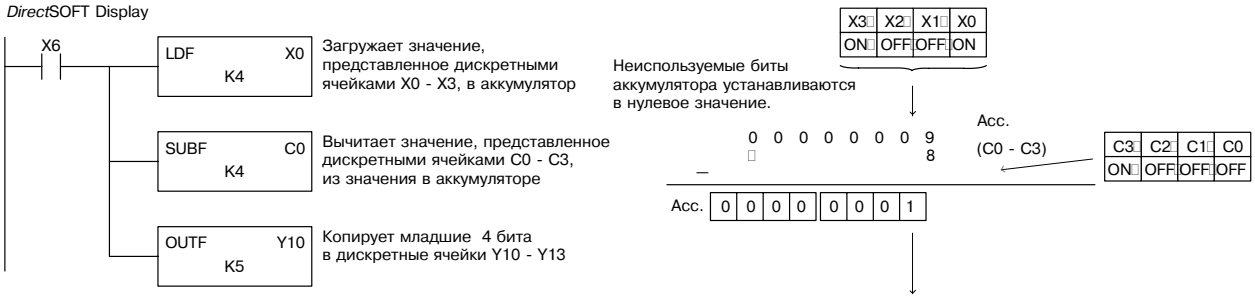
Тип данных операнда		Диапазон DL440		Диапазон DL450	
	A/B	**aaa**	**bbb**	**aaa**	**bbb**
Входы	X	0-477	--	0-1777	--
Выходы	Y	0-477	--	0-1777	--
Реле управления	C	0-1777	--	0-3777	--
Биты стадии	S	0-1777	--	0-1777	--
Биты таймера	T	0-377	--	0-377	--
Биты счетчика	CT	0-177	--	0-377	--
Специальные реле	SP	0-137 320-717	--	0-137 320-717	--
Глобальный ввод/вывод	GX	0-1777	--	0-2777	--
Константа	K	--	1-32	--	1-32

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP64	Включен, когда 16-битовая команда вычитания приводит к заимствованию
SP65	Включен, когда 32-битовая команда вычитания приводит к заимствованию
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75	Включен, когда выполняется двоично-десятичная команда не с двоично-десятичным числом

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X6 включен, сформированное дискретными ячейками X0 - X3 значение, загружается командой Load Formatted в аккумулятор. Значение, сформированное дискретными ячейками C0 - C3, вычитается из значения в аккумуляторе с использованием команды Subtract Formatted. Значение в младших четырех битах копируется в Y10 - Y13 командой Out Formatted.

*Direct*SOFT Display



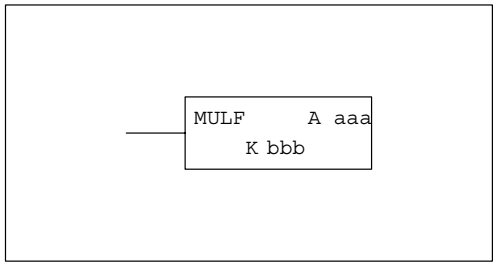
Набор на ручном программаторе

STR	X(IN)	6	↵					
LD	SHFT	F	SHFT	X(IN)	0	K(CON)	4	↵
SUBF	SHFT	F	SHFT	C(CR)	0	K(CON)	4	↵
OUT	SHFT	F	SHFT	Y(OUT)	1	0	K(CON)	4	↵

Multiply Formatted (SUBF)

X	√	√
430	440	450

Multiply Formatted является 16-битовой командой, которая умножает двоично-десятичное значение в аккумуляторе на двоично-десятичное значение (Aaaa), которым является диапазон дискретных битов. Заданным диапазоном (Kbbb) может быть последовательные биты с 1 по 32. Результат сохраняется в аккумуляторе.



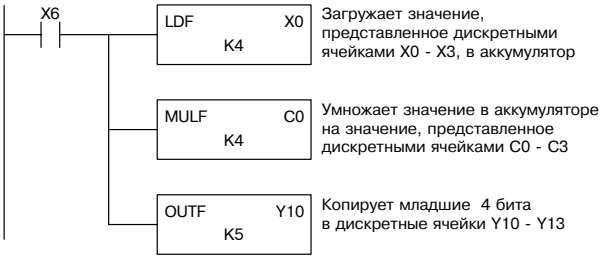
Тип данных операнда		Диапазон DL440		Диапазон DL450	
	A/B	**aaa**	**bbb**	**aaa**	**bbb**
Входы	X	0-477	--	0-1777	--
Выходы	Y	0-477	--	0-1777	--
Реле управления	C	0-1777	--	0-3777	--
Биты стадии	S	0-1777	--	0-1777	--
Биты таймера	T	0-377	--	0-377	--
Биты счетчика	CT	0-177	--	0-377	--
Специальные реле	SP	0-137 320-717	--	0-137 320-717	--
Глобальный ввод/вывод	GX	0-1777	--	0-2777	--
Константа	K	--	1-16	--	1-16

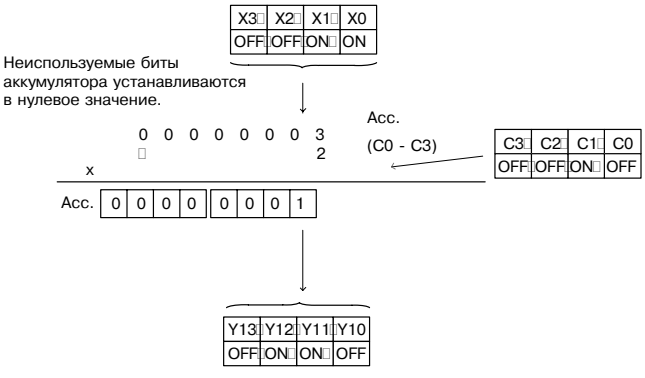
Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75	Включен, когда выполняется двоично-десятичная команда с числом не в двоично-десятичном формате

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X6 включен, сформированное дискретными ячейками X0 - X3 значение, загружается командой Load Formatted в аккумулятор. Значение, сформированное дискретными ячейками C0 - C3, умножается на значение в аккумуляторе с использованием команды Multiply Formatted. Значение в младших четырех битах копируется в Y10 - Y13 командой Out Formatted.

*Direct*SOFT Display





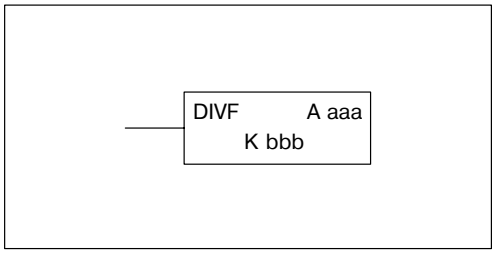
Набор на ручном программаторе

STR	X(IN)	6	↵					
LD	SHFT	F	SHFT	X(IN)	0	K(CON)	4	↵
MUL	SHFT	F	SHFT	C(CR)	0	K(CON)	4	↵
OUT	SHFT	F	SHFT	Y(OUT)	1	0	K(CON)	4	↵

Divide Formatted (DIVF)

X	√	√
430	440	450

Divide Formatted является 16-битовой командой, которая делит двоично-десятичное значение в аккумуляторе на двоично-десятичное значение (Aaaa), которым является диапазон дискретных битов. Заданным диапазоном (Kbbb) может быть последовательные биты с 1 по 16.Первая часть частного от деления сохраняется в аккумуляторе, а остаток заносится в первую ячейку стека.



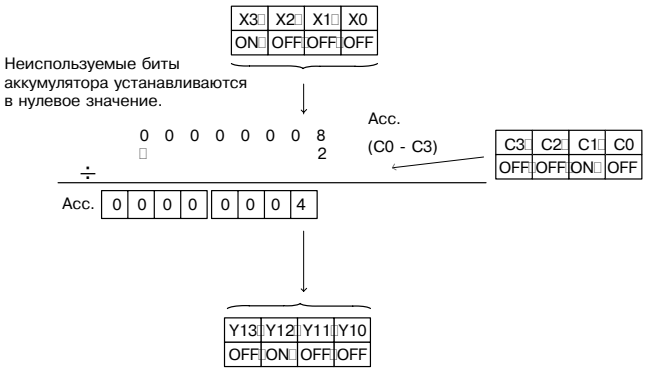
Тип данных операнда		Диапазон DL440		Диапазон DL450	
	A/B	aaa	bbb	aaa	bbb
Входы	X	0-477	--	0-1777	--
Выходы	Y	0-477	--	0-1777	--
Реле управления	C	0-1777	--	0-3777	--
Биты стадии	S	0-1777	--	0-1777	--
Биты таймера	T	0-377	--	0-377	--
Биты счетчика	CT	0-177	--	0-377	--
Специальные реле	SP	0-137 320-717	--	0-137 320-717	--
Глобальный ввод/вывод	GX	0-1777	--	0-2777	--
Константа	K	--	1-16	--	1-16

Флаги дискретных разрядов	Описание
SP53 □	Включен, когда значение операнда больше, чем значение, с которым может работать аккумулятор
SP63 □	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70 □	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75 □ □	Включен, когда выполняется двоично-десятичная команда с числом не в двоично-десятичном формате

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X6 включен, сформированное дискретными ячейками X0 - X3 значение, загружается командой Load Formatted в аккумулятор. Значение в аккумуляторе делится на значение, сформированное дискретными ячейками C0 - C3 с использованием команды Divide Formatted. Значение в младших четырех битах копируется в Y10 - Y13 командой Out Formatted.

*Direct*SOFT Display



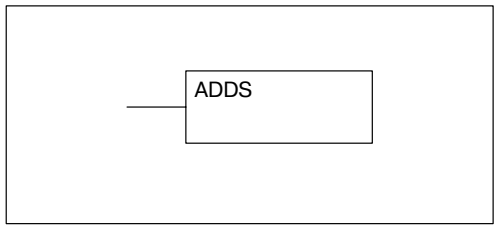
Набор на ручном программаторе

STR	X(IN)	6	↵					
LD	SHFT	F	SHFT	X(IN)	0	K(CON)	4	↵
DIV	SHFT	F	SHFT	C(CR)	0	K(CON)	4	↵
OUT	SHFT	F	SHFT	Y(OUT)	1	0	K(CON)	4	↵

Add Top of Stack (ADDS)

√	√	√
430	440	450

Add Top of Stack является 32-битовой командой, которая складывает двоично-десятичное значение в аккумуляторе с двоично-десятичным значением на первом уровне стека аккумулятора. Результат остается в аккумуляторе. Значение на первом уровне стека аккумулятора удаляется, а все значения стека сдвигаются на один уровень вверх.

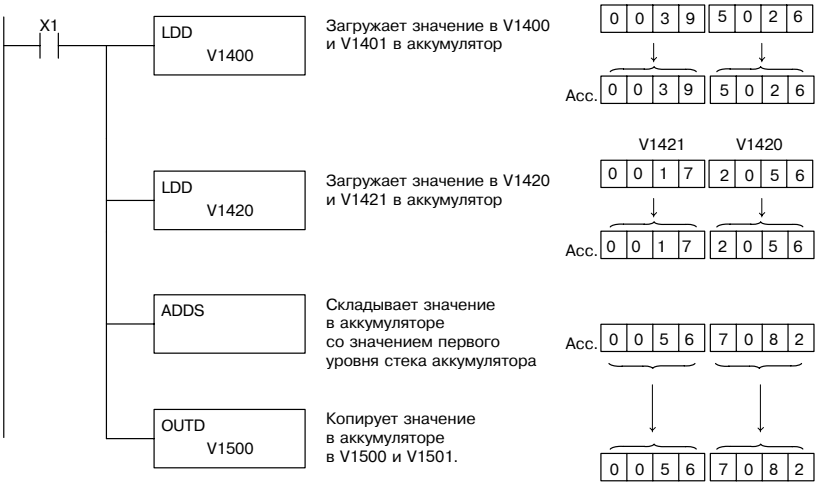


Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP66	Включен, когда 16-битовая команда сложения приводит к переносу
SP67	Включен, когда 32-битовая команда сложения приводит к переносу
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75	Включен, когда выполняется двоично-десятичная команда не с двоично-десятичным числом

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Значение в V1420 и V1421 загружается в аккумулятор командой Load Double с выталкиванием значения, ранее загруженного в аккумулятор, в стек аккумулятора. Значение первого уровня стека аккумулятора складывается со значением в аккумуляторе командой Add Stack. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.

*Direct*SOFT Display



Стек аккумулятора после первой команды LDD

Уровень 1	X	X	X	X	X	X	X	X
Уровень 2	X	X	X	X	X	X	X	X
Уровень 3	X	X	X	X	X	X	X	X
Уровень 4	X	X	X	X	X	X	X	X
Уровень 5	X	X	X	X	X	X	X	X
Уровень 6	X	X	X	X	X	X	X	X
Уровень 7	X	X	X	X	X	X	X	X
Уровень 8	X	X	X	X	X	X	X	X

Стек аккумулятора после второй команды LDD

Уровень 1	0	0	3	9	5	0	2	6
Уровень 2	X	X	X	X	X	X	X	X
Уровень 3	X	X	X	X	X	X	X	X
Уровень 4	X	X	X	X	X	X	X	X
Уровень 5	X	X	X	X	X	X	X	X
Уровень 6	X	X	X	X	X	X	X	X
Уровень 7	X	X	X	X	X	X	X	X
Уровень 8	X	X	X	X	X	X	X	X

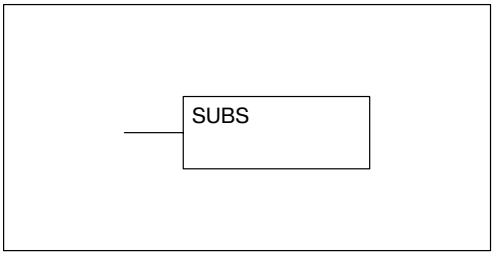
Набор на ручном программаторе

STR	X(IN)	1	⏎						
LD	SHFT	D	SHFT	V	1	4	0	0	⏎
LD	SHFT	D	SHFT	V	1	4	2	0	⏎
ADD	SHFT	S	SHFT	⏎					
OUT	SHFT	D	SHFT	V	1	5	0	0	⏎

Subtract Top of Stack (SUBS)

√	√	√
430	440	450

Subtract Top of Stack является 32-битовой командой, которая вычитает двоично-десятичное значение на первом уровне стека аккумулятора из двоично-десятичного значения в аккумуляторе. Результат остается в аккумуляторе. Значение на первом уровне стека аккумулятора удаляется, а все значения стека сдвигаются на один уровень вверх.

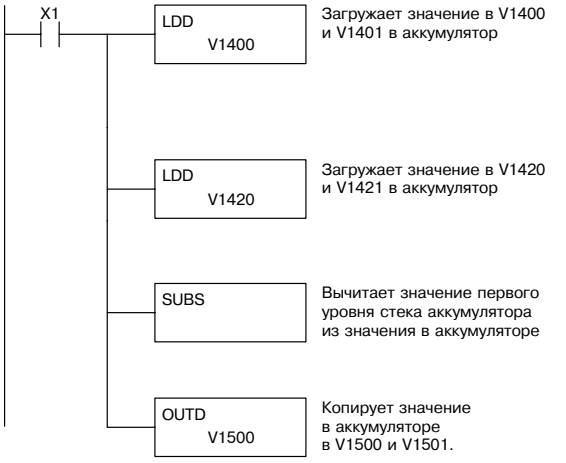


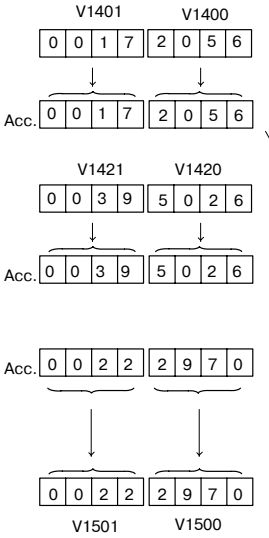
Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP64	Включен, когда 16-битовая команда вычитания приводит к заимствованию
SP65	Включен, когда 32-битовая команда вычитания приводит к заимствованию
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75	Включен, когда выполняется двоично-десятичная команда не с двоично-десятичным числом

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Значение в V1420 и V1421 загружается в аккумулятор командой Load Double с выталкиванием значения, ранее загруженного в аккумулятор, в стек аккумулятора. Двоично-десятичное значение первого уровня стека аккумулятора вычитается из двоично-десятичного значения в аккумуляторе командой Subtract Stack. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.

*Direct*SOFT Display





Стек аккумулятора после первой команды LDD

Уровень 1	X	X	X	X	X	X	X	X
Уровень 2	X	X	X	X	X	X	X	X
Уровень 3	X	X	X	X	X	X	X	X
Уровень 4	X	X	X	X	X	X	X	X
Уровень 5	X	X	X	X	X	X	X	X
Уровень 6	X	X	X	X	X	X	X	X
Уровень 7	X	X	X	X	X	X	X	X
Уровень 8	X	X	X	X	X	X	X	X

Стек аккумулятора после второй команды LDD

Уровень 1	0	0	1	7	2	0	5	6
Уровень 2	X	X	X	X	X	X	X	X
Уровень 3	X	X	X	X	X	X	X	X
Уровень 4	X	X	X	X	X	X	X	X
Уровень 5	X	X	X	X	X	X	X	X
Уровень 6	X	X	X	X	X	X	X	X
Уровень 7	X	X	X	X	X	X	X	X
Уровень 8	X	X	X	X	X	X	X	X

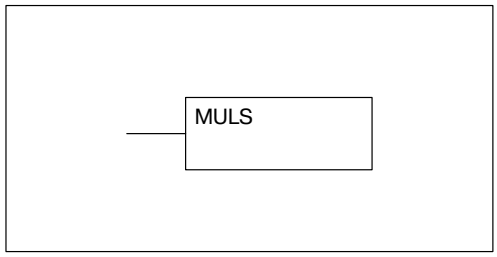
Набор на ручном программаторе

STR	X(IN)	1	↵					
LD	SHFT	D	SHFT	V	1	4	0	0 ↵
LD	SHFT	D	SHFT	V	1	4	2	0 ↵
SUB	SHFT	S	SHFT	↵				
OUT	SHFT	D	SHFT	V	1	5	0	0 ↵

Multiply Top of Stack (MULS)

√	√	√
430	440	450

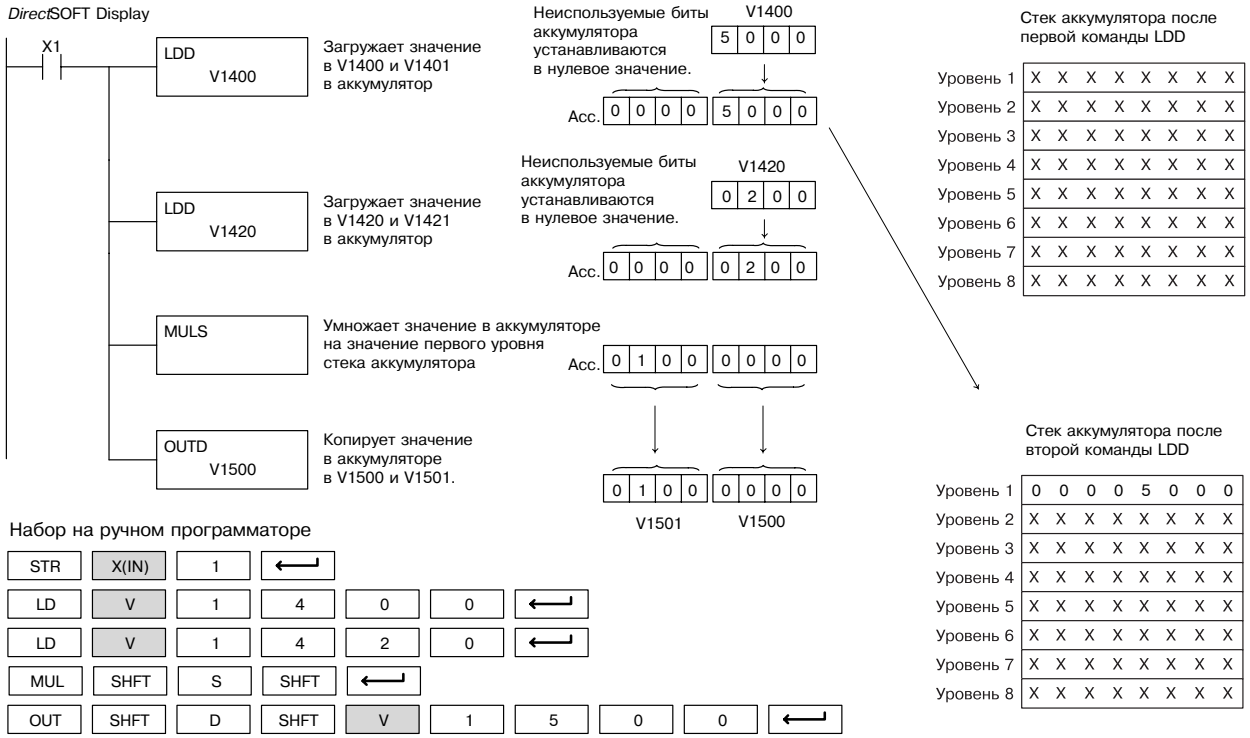
Multiply Top of Stack является 16-битовой командой, которая умножает 4-значное двоично-десятичное значение на первом уровне стека аккумулятора на 4-значное двоично-десятичное значение в аккумуляторе. Результат остается в аккумуляторе. Значение на первом уровне стека аккумулятора удаляется, а все значения стека сдвигаются на один уровень вверх.



Флаги дискретных разрядов	Описание
SP63 □	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70 □	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75 □ □	Включен, когда выполняется двоично-десятичная команда с числом не в двоично-десятичном формате

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 загружается в аккумулятор командой Load. Значение в V1420 загружается в аккумулятор командой Load Double с выталкиванием значения, ранее загруженного в аккумулятор, в стек аккумулятора. Двоично-десятичное значение первого уровня стека аккумулятора умножается на двоично-десятичное значение в аккумуляторе командой Multiply Stack. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.



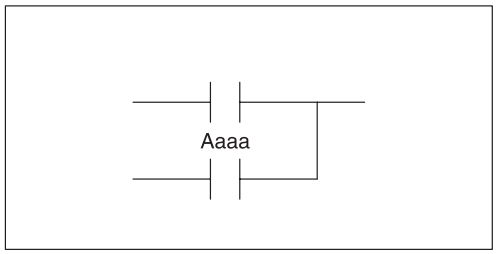
Набор на ручном программаторе

STR	X(IN)	1	↵					
LD	V	1	4	0	0	↵		
LD	V	1	4	2	0	↵		
MUL	SHFT	S	SHFT	↵				
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Divide by Top of Stack (DIVS)

√	√	√
430	440	450

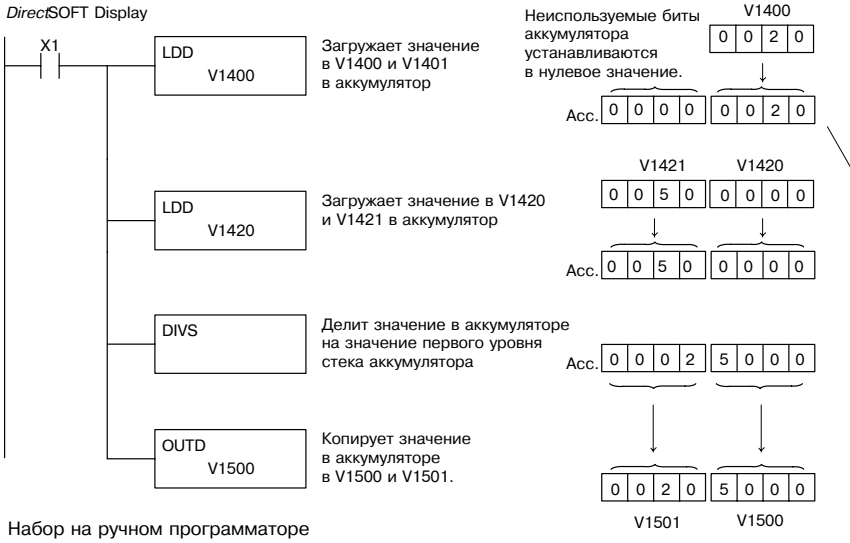
Divide by Top of Stack является 32-битовой командой, которая делит 8-значное двоично-десятичное значение в аккумуляторе на 4-значное двоично-десятичное значение на первом уровне стека аккумулятора. Результат остается в аккумуляторе, а остаток заносится в первый уровень стека аккумулятора.



Флаги дискретных разрядов	Описание
SP63 □	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70 □	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75 □ □	Включен, когда выполняется двоично-десятичная команда с числом не в двоично-десятичном формате

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 загружается в аккумулятор командой Load. Значение в V1420 загружается в аккумулятор командой Load Double с выталкиванием значения, ранее загруженного в аккумулятор, в стек аккумулятора. Двоично-десятичное значение в аккумуляторе делится на двоично-десятичное значение первого уровня стека аккумулятора командой Divide Stack. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.



Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	V	1	4	0	0	↵			
LD	SHFT	D	SHFT	V	1	4	2	0	↵
DIV	SHFT	S	SHFT	↵					
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Стек аккумулятора после первой команды LDD

Уровень 1	X	X	X	X	X	X	X	X
Уровень 2	X	X	X	X	X	X	X	X
Уровень 3	X	X	X	X	X	X	X	X
Уровень 4	X	X	X	X	X	X	X	X
Уровень 5	X	X	X	X	X	X	X	X
Уровень 6	X	X	X	X	X	X	X	X
Уровень 7	X	X	X	X	X	X	X	X
Уровень 8	X	X	X	X	X	X	X	X

Стек аккумулятора после второй команды LDD

Уровень 1	0	0	0	0	0	0	2	0
Уровень 2	X	X	X	X	X	X	X	X
Уровень 3	X	X	X	X	X	X	X	X
Уровень 4	X	X	X	X	X	X	X	X
Уровень 5	X	X	X	X	X	X	X	X
Уровень 6	X	X	X	X	X	X	X	X
Уровень 7	X	X	X	X	X	X	X	X
Уровень 8	X	X	X	X	X	X	X	X

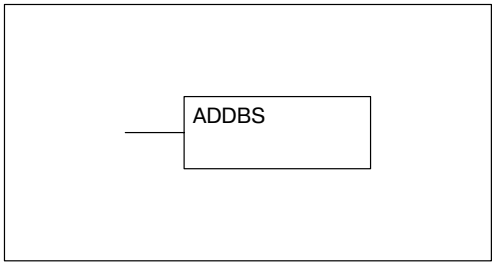
Остаток остается в первой ячейке стека

Уровень 1	0	0	0	0	0	0	0	0
Уровень 2	X	X	X	X	X	X	X	X
Уровень 3	X	X	X	X	X	X	X	X
Уровень 4	X	X	X	X	X	X	X	X
Уровень 5	X	X	X	X	X	X	X	X
Уровень 6	X	X	X	X	X	X	X	X
Уровень 7	X	X	X	X	X	X	X	X
Уровень 8	X	X	X	X	X	X	X	X

Add Binary Top of Stack (ADDBS)

X	√	√
430	440	450

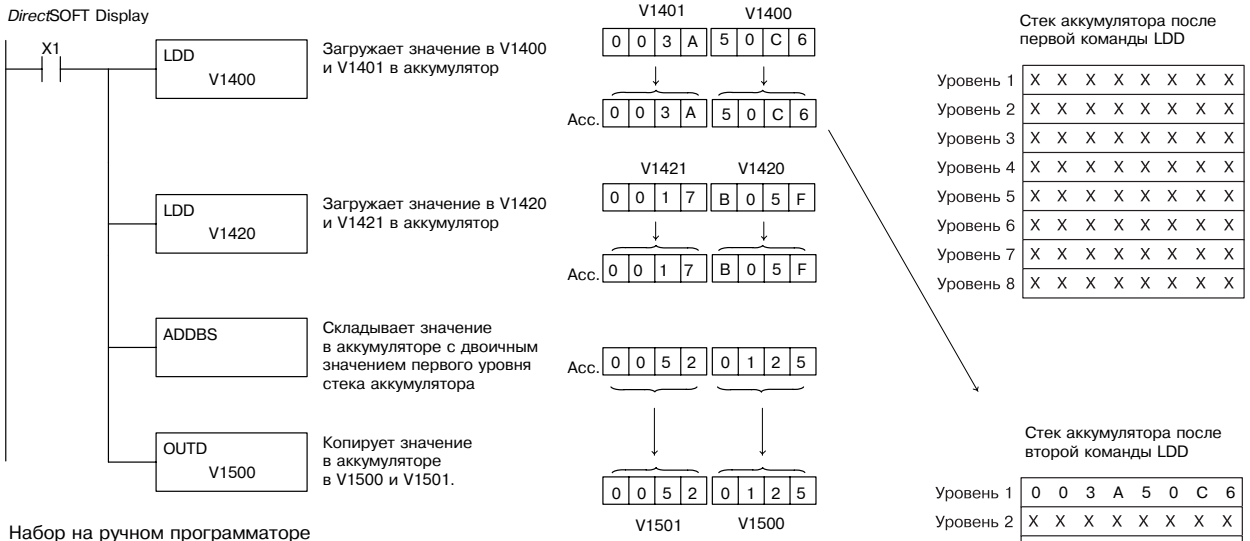
Add Binary Top of Stack является 32-битовой командой, которая складывает двоичное значение в аккумуляторе с двоичным значением на первом уровне стека аккумулятора. Результат остается в аккумуляторе. Значение на первом уровне стека аккумулятора удаляется, а все значения стека сдвигаются на один уровень вверх.



Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP66	Включен, когда 16-битовая команда сложения приводит к переносу
SP67	Включен, когда 32-битовая команда сложения приводит к переносу
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP75	Включен, когда выполняется двоично-десятичная команда не с двоично-десятичным числом

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Значение в V1420 и V1421 загружается в аккумулятор командой Load Double с выталкиванием значения, ранее загруженного в аккумулятор, в стек аккумулятора. Двоичное значение первого уровня стека аккумулятора складывается с двоичным значением в аккумуляторе командой Add Stack. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.



Стек аккумулятора после первой команды LDD

Уровень 1	X	X	X	X	X	X	X	X
Уровень 2	X	X	X	X	X	X	X	X
Уровень 3	X	X	X	X	X	X	X	X
Уровень 4	X	X	X	X	X	X	X	X
Уровень 5	X	X	X	X	X	X	X	X
Уровень 6	X	X	X	X	X	X	X	X
Уровень 7	X	X	X	X	X	X	X	X
Уровень 8	X	X	X	X	X	X	X	X

Стек аккумулятора после второй команды LDD

Уровень 1	0	0	3	A	5	0	C	6
Уровень 2	X	X	X	X	X	X	X	X
Уровень 3	X	X	X	X	X	X	X	X
Уровень 4	X	X	X	X	X	X	X	X
Уровень 5	X	X	X	X	X	X	X	X
Уровень 6	X	X	X	X	X	X	X	X
Уровень 7	X	X	X	X	X	X	X	X
Уровень 8	X	X	X	X	X	X	X	X

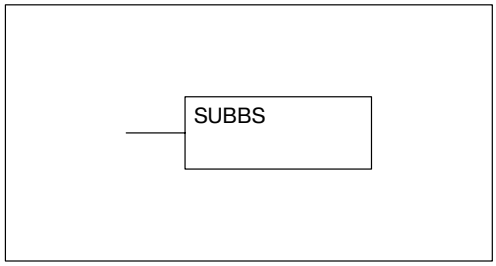
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
LD	SHFT	D	SHFT	V	1	4	2	0	↵
ADD	SHFT	B	S	SHFT	↵				
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Subtract Binary Top of Stack (SUBBS)

X	√	√
430	440	450

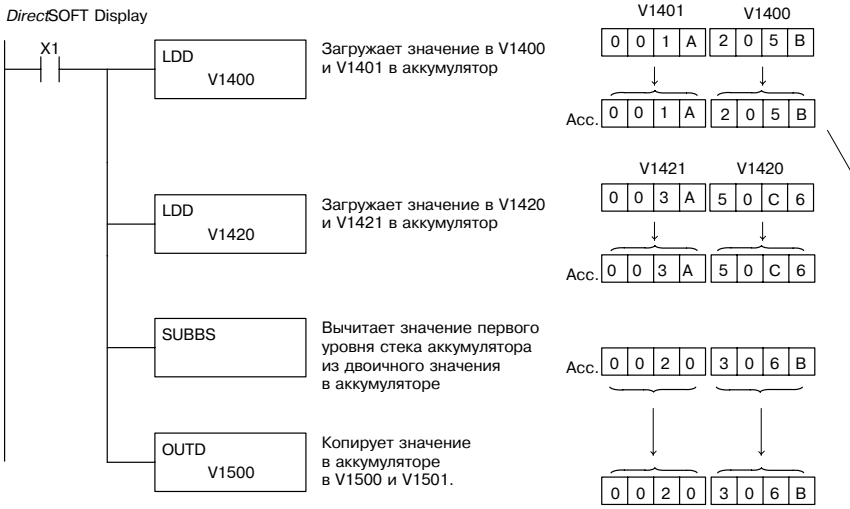
Subtract Binary Top of Stack является 32-битовой командой, которая вычитает двоичное значение на первом уровне стека аккумулятора из двоичного значения в аккумуляторе. Результат остается в аккумуляторе. Значение на первом уровне стека аккумулятора удаляется, а все значения стека сдвигаются на один уровень вверх.



Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP64	Включен, когда 16-битовая команда вычитания приводит к заимствованию
SP65	Включен, когда 32-битовая команда вычитания приводит к заимствованию
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Значение в V1420 и V1421 загружается в аккумулятор командой Load Double с выталкиванием значения, ранее загруженного в аккумулятор, в стек аккумулятора. Двоичное значение первого уровня стека аккумулятора вычитается из двоичного значения в аккумуляторе командой Subtract Stack. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.



Стек аккумулятора после первой команды LDD

Уровень 1	X	X	X	X	X	X	X	X
Уровень 2	X	X	X	X	X	X	X	X
Уровень 3	X	X	X	X	X	X	X	X
Уровень 4	X	X	X	X	X	X	X	X
Уровень 5	X	X	X	X	X	X	X	X
Уровень 6	X	X	X	X	X	X	X	X
Уровень 7	X	X	X	X	X	X	X	X
Уровень 8	X	X	X	X	X	X	X	X

Стек аккумулятора после второй команды LDD

Уровень 1	0	0	1	A	2	0	5	B
Уровень 2	X	X	X	X	X	X	X	X
Уровень 3	X	X	X	X	X	X	X	X
Уровень 4	X	X	X	X	X	X	X	X
Уровень 5	X	X	X	X	X	X	X	X
Уровень 6	X	X	X	X	X	X	X	X
Уровень 7	X	X	X	X	X	X	X	X
Уровень 8	X	X	X	X	X	X	X	X

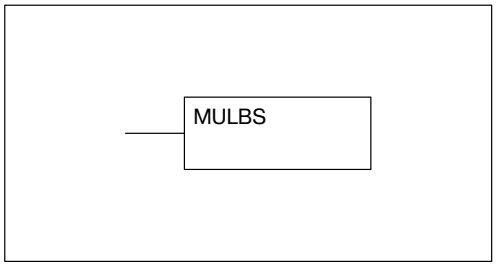
Набор на ручном программаторе

STR	X(IN)	1	⏎						
LD	SHFT	D	SHFT	V	1	4	0	0	⏎
LD	SHFT	D	SHFT	V	1	4	2	0	⏎
SUB	SHFT	B	S	SHFT	⏎				
OUT	SHFT	D	SHFT	V	1	5	0	0	⏎

Multiply Binary Top of Stack (MULBS)

X	√	√
430	440	450

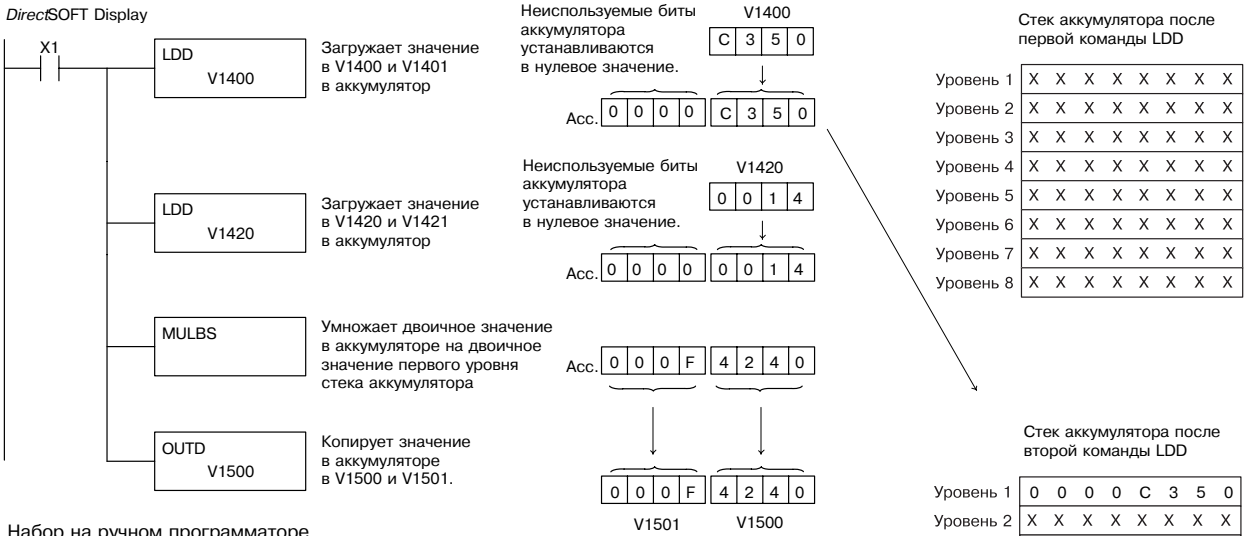
Multiply Binary Top of Stack является 16-битной командой, которая умножает 16-битное двоичное значение на первом уровне стека аккумулятора на 16-битовое двоичное значение в аккумуляторе. Результат остается в аккумуляторе и может быть 32-битовым (максимум 8 цифровых знаков). Значение на первом уровне стека аккумулятора удаляется, а все значения стека сдвигаются на один уровень вверх.



Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 загружается в аккумулятор командой Load. Значение в V1420 загружается в аккумулятор командой Load с выталкиванием значения, ранее загруженного в аккумулятор, в стек аккумулятора. Двоичное значение первого уровня стека аккумулятора умножается на двоичное значение в аккумуляторе командой Multiply Binary Stack. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.



Стек аккумулятора после первой команды LDD

Уровень 1	X	X	X	X	X	X	X	X
Уровень 2	X	X	X	X	X	X	X	X
Уровень 3	X	X	X	X	X	X	X	X
Уровень 4	X	X	X	X	X	X	X	X
Уровень 5	X	X	X	X	X	X	X	X
Уровень 6	X	X	X	X	X	X	X	X
Уровень 7	X	X	X	X	X	X	X	X
Уровень 8	X	X	X	X	X	X	X	X

Стек аккумулятора после второй команды LDD

Уровень 1	0	0	0	0	C	3	5	0
Уровень 2	X	X	X	X	X	X	X	X
Уровень 3	X	X	X	X	X	X	X	X
Уровень 4	X	X	X	X	X	X	X	X
Уровень 5	X	X	X	X	X	X	X	X
Уровень 6	X	X	X	X	X	X	X	X
Уровень 7	X	X	X	X	X	X	X	X
Уровень 8	X	X	X	X	X	X	X	X

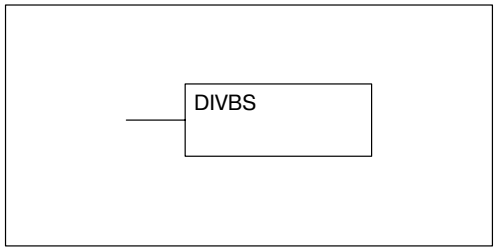
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	V	1	4	0	0	↵			
LD	V	1	4	2	0	↵			
MUL	SHFT	B	S	SHFT	↵				
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Divide Binary by Top of Stack (DIVBS)

X	√	√
430	440	450

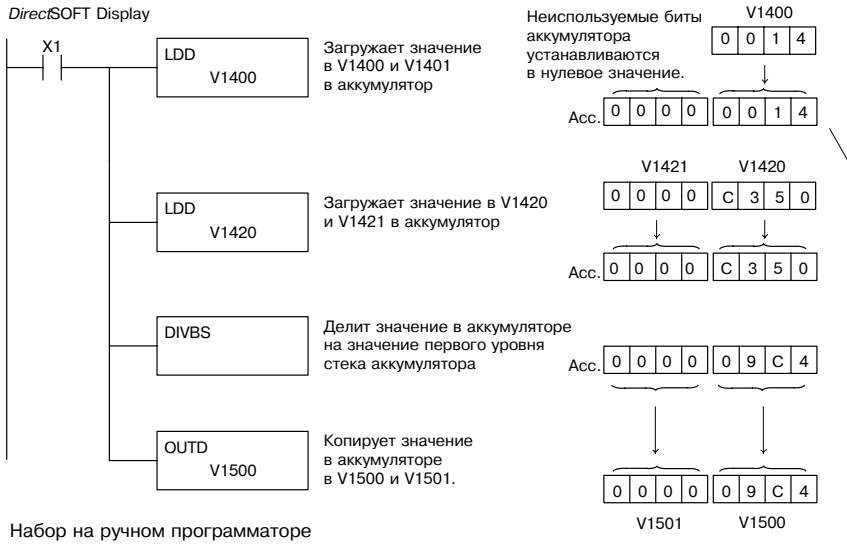
Divide Binary by Top of Stack является 32-битной командой, которая делит 32-битовое двоичное значение в аккумуляторе на 16-битное двоичное значение на первом уровне стека аккумулятора. Результат остается в аккумуляторе, а остаток заносится в первый уровень стека аккумулятора.



Флаги дискретных разрядов	Описание
SP53	Включен, когда значение операнда больше, чем значение, с которым может работать аккумулятор
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 загружается в аккумулятор командой Load. Значение в V1420 и V1421 загружается в аккумулятор командой Load Double с выталкиванием значения, ранее загруженного в аккумулятор, в стек аккумулятора. Двоичное значение в аккумуляторе делится на двоичное значение первого уровня стека аккумулятора командой Divide Binary Stack. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.



Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	V	1	4	0	0	↵			
LD	SHFT	D	SHFT	V	1	4	2	0	↵
DIV	SHFT	B	S	SHFT	↵				
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Стек аккумулятора после первой команды LDD

Уровень 1	X	X	X	X	X	X	X	X
Уровень 2	X	X	X	X	X	X	X	X
Уровень 3	X	X	X	X	X	X	X	X
Уровень 4	X	X	X	X	X	X	X	X
Уровень 5	X	X	X	X	X	X	X	X
Уровень 6	X	X	X	X	X	X	X	X
Уровень 7	X	X	X	X	X	X	X	X
Уровень 8	X	X	X	X	X	X	X	X

Стек аккумулятора после второй команды LDD

Уровень 1	0	0	0	0	0	0	2	4
Уровень 2	X	X	X	X	X	X	X	X
Уровень 3	X	X	X	X	X	X	X	X
Уровень 4	X	X	X	X	X	X	X	X
Уровень 5	X	X	X	X	X	X	X	X
Уровень 6	X	X	X	X	X	X	X	X
Уровень 7	X	X	X	X	X	X	X	X
Уровень 8	X	X	X	X	X	X	X	X

Остаток остается в первой ячейке стека

Уровень 1	0	0	0	0	0	0	0	0
Уровень 2	X	X	X	X	X	X	X	X
Уровень 3	X	X	X	X	X	X	X	X
Уровень 4	X	X	X	X	X	X	X	X
Уровень 5	X	X	X	X	X	X	X	X
Уровень 6	X	X	X	X	X	X	X	X
Уровень 7	X	X	X	X	X	X	X	X
Уровень 8	X	X	X	X	X	X	X	X

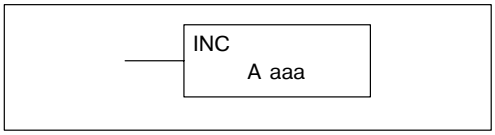
Increment (INC)

√	√	√
430	440	450

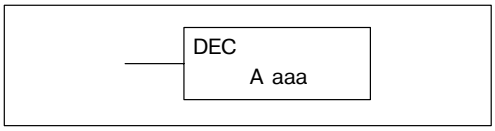
Decrement (DEC)

√	√	√
430	440	450

Команда Increment увеличивает двоично-десятичное значение в определенной ячейке V-памяти на ”1” каждый раз при выполнении команды.



Команда Decrement уменьшает двоично-десятичное значение в определенной ячейке V-памяти на ”1” каждый раз при выполнении команды.

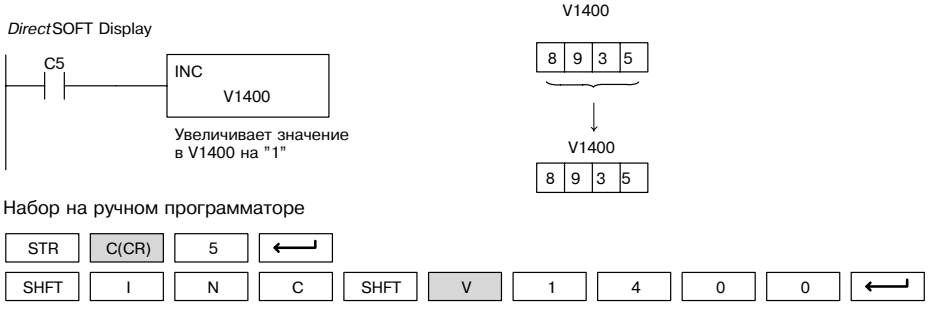


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP75	Включен, когда выполняется двоично-десятичная команда с числом не в двоично-десятичном формате

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

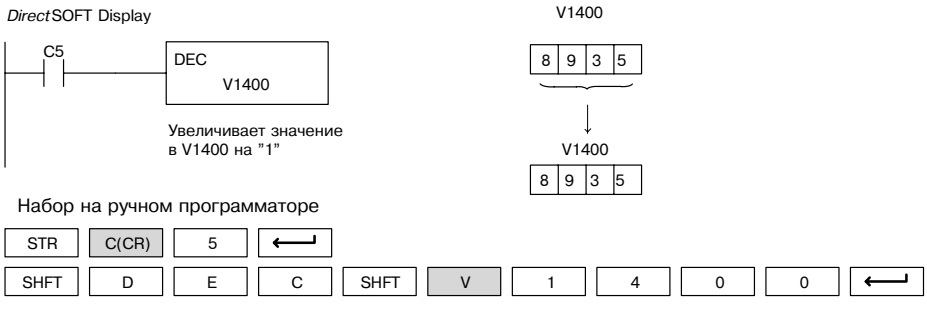
В следующем примере, когда C5 включен, команда Increment увеличивает значение в V1400 на единицу.



Набор на ручном программаторе

STR	C(CR)	5	↵							
SHFT	I	N	C	SHFT	V	1	4	0	0	↵

В следующем примере, когда C5 включен, команда Decrement уменьшает значение в V1400 на единицу.



Набор на ручном программаторе

STR	C(CR)	5	↵							
SHFT	D	E	C	SHFT	V	1	4	0	0	↵

Трансцендентные функции

Процессор DL450 имеет специальные числовые функции, дополняющие возможности работы с вещественными числами. Трансцендентные функции включают тригонометрические функции - синус, косинус и тангенс, а также их обратные функции (арксинус, арккосинус и арктангенс). Функция извлечения квадратного корня также входит в состав этих функций.

Трансцендентные математические команды работают с вещественным числом в аккумуляторе (они не могут использовать ни двоичные, ни двоично-десятичные числа). Результат выполнения команды в виде вещественного числа остается в аккумуляторе. Функция извлечения квадратного корня оперирует со всем диапазоном положительных вещественных чисел. Функции синуса, косинуса и тангенса требуют выражения чисел в радианах. Вы можете работать с углами, выраженными в градусах, преобразуя их до выполнения тригонометрической функции в радианы с помощью команды Radian (RADR). Все трансцендентные функции используют следующие биты флага.

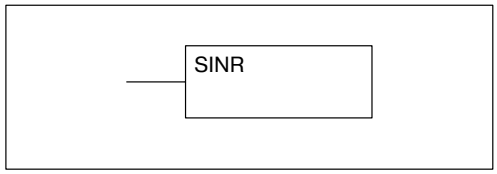
Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP72	Включен в любое время, когда значение в аккумуляторе является числом с плавающей точкой
SP73	Включен, когда команды сложения или вычитания приводят к неправильному знаковому биту
SP75	Включен, когда выполняется команда для вещественного числа с невещественным числом

Математическая функция	Область аргумента
SP53	Включен, когда значение операнда больше, чем значение, с которым может работать аккумулятор

Sine Real (SINR)

X	X	√
430	440	450

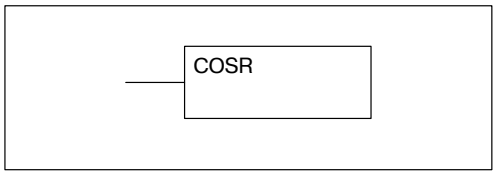
Команда Sine Real берет синус вещественного числа, находящегося в аккумуляторе. Результат остается в аккумуляторе. Как исходное число, так и результат имеют 32-битовый формат IEEE.



Cosine Real (COSR)

X	X	√
430	440	450

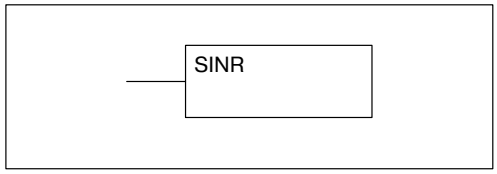
Команда Cosine Real берет косинус вещественного числа, находящегося в аккумуляторе. Результат остается в аккумуляторе. Как исходное число, так и результат имеют 32-битовый формат IEEE.



Tangent Real (TANR)

X	X	√
430	440	450

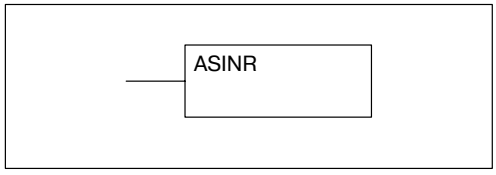
Команда Tangent Real берет тангенс вещественного числа, находящегося в аккумуляторе. Результат остается в аккумуляторе. Как исходное число, так и результат имеют 32-битовый формат IEEE.



Arc Sine Real (ASINR)

X	X	√
430	440	450

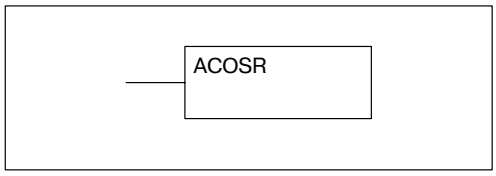
Команда Arc Sine Real берет арксинус вещественного числа, находящегося в аккумуляторе. Результат остается в аккумуляторе. Как исходное число, так и результат имеют 32-битовый формат IEEE.



Arc Cosine Real (ACOSR)

X	X	√
430	440	450

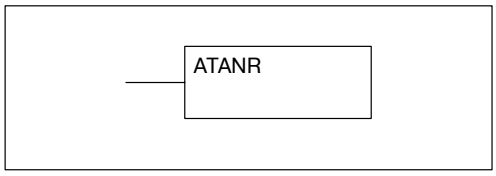
Команда Arc Cosine Real берет арккосинус вещественного числа, находящегося в аккумуляторе. Результат остается в аккумуляторе. Как исходное число, так и результат имеют 32-битовый формат IEEE.



Arc Tangent Real (ATANR)

X	X	√
430	440	450

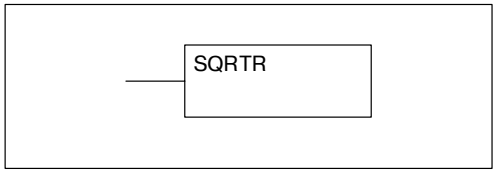
Команда Arc Tangent Real берет арктангенс вещественного числа, находящегося в аккумуляторе. Результат остается в аккумуляторе. Как исходное число, так и результат имеют 32-битовый формат IEEE.



Square Root Real (SQRTR)

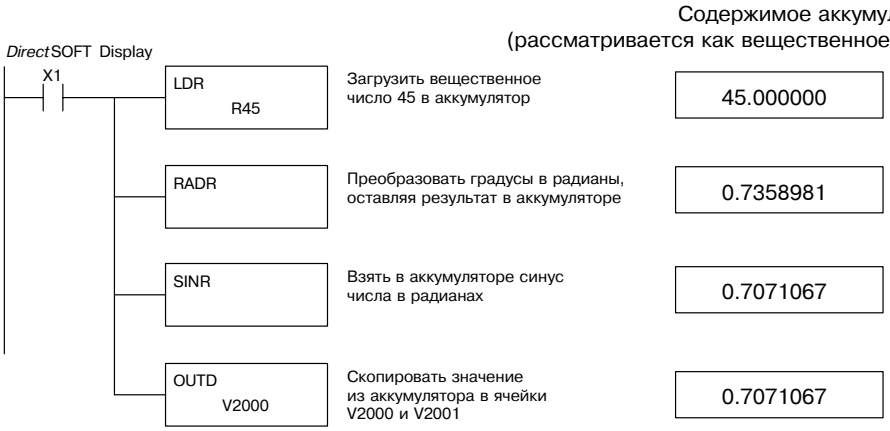
X	X	√
430	440	450

Команда Square Root Real извлекает квадратный корень из вещественного числа, находящегося в аккумуляторе. Результат остается в аккумуляторе. Как исходное число, так и результат имеют 32-битовый формат IEEE.



ПРИМЕЧАНИЕ: Функция извлечения квадратного корня может быть полезна в некоторых случаях. Однако, если Вы попытаетесь применить эту функцию к показанию диафрагменного расходомера в качестве переменной при ПИД- регулировании, то вам необходимо напомнить, что контур ПИД сам имеет встроенную функцию извлечения квадратного корня.

В следующем примере вычисляется синус 45 градусов. Поскольку эта трансцендентная функция работает только с вещественными числами, мы должны выполнить команду LDR (загрузка вещественного числа) с числом 45. Тригонометрические функции действуют только с радианами, поэтому мы должны преобразовать градусы в радианы с помощью команды RADR. После применения команды Sine Real (SINR) мы должны использовать команду Out Double (OUTD), чтобы переместить результат из аккумулятора в V-память. Результат имеет длину 32 бита, поэтому для его перемещения требуется команда Out Double.



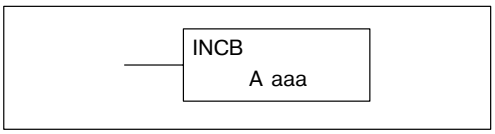
ПРИМЕЧАНИЕ: Имеющиеся ручные программаторы не поддерживают ввод вещественных чисел с автоматическим преобразованием в 32-битовый формат IEEE. Вы должны использовать DirectSOFT для ввода вещественных чисел с помощью команды команду LDR (Load Real).

Increment Binary (INCB)

√	√	√

430 440 450

Команда Increment Binary увеличивает двоичное значение в определенной ячейке V-памяти на "1" каждый раз при выполнении команды.

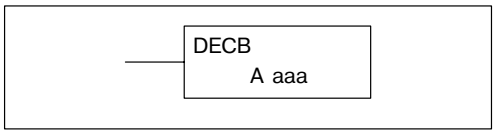


Decrement Binary (DECB)

√	√	√

430 440 450

Команда Decrement Binary уменьшает двоичное значение в определенной ячейке V-памяти на "1" каждый раз при выполнении команды.

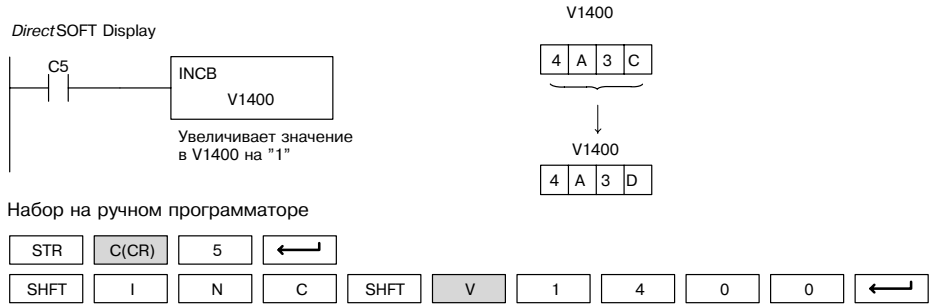


Тип данных операнда	Диапазон DL430	Диапазон DL440	Диапазон DL450
A	**aaa**	**aaa**	**aaa**
V-память V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель P	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

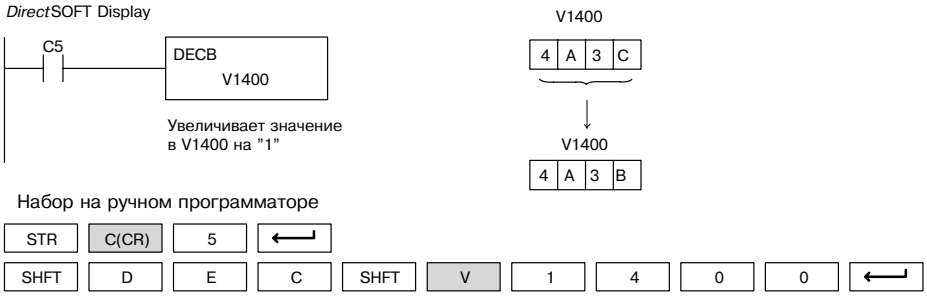
В следующем примере, когда C5 включен, двоичное значение в V1400 увеличивается на 1.



Набор на ручном программаторе

STR	C(CR)	5	↵							
SHFT	I	N	C	SHFT	V	1	4	0	0	↵

В следующем примере, когда C5 включен, двоичное значение в V1400 уменьшается на 1.



Набор на ручном программаторе

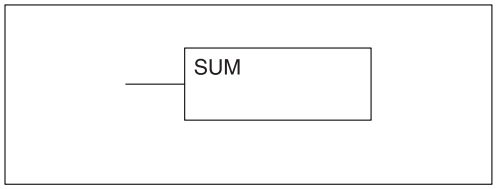
STR	C(CR)	5	↵							
SHFT	D	E	C	SHFT	V	1	4	0	0	↵

Команды работы с битами

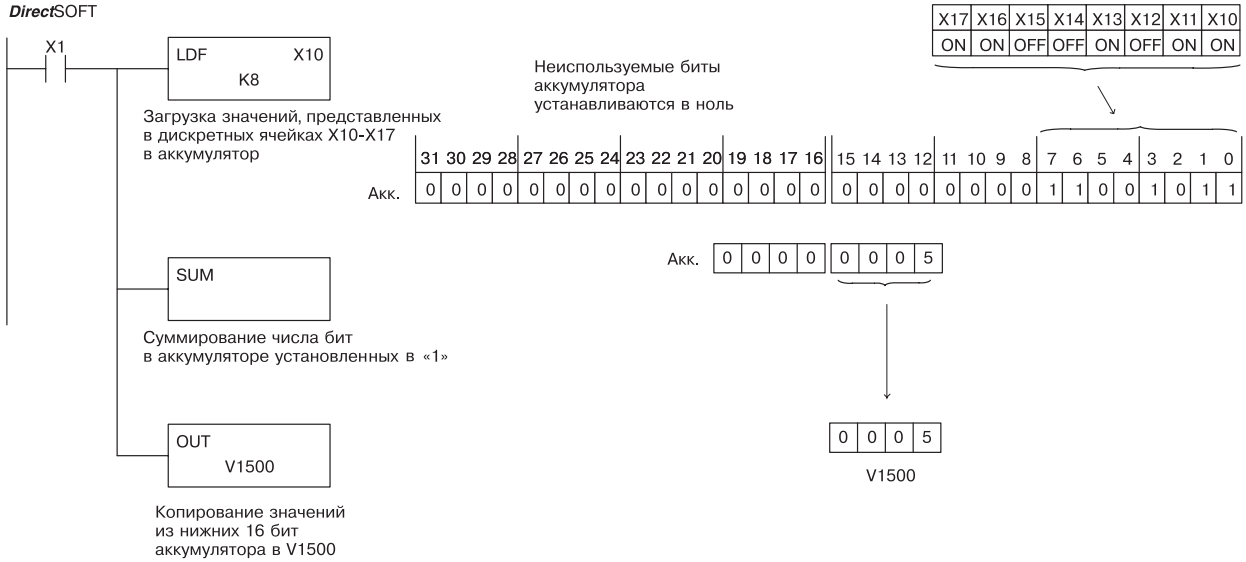
Sum (SUM)

√	√	√
430	440	450

Команда Sum считает число битов в аккумуляторе, которые установлены в ”1”. Результат в шестнадцатиричном виде находится в аккумуляторе.



В следующем примере, когда X1 включен, значение, составленное из дискретных ячеек X10-X17, загружается в аккумулятор командой Load Formatted. Подсчитывается число битов в аккумуляторе, установленных в ”1”, командой Sum. Значение в аккумуляторе копируется в V1500 командой Out.



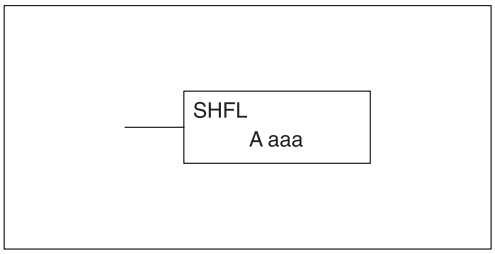
Набор на ручном программаторе

STR	X(IN)	1	↵					
LD	SHFT	F	X(IN)	1	0	K(CON)	8	↵
SHFT	S	U	M	↵				
OUT	V	1	5	0	0	↵		

Shift Left (SHFL)

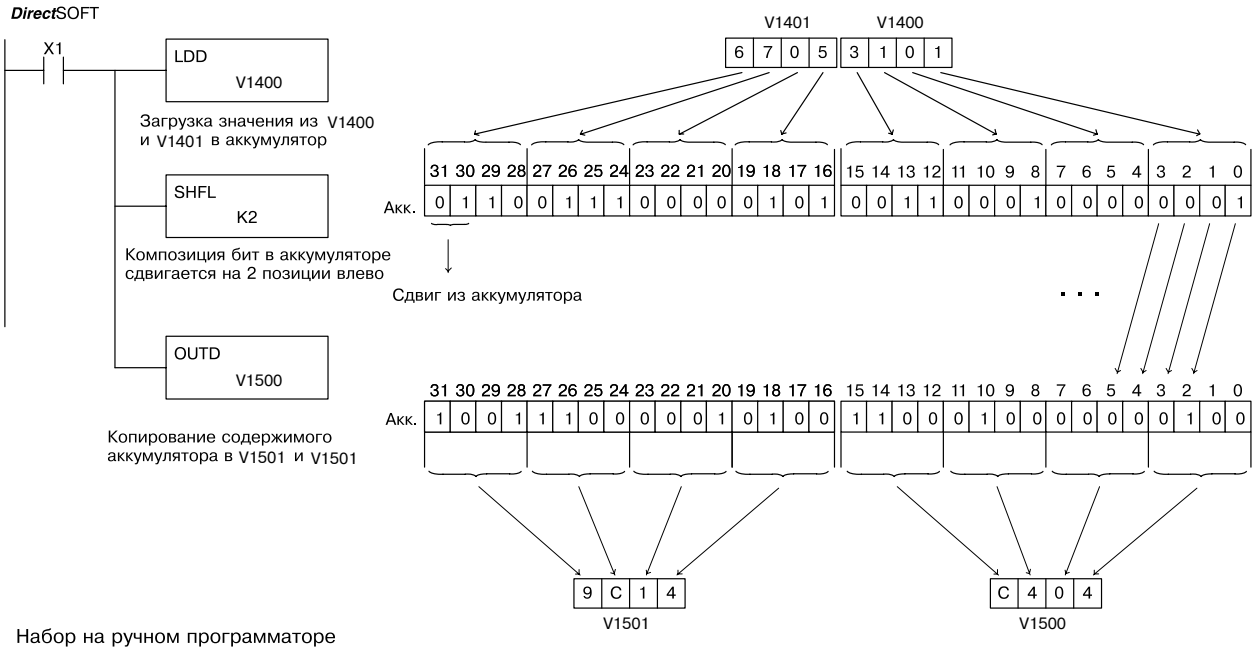
√	√	√
430	440	450

Shift Left - 32-битная команда, которая сдвигает биты в аккумуляторе на указанное число (Аааа) мест влево. Незанятые позиции заполняются нулями, а биты, сдвинутые из аккумулятора, теряются.



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	1-32	1-32	1-32

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Битовый набор в аккумуляторе сдвигается на 2 бита влево командой Shift Left. Значение в аккумуляторе копируется в V1500 и V1501, командой Out Double.



Набор на ручном программаторе

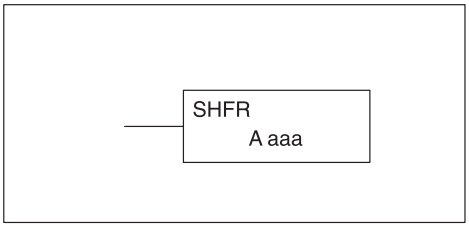
STR	X(IN)	1	⏎						
LD	SHFT	D	SHFT	V	1	4	0	0	⏎
SHFT	S	H	F	L	SHFT	K(CON)	2	⏎	
OUT	SHFT	D	SHFT	V	1	5	0	0	⏎

Shift Right (SHFR)

√	√	√

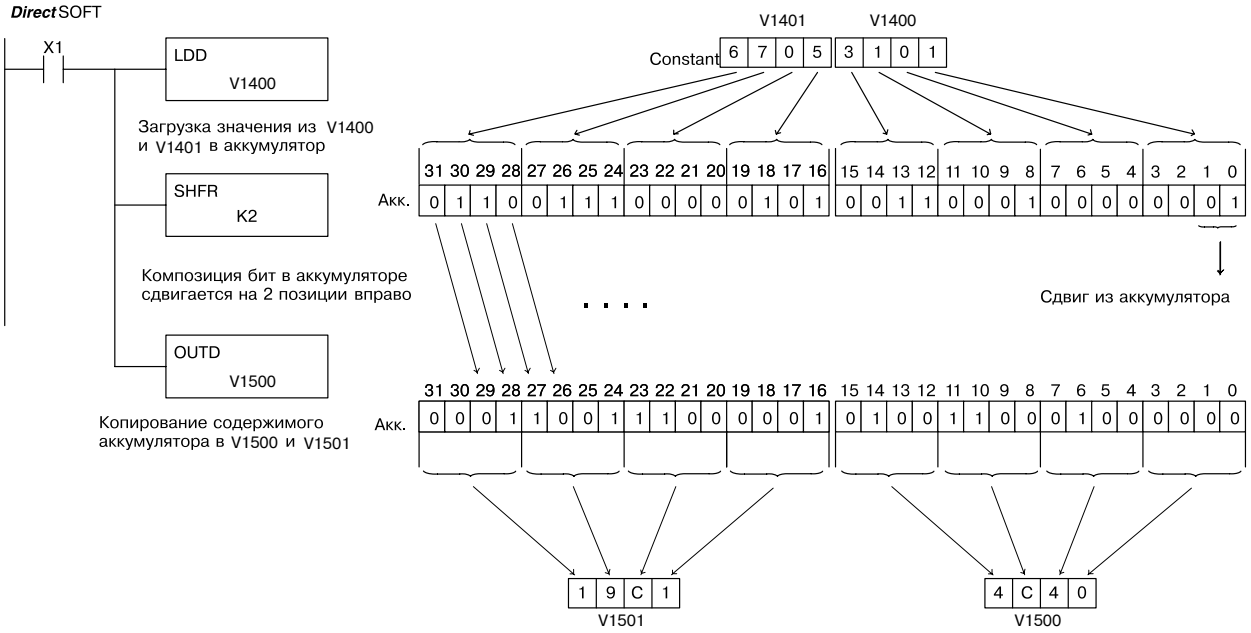
430 440 450

Shift Right - 32-битная команда, которая сдвигает биты в аккумуляторе на указанное число (Aaaa) мест вправо. Незанятые позиции заполняются нулями, а биты, сдвинутые из аккумулятора, теряются.



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	aaa	aaa	aaa
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	1-32	1-32	1-32

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Биты в аккумуляторе сдвигаются на 2 бита вправо, командой Shift Right. Значение в аккумуляторе копируется в V1500 и V1501 , командой Out Double.



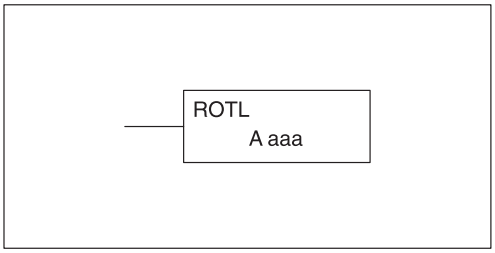
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SHFT	S	H	F	R	SHFT	K(CON)	2	↵	
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Rotate Left (ROTL)

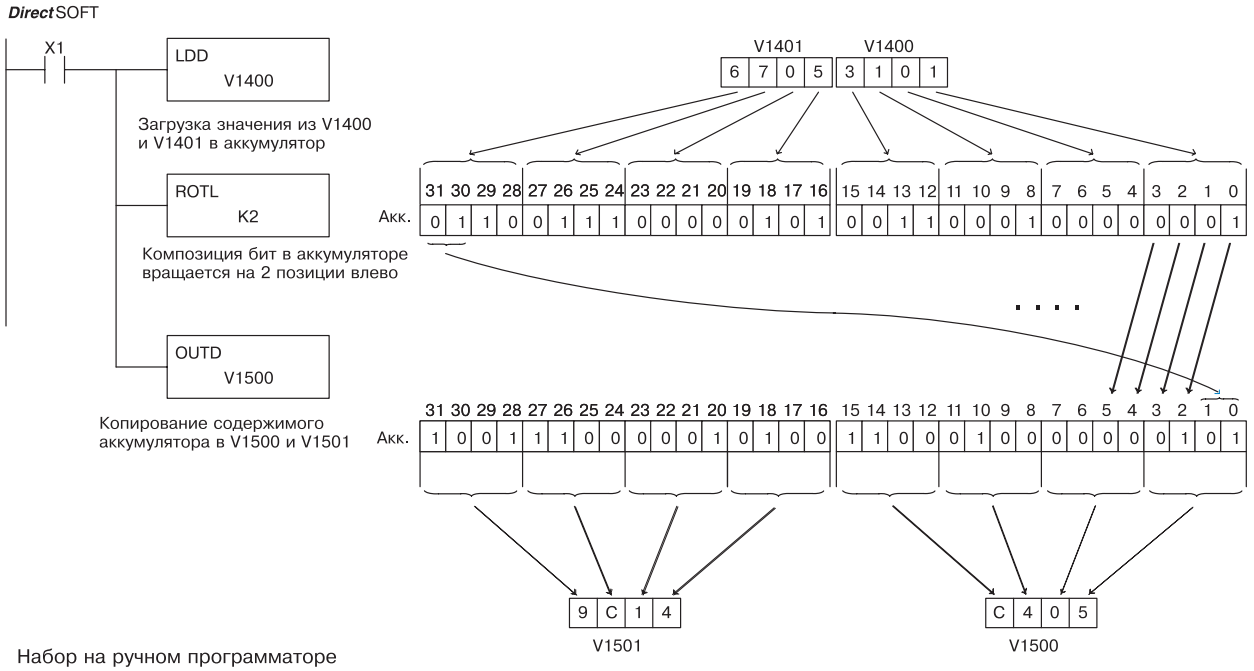
√	√	√
430	440	450

Rotate Left - 32-битовая команда, которая сдвигает биты в аккумуляторе на указанное число (Aaaa) разрядов по кругу влево.



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	1-32	1-32	1-32

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Биты в аккумуляторе сдвигаются на 2 разряда по кругу влево командой Rotate Left. Значение в аккумуляторе копируется в V1500 и V1500 командой Out Double.



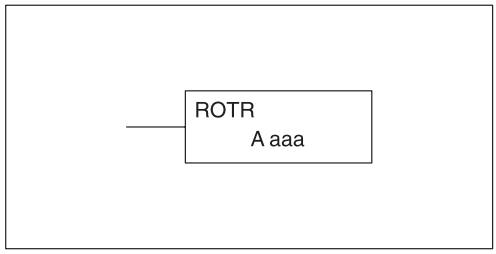
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SHFT	R	O	T	L	SHFT	K(CON)	2	↵	
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Rotate Right (ROTR)

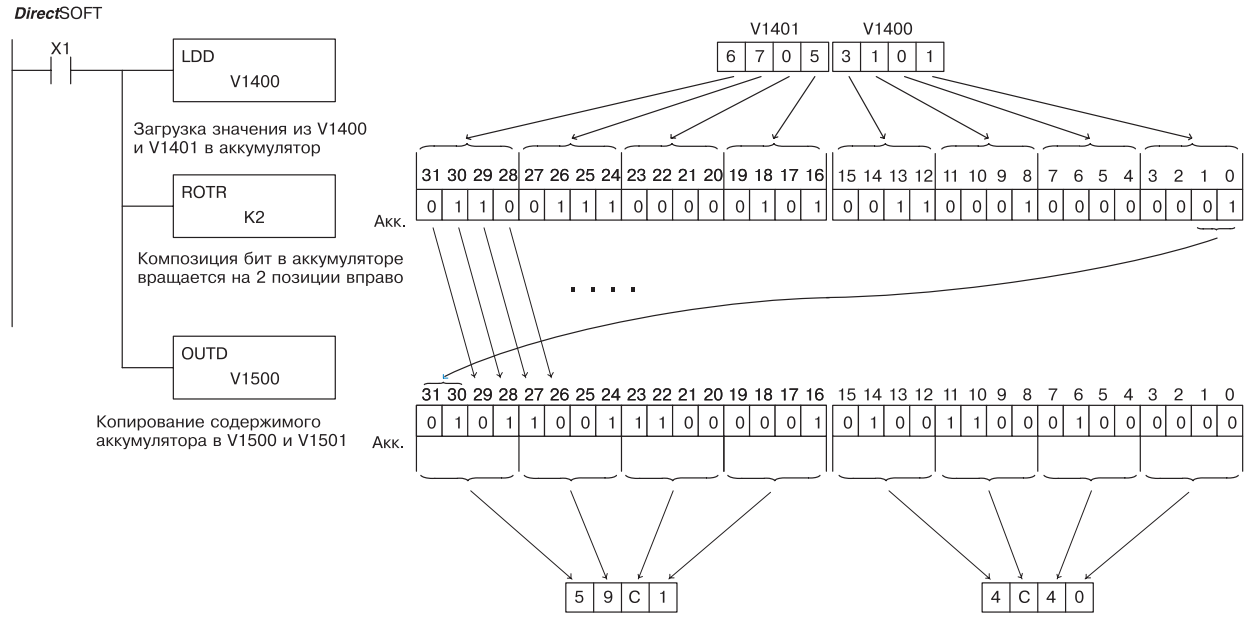
√	√	√
430	440	450

Rotate Right - 32-битная команда, которая циклически сдвигает биты в аккумуляторе на указанное число (Aaaa) разрядов по кругу вправо.



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	1-32	1-32	1-32

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Биты в аккумуляторе сдвигаются на 2 разряда по кругу вправо командой Rotate Right. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.



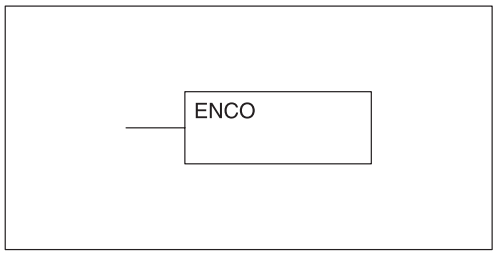
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SHFT	R	O	T	R	SHFT	K(CON)	2	↵	
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Encode (ENCO)

√	√	√
430	440	450

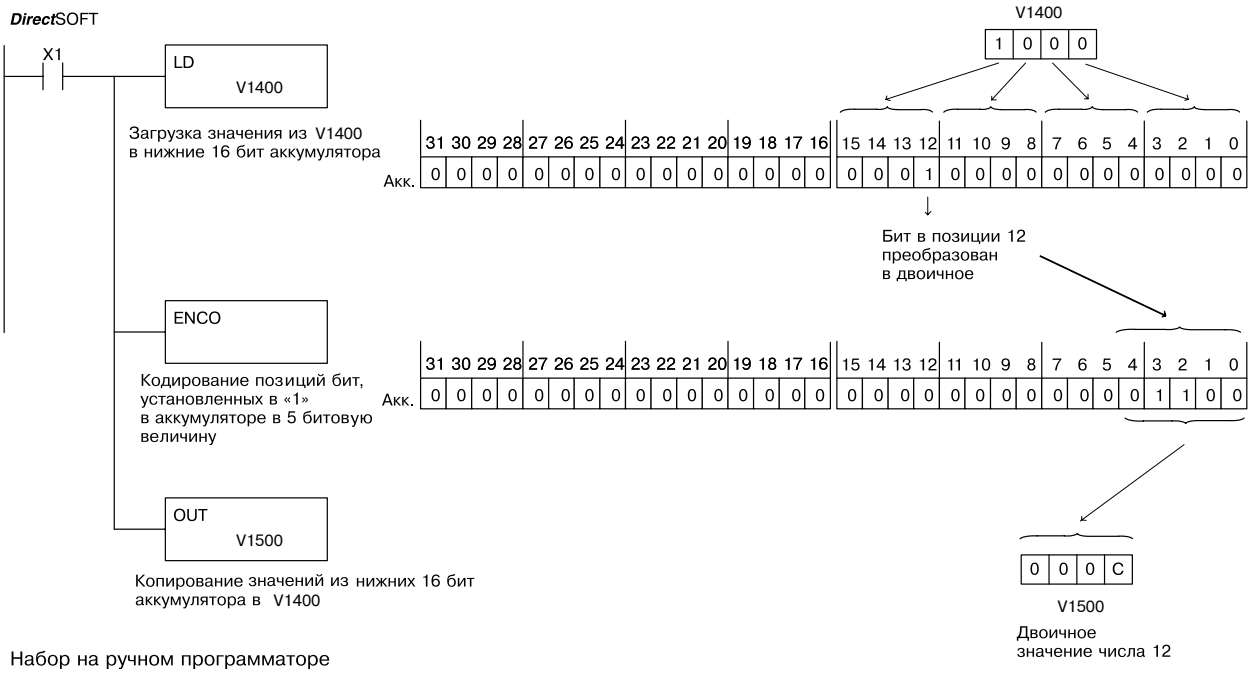
Команда Encode кодирует позицию бита в аккумуляторе, имеющего значение 1, и возвращает соответствующее двоичное представление. Если старший бит установлен в 1 (бит 31), то команда Encode поместит значение 1F шестнадцатиричное (десятичное число 31) в аккумулятор. Если кодируемое значение 0000 или 0001, то команда поместит нуль в аккумулятор. Если кодируемое значение имеет более чем одну позицию бита, установленную в "1", то самая младшая "1" будет закодирована, и SP53 будет включен. SP53 включается также, когда аккумулятор равен нулю.



Флаги дискретных разрядов	Описание
SP53	Включен, когда значение операнда больше, чем значение, с которым может работать аккумулятор

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V1400 загружается в аккумулятор командой Load. Битовый разряд, установленный в "1" в аккумуляторе кодируется в соответствующее 5-битовое двоичное значение командой Encode. Значение в младших 16 битах аккумулятора копируется в V1500 командой Out.

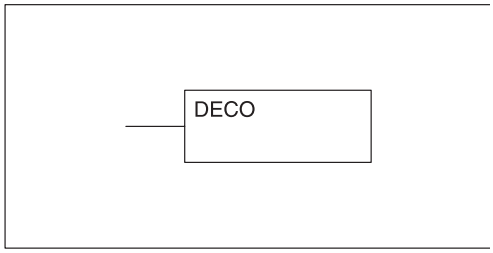


Набор на ручном программаторе

STR	X(IN)	1	↵			
LD	V	1	4	0	0	↵
SHFT	E	N	C	O	↵	
OUT	V	1	5	0	0	↵

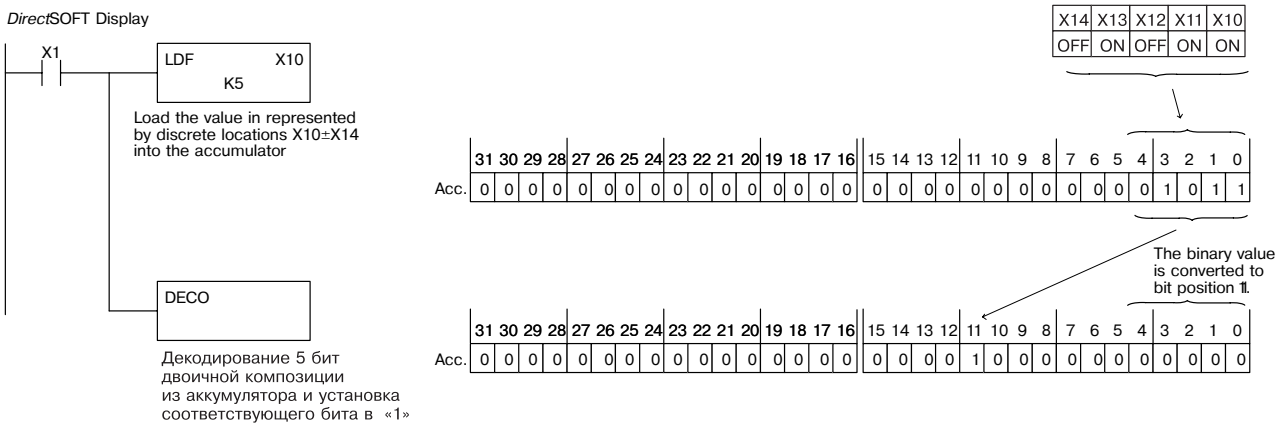
Decode (DECO)

√	√	√
430	440	450



Команда Decode декодирует 5-битовое двоичное значение 0-31 (0-1F шестнадцатеричное) в аккумуляторе, устанавливая соответствующий битовый разряд в 1. Если аккумулятор содержит значение F (шестнадцатиричное), то в аккумуляторе будет установлен бит 15. Если декодируемое значение больше, чем 31, то число делится на 32 до тех пор, пока значение не станет меньше 32, и затем это значение декодируется.

В следующем примере, когда X1 включен, значение, составленное из дискретных ячеек X10-X14, загружается в аккумулятор командой Load Formatted. Пять битов в аккумуляторе декодируются, устанавливая соответствующую позицию бита в "1" командой Decode.



Набор на ручном программаторе

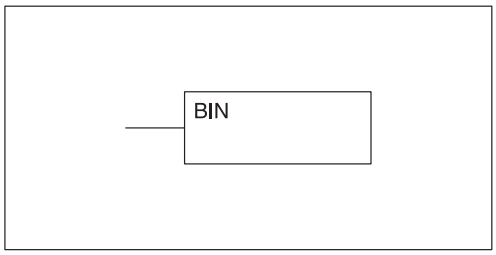
STR	X(IN)	1	↵							
LD	SHFT	F	SHFT	X(IN)	1	0	SHFT	K(CON)	5	↵
SHFT	D	E	C	O	↵					

Команды преобразования чисел

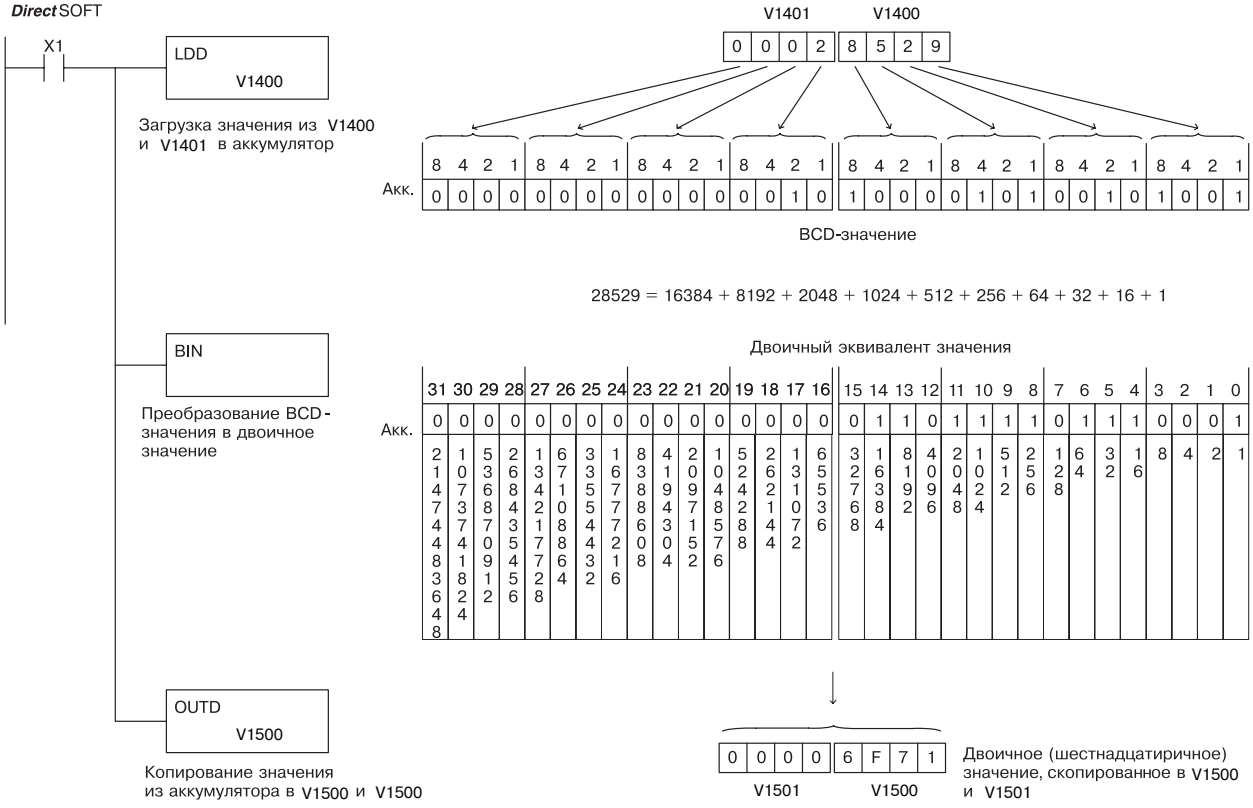
Binary (BIN)

√	√	√
430	440	450

Команда Binary преобразует двоично-десятичное значение в аккумуляторе в эквивалентное двоичное значение. Результат остается в аккумуляторе.



В следующем примере, когда X1 включен, значение в V1400 и V1401 загружается в аккумулятор командой Load Double. Двоично-десятичное значение в аккумуляторе преобразуется в двоичный (шестнадцатиричный) эквивалент командой Binary. Двоичное значение в аккумуляторе копируется в V1500 и V1501 командой Out Double. Ручной программатор будет показывать двоичное значение в V1500 и V1501 как шестнадцатеричное значение.



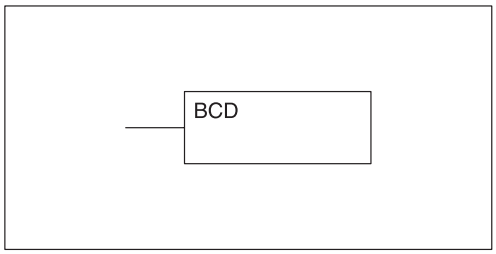
Набор на ручном программаторе

STR	X(IN)	1	⏎						
LD	SHFT	D	SHFT	V	1	4	0	0	⏎
BIN	⏎								
OUT	SHFT	D	SHFT	V	1	5	0	0	⏎

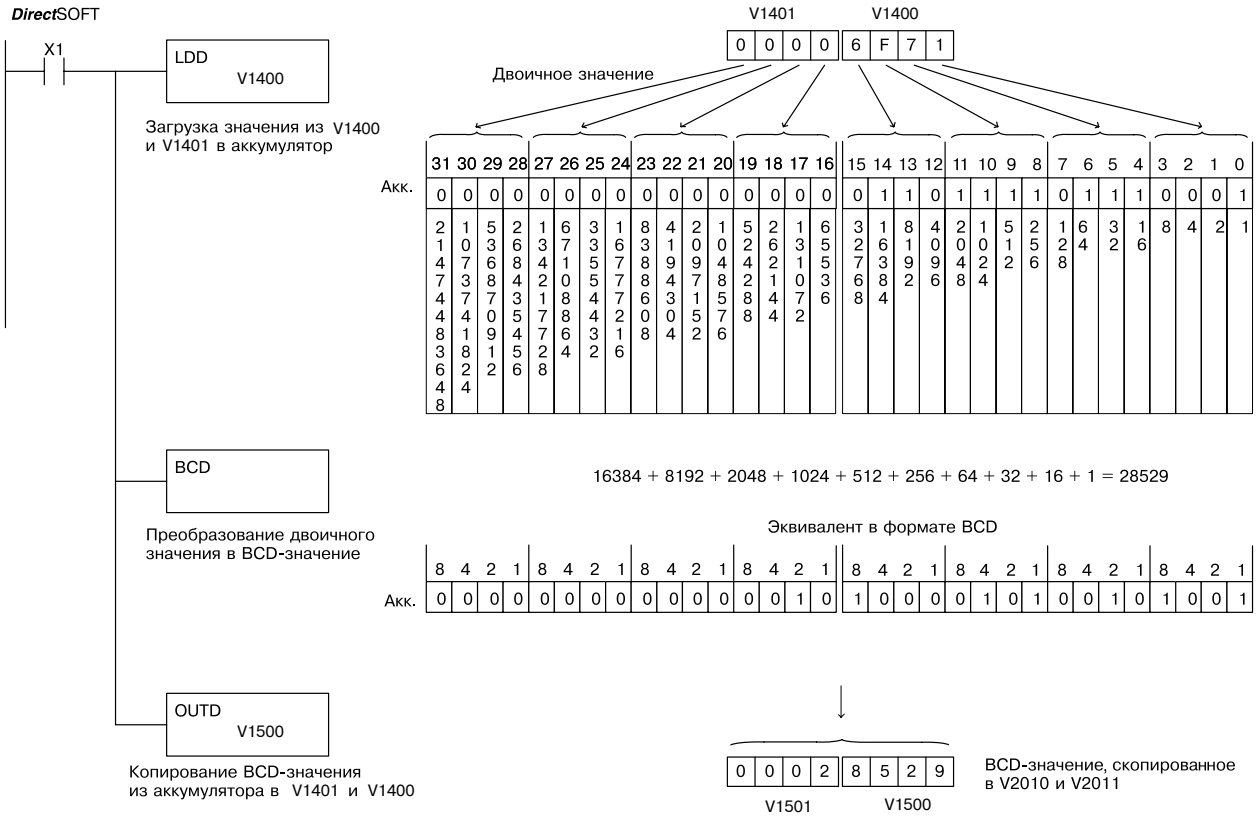
Binary Coded Decimal (BCD)

√	√	√
430	440	450

Команда Binary Coded Decimal преобразует двоичное значение в аккумуляторе в двоично-десятичное значение. Результат находится в аккумуляторе.



В следующем примере, когда X1 включен, двоичное (шестнадцатиричное) значение в V1400 и V1401 загружается в аккумулятор командой Load Double. Двоичное значение в аккумуляторе преобразуется в двоично-десятичное эквивалентное значение командой Binary Coded Decimal. Двоично-десятичное значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.



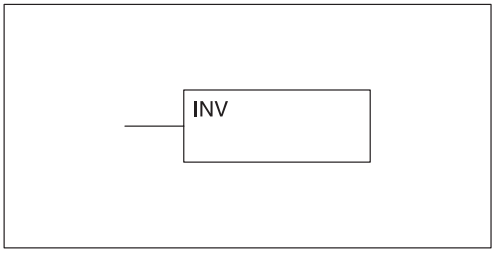
Набор на ручном программаторе

STR	X(IN)	1	⏎						
LD	SHFT	D	SHFT	V	1	4	0	0	⏎
BCD	⏎								
OUT	SHFT	D	SHFT	V	1	5	0	0	⏎

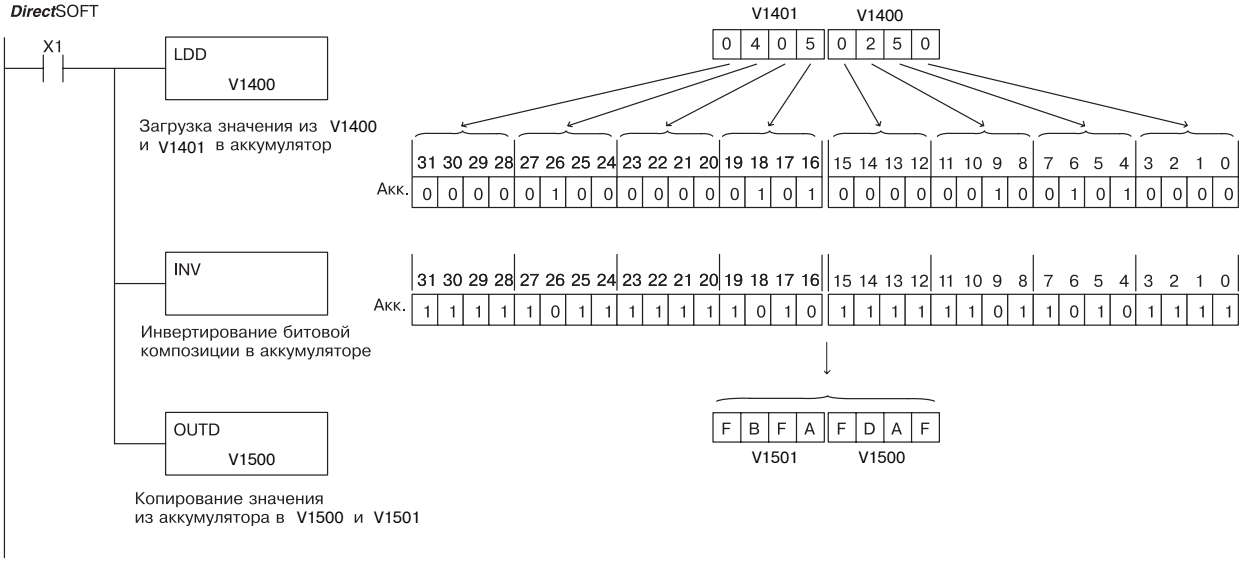
Invert (INV)

√	√	√
430	440	450

Команда Invert инвертирует или делает формат ”дополнение до 1” для 32-битового значения в аккумуляторе. Результат хранится в аккумуляторе.



В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Значение в аккумуляторе инвертируется командой Invert. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.



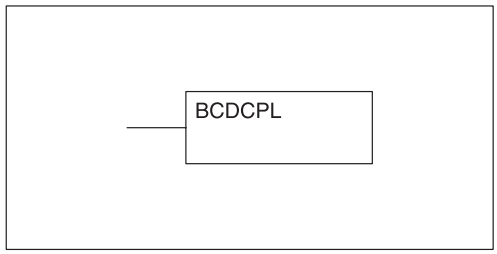
Набор на ручном программаторе

STR	X(IN)	1	⟵						
LD	SHFT	D	SHFT	V	1	4	0	0	⟵
SHFT	I	N	V	⟵					
OUT	SHFT	D	SHFT	V	1	5	0	0	⟵

Ten's Complement (BCDCPL)

√	√	√
430	440	450

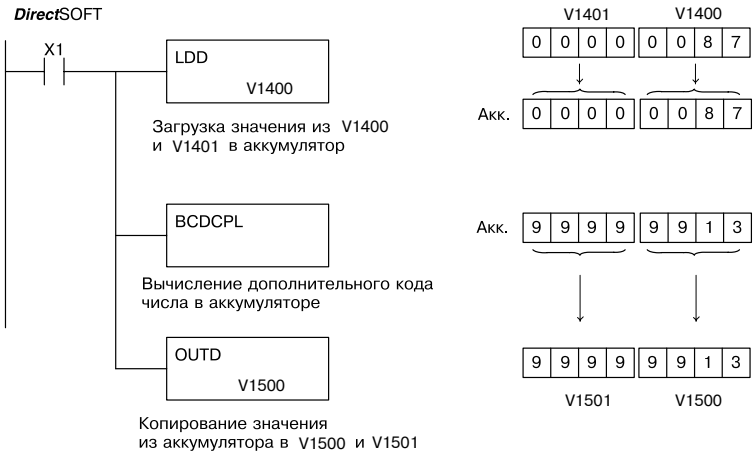
Команда Ten's Complement делает формат "дополнение до 10" (двоично-десятичный) из 8-значного цифрового значения аккумулятора. Результат находится в аккумуляторе. Вычисление для этой команды происходит по следующей схеме:



100000000
– значение аккумулятора

число в формате "дополнение до 10".

В следующем примере, когда X1 включен, значение в V1400 и V1401 загружается в аккумулятор. Формат "дополнение до 10" делается для 8 цифровых знаков аккумулятора командой Ten's Complement. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.



Набор на ручном программаторе

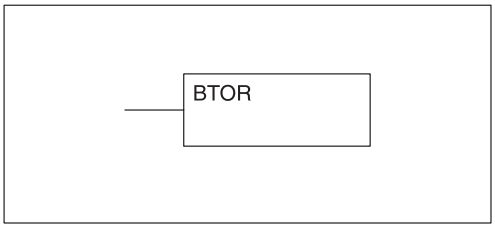
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SHFT	B	T	O	R	SHFT	↵			
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

ПРИМЕЧАНИЕ: Если в вашей программе выполняется операция вычитания, которая приводит к отрицательному переносу цифр, то можно использовать команду BCDCPL для того, чтобы найти абсолютную разность между этими двумя значениями при вычитании.

Binary to Real Conversion (BTOR)

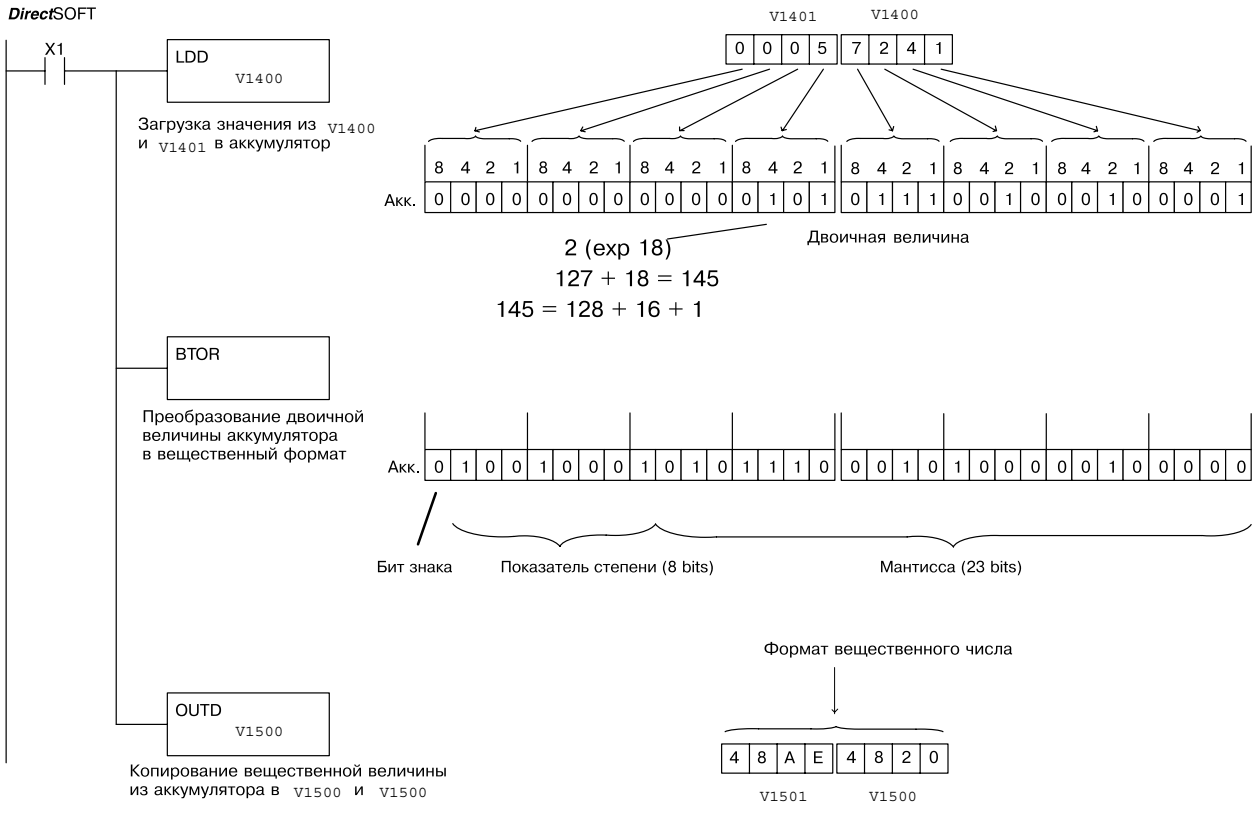
X	X	√
430	440	450

Команда Binary-to-Real преобразует двоичное значение в аккумуляторе в его эквивалентное вещественное число (с плавающей точкой). Результат находится в аккумуляторе. Оба, двоичное и вещественное, числа могут использовать все 32 бита аккумулятора.



Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное

В следующем примере, когда X1 включен, значение в V1400 и V1401 загружается в аккумулятор командой Load Double. Команда Binary-to-Real преобразует двоичное значение в аккумуляторе в эквивалентное вещественное число. Двоичный вес самого старшего двоичного разряда (MSB) преобразуется в порядок вещественного числа при добавлении к нему 127 (десятичных). Затем оставшиеся биты копируются в мантиссу, как показано на рисунке. Значение в аккумуляторе копируется в V1500 и V1501, командой Out Double. Ручной программатор будет показывать двоичное значение в V1500 и V1501 как шестнадцатиричное значение.



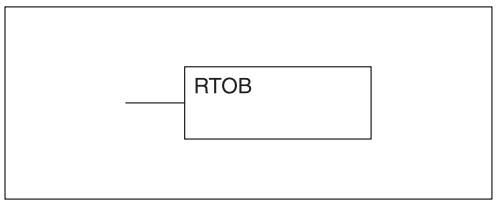
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SHFT	B	T	O	R	SHFT	↵			
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Real to Binary Conversion (RTOB)

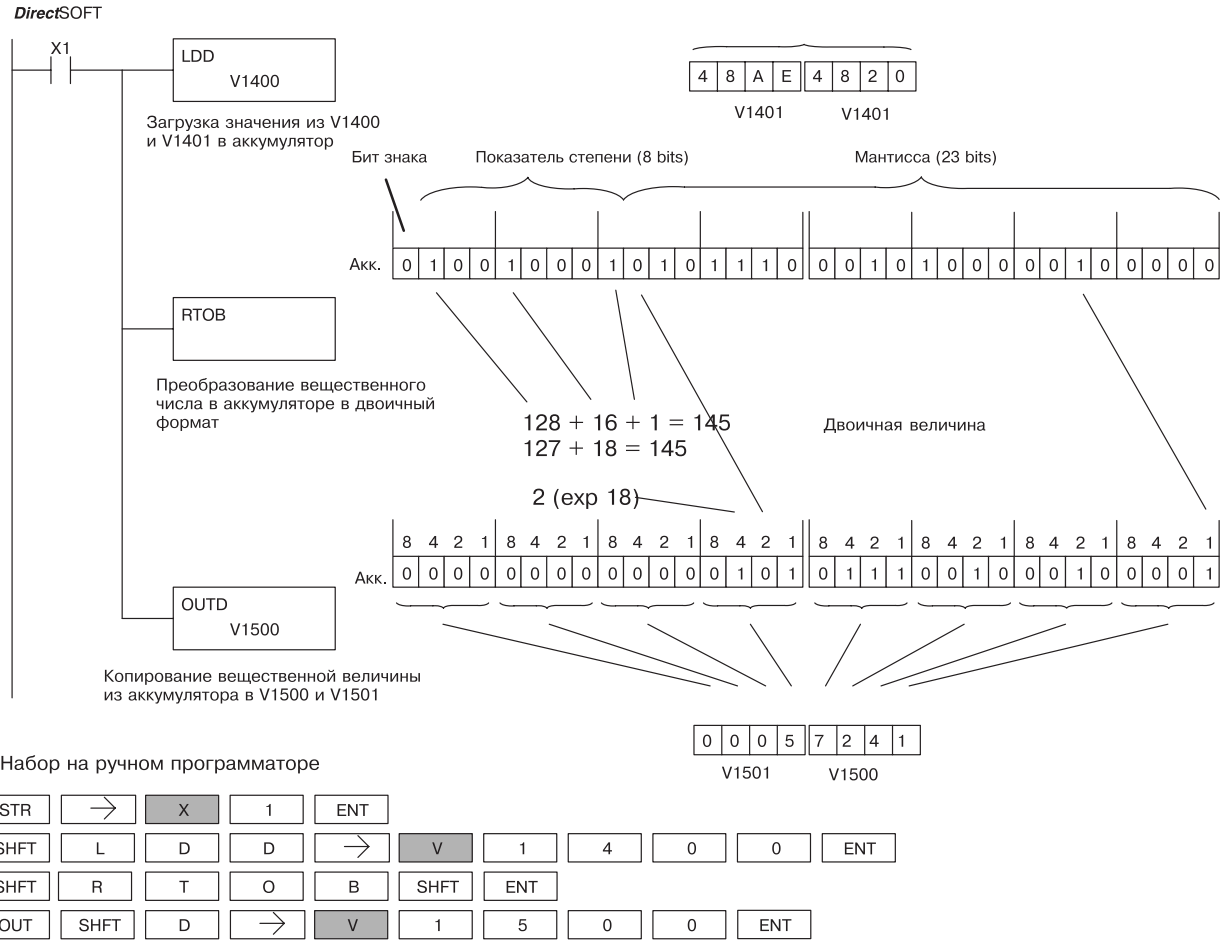
X	X	√
430	440	450

Команда Real-to-Binary преобразует вещественное число в аккумуляторе в двоичное значение. Результат находится в аккумуляторе. Оба, двоичное и вещественное, числа могут использовать все 32 бита аккумулятора.



Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP72	Включен в любое время, когда значение в аккумуляторе является числом с плавающей точкой
SP74	Включен, когда команды сложения или вычитания приводят к неправильному знаковому биту
SP75	Включен, когда выполняется команда для вещественного числа с невещественным числом

В следующем примере, когда X1 включен, значение в V1400 и V1401 загружается в аккумулятор командой Load Double. Команда Real-to-Binary преобразует вещественное значение в аккумуляторе в эквивалентное двоичное число. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double. Ручной программатор будет показывать двоичное значение в V1500 и V1501 как шестнадцатеричное значение.



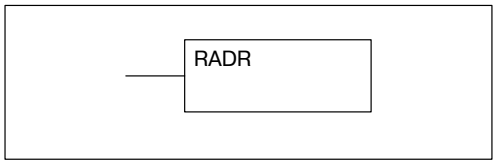
Набор на ручном программаторе

STR	→	X	1	ENT					
SHFT	L	D	D	→	V	1	4	0	0	ENT
SHFT	R	T	O	B	SHFT	ENT				
OUT	SHFT	D	→	V	1	5	0	0	ENT	

Radian Real Conversion (RADR)

X	X	√
430	440	450

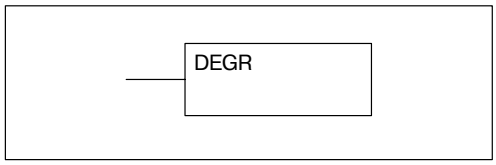
Команда Radian Real Conversion преобразует вещественное значение градусов, находящееся в аккумуляторе, в эквивалентное вещественное число в радианах. Результат остается в аккумуляторе.



Degree Real Conversion (DEGR)

X	X	√
430	440	450

Команда Degree Real Conversion преобразует вещественное значение радианов, находящееся в аккумуляторе, в эквивалентное вещественное число в градусах. Результат остается в аккумуляторе.

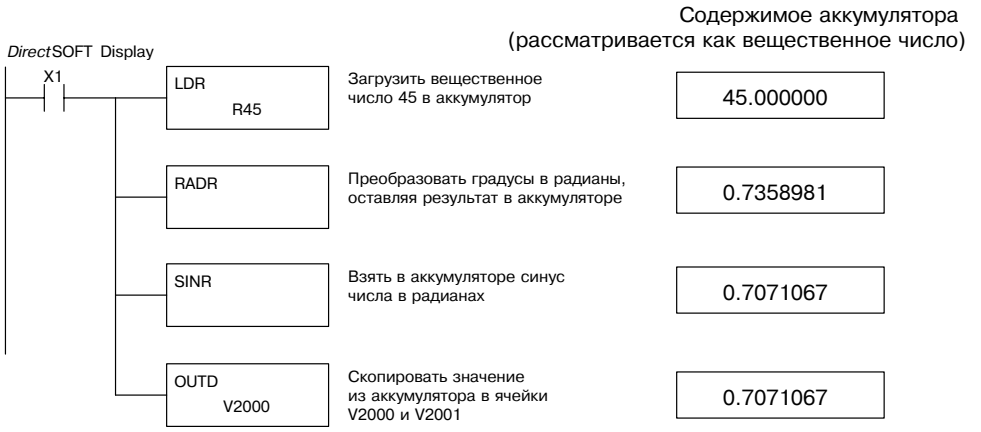


Эти две команды преобразуют вещественные числа в аккумуляторе из формата градусов в формат радианов и обратно. В формате градусов полный круг содержит 360 градусов. В формате радианов полный круг содержит 2p радиан. Команды преобразуют как положительные, так и отрицательные вещественные числа, а углы могут быть больше, чем полный круг. Эти функции очень полезны в сочетании с тригонометрическими функциями (см. раздел по математическим командам).

Флаги дискретных разрядов	Описание
SP63	Включен, когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	Включен в любое время, когда значение в аккумуляторе отрицательное
SP71	Включен в любое время, когда V-память, определенная как указатель (P) не действительна
SP72	Включен в любое время, когда значение в аккумуляторе является числом с плавающей точкой
SP74	Включен в любое время, когда математическая операция с плавающей точкой приводит к исчезновению разрядов (сигнал ошибки)
SP75	Включен, когда выполняется команда для вещественного числа с невещественным числом

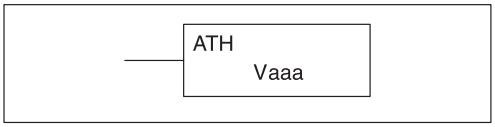
ПРИМЕЧАНИЕ: Имеющиеся ручные программаторы не поддерживают ввод вещественных чисел с автоматическим преобразованием в 32-битовый формат IEEE. Вы должны использовать DirectSOFT для ввода вещественных чисел с помощью команды команду LDR (Load Real).

В следующем примере вычисляется синус 45 градусов. Поскольку трансцендентная функция работает только с вещественными числами, мы должны выполнить команду LDR (загрузка вещественного числа) с числом 45. Тригонометрические функции действуют только с радианами, поэтому мы должны преобразовать градусы в радианы с помощью команды RADR. После применения команды Sine Real (SINR) мы должны использовать команду Out Double (OUTD), чтобы переместить результат из аккумулятора в V-память. Результат имеет длину 32 бита, поэтому для его перемещения требуется команда Out Double.



ASCII to HEX (ATH)

X	√	√
430	440	450



Команда ASCII-to-HEX преобразует таблицу ASCII значений в таблицу шестнадцатеричных значений. ASCII значения состоят из двух цифр, а их шестнадцатиричные эквиваленты - из одной. Это означает, что ASCII таблица четырех ячеек V-памяти будет требовать только две ячейки V-памяти для эквивалентной шестнадцатиричной таблицы. Функциональные параметры загружаются в стек аккумулятора и в аккумулятор двумя дополнительными командами. Ниже перечислены шаги, необходимые для создания программы с функцией преобразования ASCII в HEX. Пример на следующей странице показывает программу для функции преобразования ASCII в HEX.

Шаг 1: Загрузить число ячеек V-памяти для ASCII таблицы в первый уровень стека аккумулятора.

Шаг 2: Загрузить начальную ячейку V-памяти для ASCII таблицы в аккумулятор. Этот параметр должен быть шестнадцатиричным значением.

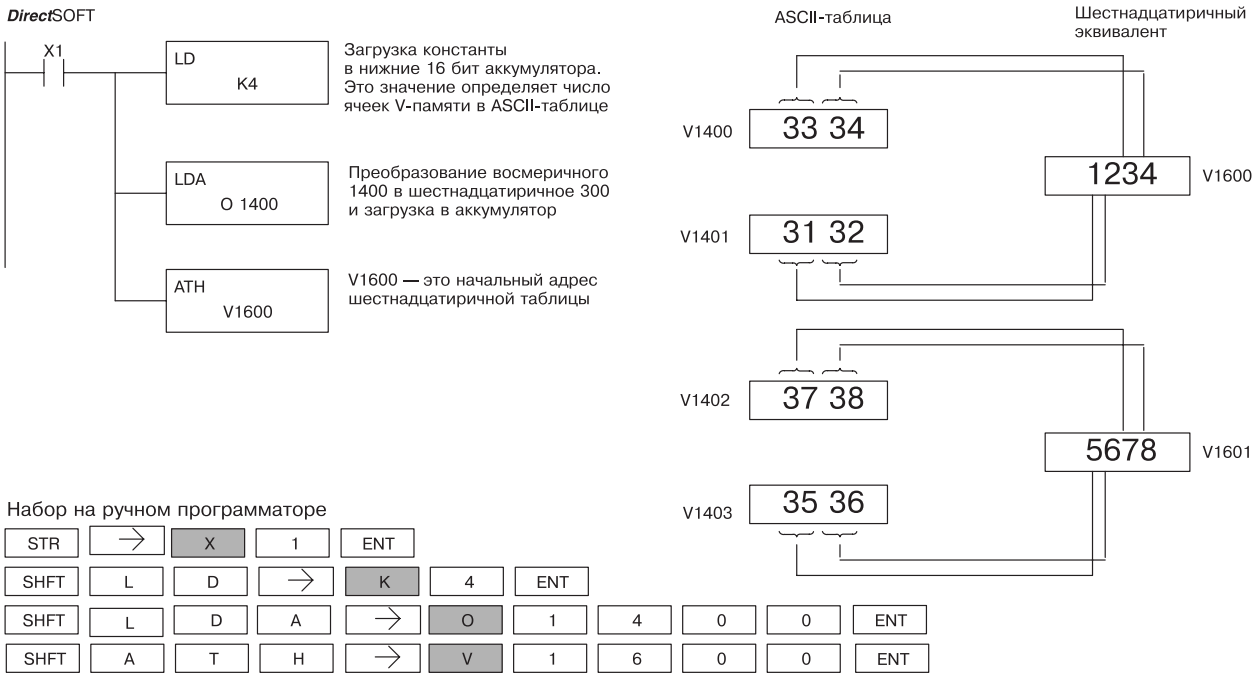
Шаг 3: Определить стартовую ячейку V-памяти (Vaaa) для шестнадцатиричной таблицы в команде ASCII-to-HEX.

Полезная подсказка: Для параметров, которые требуют шестнадцатиричных значений при ссылке на ячейки памяти, может быть использована команда LDA для преобразования восьмеричного адреса в шестнадцатиричный эквивалент и для загрузки значения в аккумулятор.

Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42

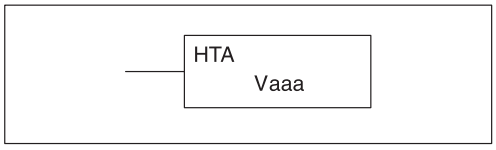
В примере на следующей странице, когда X1 включен, константа (K4) загружается в аккумулятор командой Load и помещается на первый уровень стека аккумулятора, когда выполняется следующая команда Load. Стартовая ячейка для ASCII таблицы (V1400) загружается в аккумулятор командой Load Address. Стартовая ячейка для шестнадцатиричной таблицы (V1600) указывается в команде ASCII-to-HEX. Таблица ниже показывает список возможных ASCII значений для преобразования ASCII в HEX.

Возможные значения ASCII для преобразования ASCII в HEX			
Значение ASCII	HEX значение	Значение ASCII	HEX значение
30	0	38	8
31	1	39	9
32	2	41	A
33	3	42	B
34	4	43	C
35	5	44	D
36	6	45	E
37	7	46	F



HEX to ASCII (HTA)

X	√	√
430	440	450



Команда HEX-to-ASCII преобразует таблицу шестнадцатиричных значений в таблицу ASCII значений. Шестнадцатиричные значения состоят из одной цифры, а их ASCII эквиваленты - из двух. Это означает, что шестнадцатиричная таблица двух ячеек V-памяти будет требовать четыре ячейки V-памяти для эквивалентной ASCII таблицы. Функциональные параметры загружаются в стек аккумулятора и в аккумулятор двумя дополнительными командами. Ниже перечислены шаги, необходимые для создания программы с функцией преобразования HEX в ASCII. Пример на следующей странице показывает программу для функции преобразования HEX в ASCII.

Шаг 1: Загрузить число ячеек V-памяти для шестнадцатиричной таблицы в первый уровень стека аккумулятора.

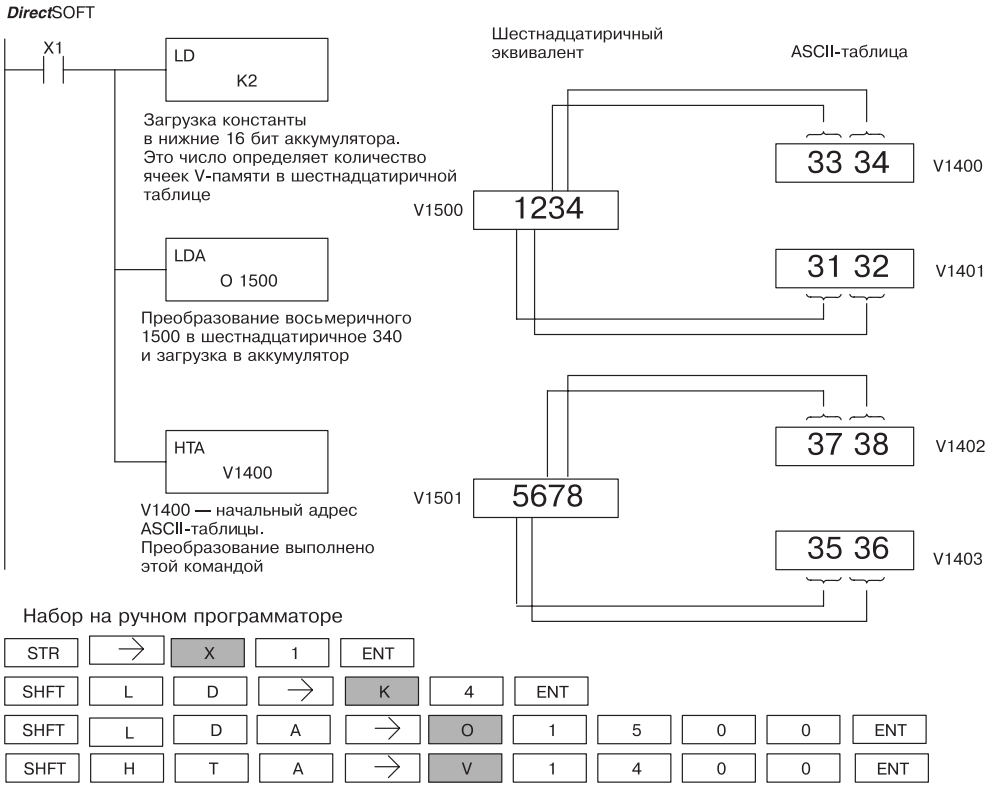
Шаг 2: Загрузить начальную ячейку V-памяти для шестнадцатиричной таблицы в аккумулятор. Этот параметр должен быть шестнадцатиричным значением.

Шаг 3: Определить стартовую ячейку V-памяти (Vaaa) для ASCII таблицы в команде HEX-to-ASCII.

Полезная подсказка: Для параметров, которые требуют шестнадцатиричных значений при ссылке на ячейки памяти, может быть использована команда LDA для преобразования восьмеричного адреса в шестнадцатиричный эквивалент и для загрузки значения в аккумулятор.

Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42

В следующем примере, когда X1 включен, константа (K2) загружается в аккумулятор командой Load. Стартовая ячейка для шестнадцатиричной таблицы (V1500) загружается в аккумулятор командой Load Address. Стартовая ячейка для ASCII таблицы (V1400) указывается в команде HEX-to-ASCII.



Набор на ручном программаторе

STR	→	X	1	ENT						
SHFT	L	D	→	K	4	ENT				
SHFT	L	D	A	→	O	1	5	0	0	ENT
SHFT	H	T	A	→	V	1	4	0	0	ENT

В таблице ниже перечислены допустимые значения ASCII для преобразования HEX в ASCII.

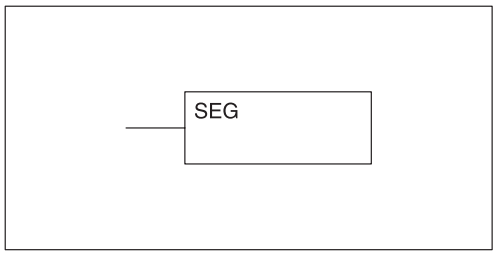
Возможные значения ASCII для преобразования ASCII в HEX

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

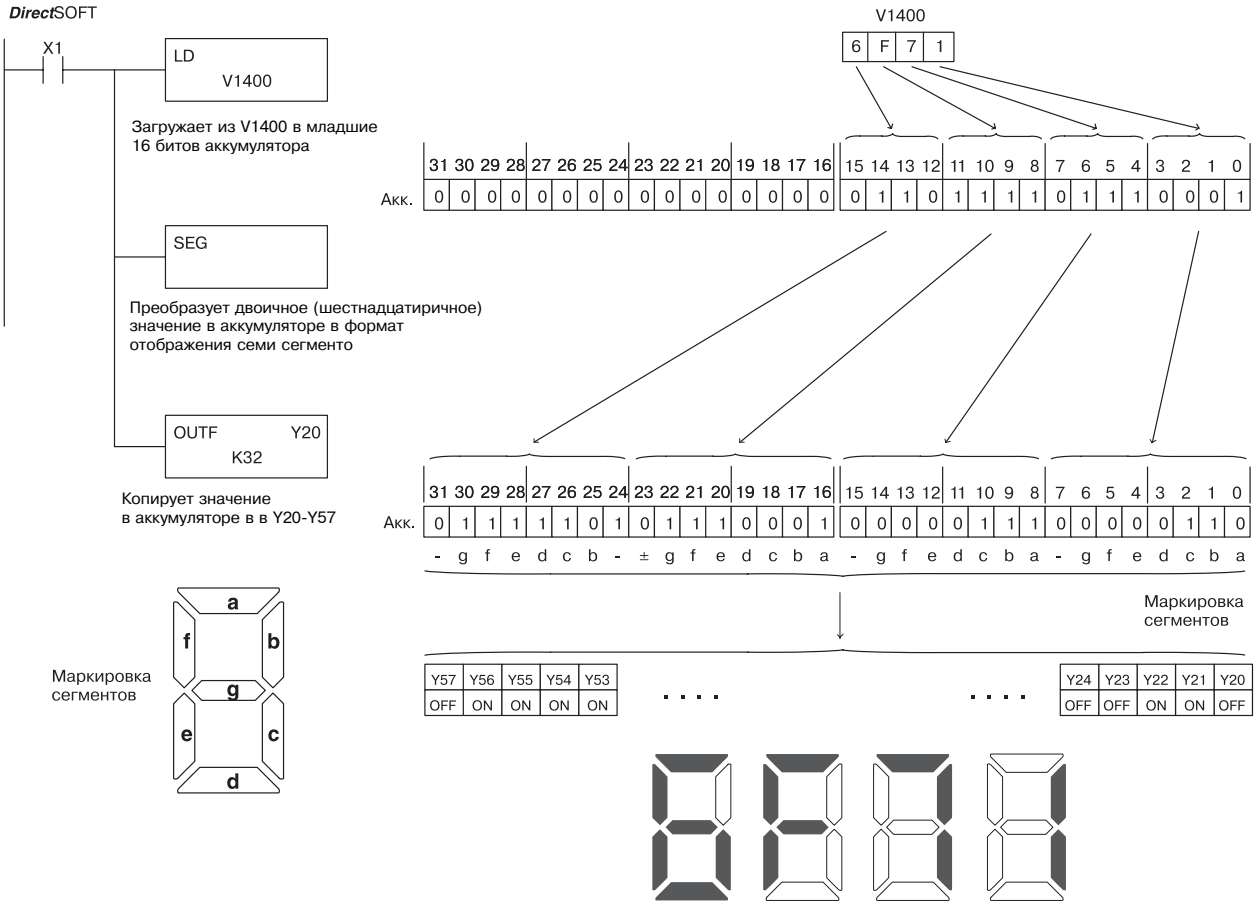
Segment (SEG)

√	√	√
430	440	450

Команда BCD/Segment преобразует 4-знаковое шестнадцатиричное значение в аккумуляторе в формат отображения из семи сегментов. Результат находится в аккумуляторе.



В следующем примере, когда X1 включен, значение в V1400 загружается в младшие 16 бит аккумулятора командой Load. Двоичное (шестнадцатиричное) значение в аккумуляторе преобразуется в формат из семи сегментов командой Segment. Набор битов в аккумуляторе копируется в Y20-Y57 командой Out Formatted.

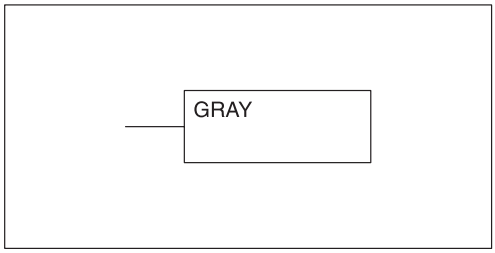


Набор на ручном программаторе

STR	X(IN)	1	↵							
LD	SHFT	V	1	4	0	0	↵			
SHFT	S	E	G	↵						
OUT	SHFT	F	SHFT	Y(OUT)	2	0	K(CON)	3	2	↵

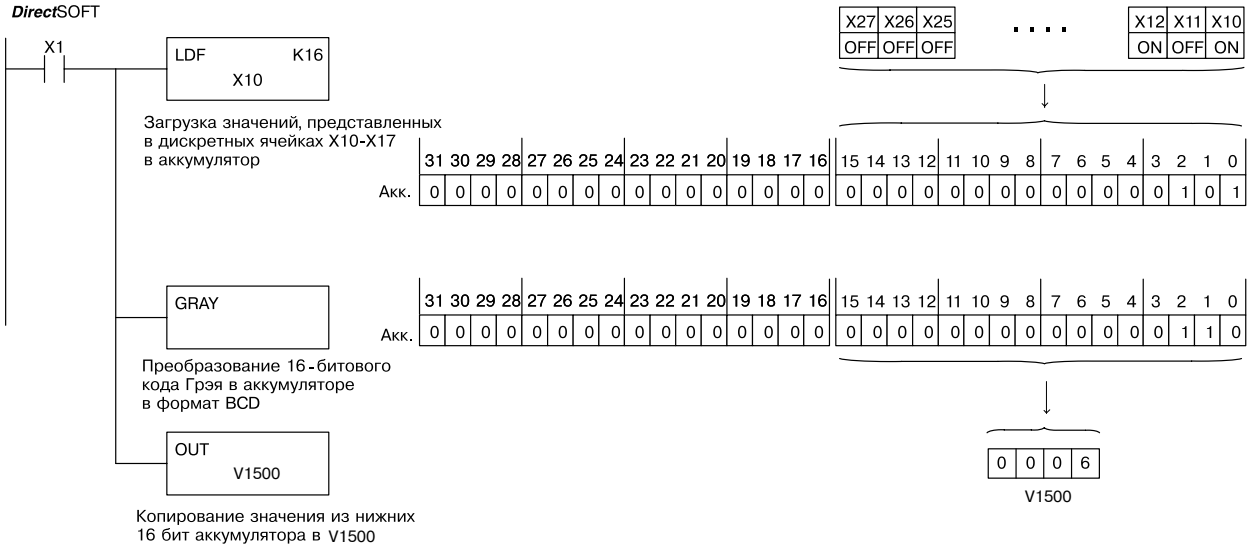
Gray Code (GRAY)

X	√	√
430	440	450



Команда Gray Code преобразует 16-битовое значение в коде Грея (двоичный циклический код) в двоично-десятичное значение. Двоично-десятичное преобразование требует 10 бит аккумулятора. Старшие 22 бита установлены в ”0”. Эта команда создана для использования с устройствами (обычно с кодировщиками), которые используют код Грея для схем нумерации. Команда Gray Code непосредственно преобразует число в коде Грея в двоично-десятичное число для устройств, имеющих разрешение 512 или 1024 точек на один оборот. Чтобы использовать устройство, имеющее разрешение 360 точек на один оборот, Вы должны вычесть двоично-десятичное значение 76 из преобразованного значения, чтобы получить соответствующий результат. Для устройства, имеющего разрешение 720 точек на один оборот, Вы должны вычесть двоично-десятичное значение 152.

В следующем примере, когда X1 включен, двоичное значение, представленное X10-X27 загружается в аккумулятор командой Load Formatted. Значение в коде Грея в аккумуляторе преобразуется в двоично-десятичное командой Gray Code. Значение в младших 16 битах аккумулятора копируется в V1500 .



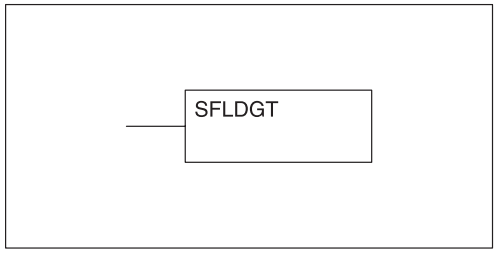
Набор на ручном программаторе

STR	X(IN)	1	↵							
LD	SHFT	F	SHFT	X(IN)	1	0	K(CON)	1	6	↵
SHFT	G	R	A	Y	SHFT	↵				
OUT	V	1	5	0	0	↵				

Код Грэя	BCD
0000000000	0000
0000000001	0001
000000001	0002
0000000010	0003
000000010	0004
000000011	0005
0000000101	0006
0000000100	0007
.	.
.	.
.	.
.	.
1000000001	1022
1000000000	1023

Shuffle Digits (SFLDGT)

X	√	√
430	440	450



Команда Shuffle Digits перетасовывает (меняет местами) максимум 8 цифровых знаков, располагая их в заданном порядке. Эта функция требует загрузки параметров в первый уровень стека аккумулятора и в аккумулятор двумя дополнительными командами. Ниже показаны необходимые шаги для использования функции Shuffle Digits. Пример на следующей странице показывает программу для функции Shuffle Digits.

Шаг 1: Загрузить перемешиваемые значения (цифры) в первый уровень стека аккумулятора.

Шаг 2: Загрузить в аккумулятор порядок цифр с нумерацией от 1 до 8 (”1” - это наименьшая значащая цифра, а ”8” наибольшая значащая цифра).

Замечание: Если число, используемое для определения порядка, содержит 0 или 9-F, то соответствующий разряд должен быть установлен в 0.

Замечание: Если число, используемое для определения порядка, содержит идентичные числа, то действует самое старшее из них. Результат находится в аккумуляторе. См. пример.

Шаг 3: Вставить команду Shuffle Digits.

Блок-схема команды Shuffle Digits

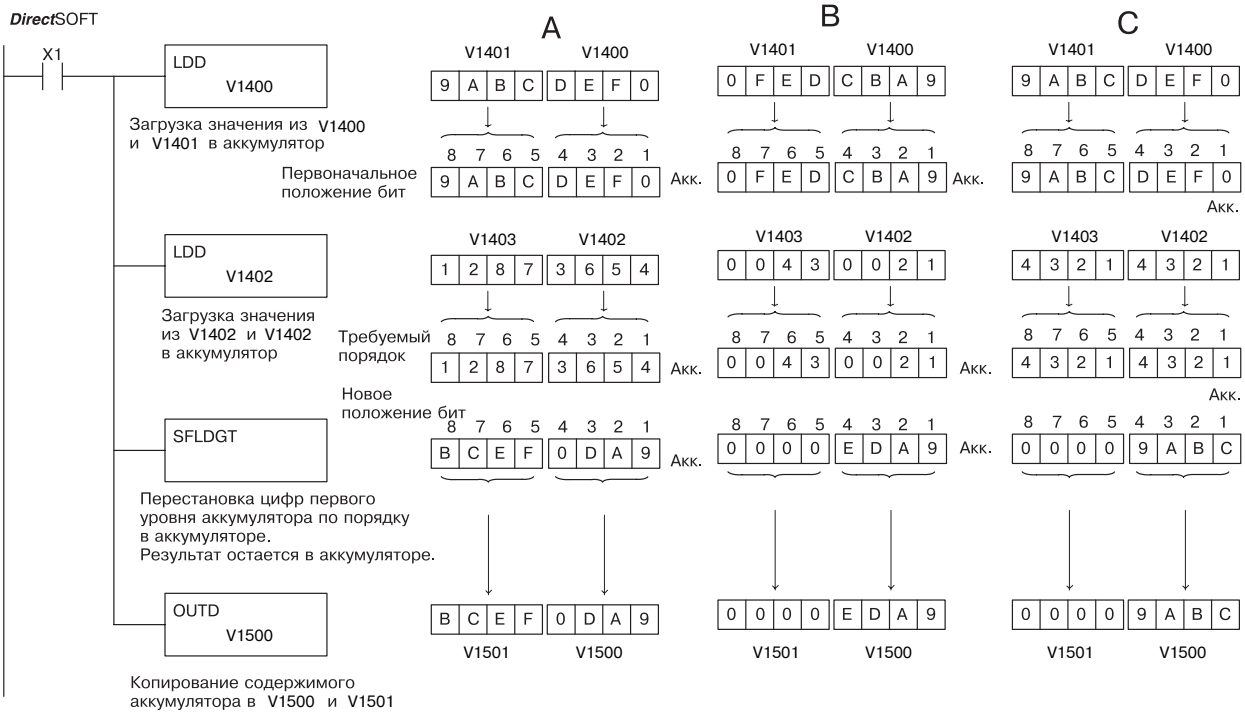
Могут быть переставлены максимум 8 цифр. Битовые разряды в первом уровне стека аккумулятора определяют цифры, которые будут переставлены. Они соответствуют битовым разрядам в аккумуляторе, которые определяют порядок, по которому будут перетасованы цифры. Цифры переставляются и результат остается в аккумуляторе

В следующем примере, когда X1 включен, значение в первом уровне стека аккумулятора будет реорганизовано в порядке, определенном значением в аккумуляторе.

Пример A показывает, как Shuffle Digits работает, когда 0 или 9-F не используются при определении порядка перестановки цифр, а также отсутствуют одинаковые номера при определении порядка.

Пример B показывает, как Shuffle Digits работает, когда 0 или 9-F используются при определении порядка перестановки цифр. Обращаем внимание на то, что при выполнении команды Shuffle Digits битовые разряды в первой ячейке стека, которые имели соответственно 0 или 9-F в аккумуляторе (в заданном порядке), устанавливаются в ”0”.

Пример C показывает, как Shuffle Digits работает, когда используются одинаковые номера при задании порядка перестановки цифр. Обращаем внимание на то, что при выполнении команды Shuffle при наличии дублированных номеров для задания порядка используется самый старший номер.



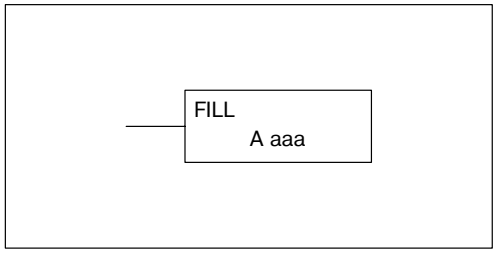
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
LD	SHFT	D	SHFT	V	1	4	0	2	↵
SHFT	S	F	L	D	G	T	↵		
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Табличные команды

Fill
(FILL)

√	√	√
430	440	450



Команда Fill заполняет таблицу вплоть до 255 ячеек V - памяти значением (Aaaa), которое является ячейкой V-памяти или константой из 4 цифр. Функциональные параметры загружаются в первый уровень стека аккумулятора и в аккумулятор двумя дополнительными командами. Ниже перечислены шаги, необходимые для программирования функции Fill.

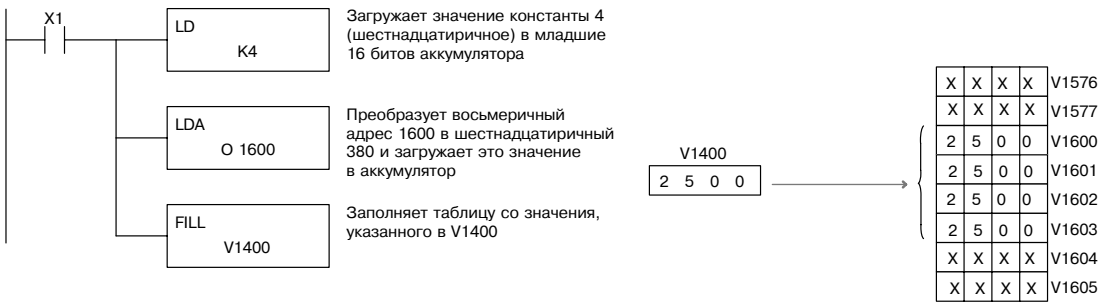
Шаг 1: - Загружается число ячеек V-памяти, которые надо заполнить, в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным значением.
Шаг 2: - Загружает стартовый адрес V-памяти для таблицы в аккумулятор. Этот параметр должен быть шестнадцатиричным значением.
Шаг 3: - Вставляет команду Fill, которая определяет значение для заполнения таблицы

Полезная Подсказка: Для параметров, которые требуют шестнадцатиричных значений при ссылке на ячейки памяти, для преобразования восьмеричного адреса в шестнадцатиричный эквивалент и загрузки значения в аккумулятор может использоваться команда LDA.

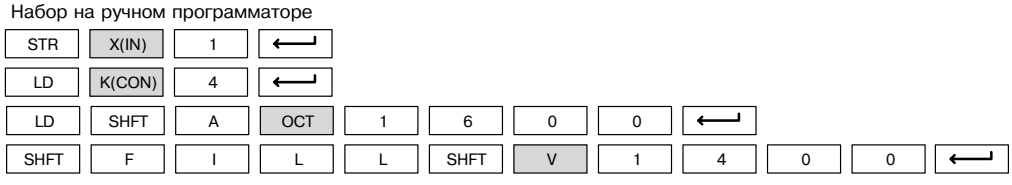
Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	0-FF	0-FF	0-FF

В следующем примере, когда X1 включен, константа (K4) загружается в аккумулятор командой Load. Это значение определяет длину таблицы и помещается в первый уровень стека аккумулятора при выполнении команды Load Address. Восьмеричный адрес 1600 (V1600) - стартовый адрес для таблицы, загружается в аккумулятор с использованием команды Load Address. Значение, с которого заполняется таблица (V1400), определено в команде Fill.

*Direct*SOFT



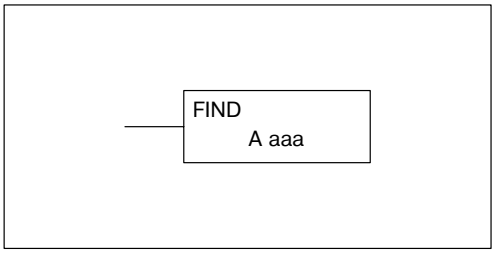
Набор на ручном программаторе



Find (FIND)

X	√	√
430	440	450

Команда Find используется для поиска заданной величины в таблице V-памяти, включающей до 255 ячеек. Функциональные параметры загружаются в первый и второй уровни стека аккумулятора и аккумулятор тремя дополнительными командами. Ниже перечислены шаги, необходимые для программирования функции Find.



Шаг 1: - Загрузить длину таблицы (число ячеек V-памяти) во второй уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным значением.
Шаг 2: - Загрузить стартовый адрес V - памяти для таблицы в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным значением.
Шаг 3: - Загрузить смещение от стартового адреса для начала поиска. Этот параметр должен быть шестнадцатиричным значением.
Шаг 4: - Вставить команду Find, которая определяет первое значение, которое будет найдено в таблице.
Результат: - В аккумулятор будет возвращено смещение от стартового адреса до первой ячейки V-памяти, которая содержит значение поиска. SP53 будет установлен в состояние ON (ВКЛ.), если адрес смещения определен вне таблицы, или если значение не найдено. Если значение не найдено, в аккумулятор будет возвращен 0.
Полезная Подсказка: - Для параметров, которые требуют шестнадцатиричное значение при ссылке на ячейки памяти, для преобразования восьмеричного адреса в шестнадцатиричный эквивалент и загрузки этого значения в аккумулятор может использоваться команда LDA.

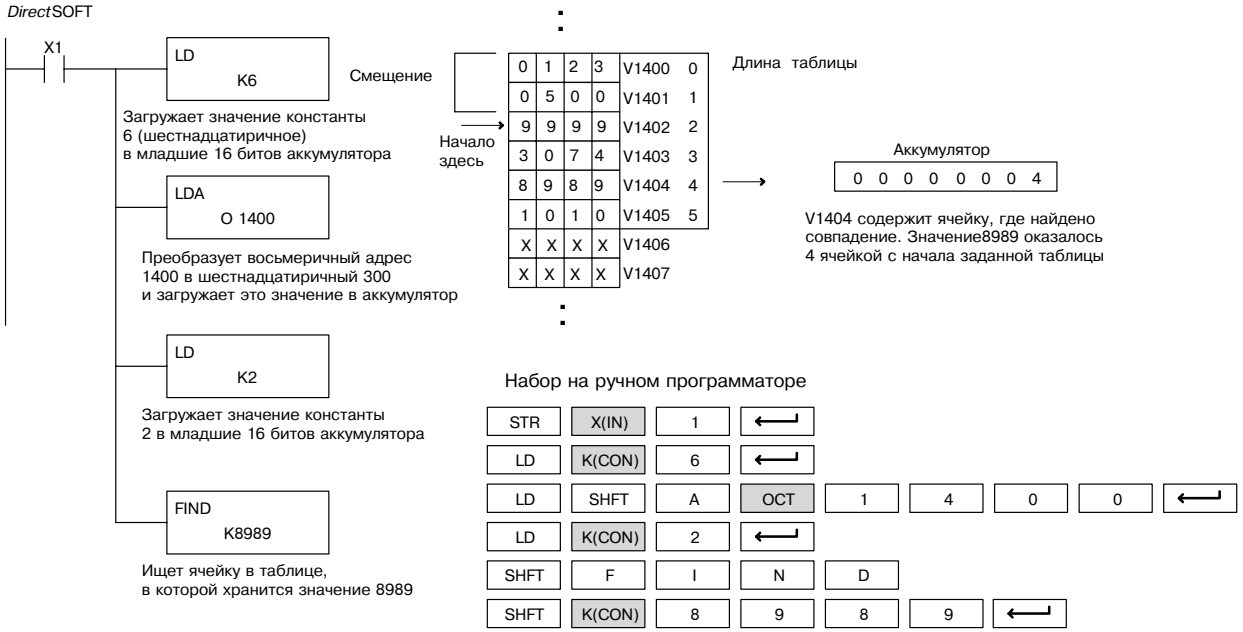
Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	0-FFFF	0-FFFF

Флаги дискретных разрядов	Описание
SP53	Включен, когда значение операнда больше, чем значение, с которым может работать аккумулятор

ПРИМЕЧАНИЕ: Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

Указатель для этой команды начинается с 0 и помещается в аккумулятор

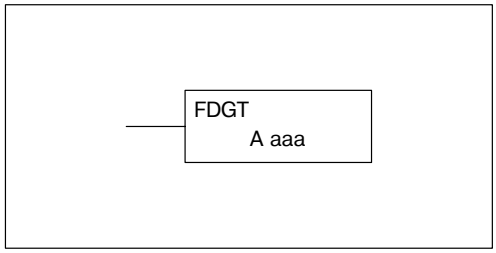
В следующем примере, когда X1 включен, константа (K6) загружается в аккумулятор командой Load. Это значение определяет длину таблицы и помещается во вторую ячейку стека, когда будут выполнены следующие команды Load Address и Load. Восьмеричный адрес 1400 (V1400) - стартовый адрес для таблицы, и он загружается в аккумулятор. Это значение помещается в первый уровень стека аккумулятора когда будет выполнена следующая команда Load. Смещение (K2) загружается в младшие 16 битов аккумулятора, используя команду Load. Значение, которое необходимо найти в таблице, задано в команде Find. Если найдено значение, равное значению поиска, смещение (от стартового адреса таблицы до ячейки, где размещено найденное значение) находится в аккумуляторе.



Find Greater Than (FDGT)

X	√	√
430	440	450

Команда Find Greater Than используется для поиска первого появления в таблице V-памяти значения, большего, чем заданная величина (Аааа). Заданная величина может быть или ячейкой V-памяти или константой из 4 цифр. Функциональные параметры загружаются в первый уровень стека аккумулятора и аккумулятор двумя дополнительными командами. Ниже перечислены шаги, необходимые для программирования функция Find Greater Than.



ПРИМЕЧАНИЕ: Эта команда не имеет смещения, которое требуется, например, для команды FIND.

Шаг 1: - Загрузить длину таблицы в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным значением.

Шаг 2: - Загрузить в аккумулятор стартовый адрес V-памяти для таблицы. Этот параметр должен быть шестнадцатиричным значением.

Шаг 3: - Вставить команду FDGT, которая определяет значение, большее, чем значение поиска.

Результат: - смещение от стартового адреса до первой ячейки V-памяти, которая содержит значение, большее, чем значение поиска, возвращается аккумулятору. SP53 будет установлен в положение ВКЛ., если значение не найдено, а в аккумуляторе будет 0.

Полезная Подсказка: - Для параметров, которые требуют шестнадцатиричного значения при ссылке на ячейки памяти, для преобразования восьмеричного адреса в шестнадцатиричный эквивалент и для загрузки значения в аккумулятор может использоваться команда LDA.

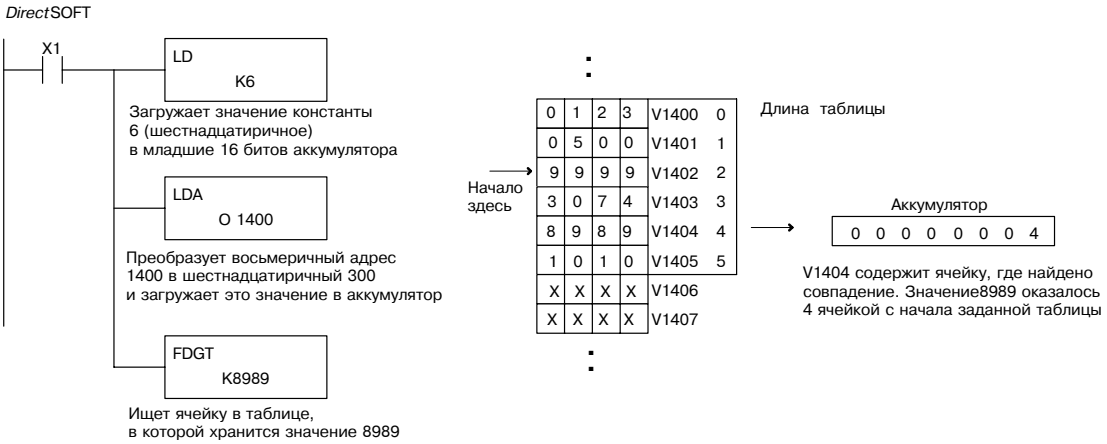
Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	0-FFFF	0-FFFF

Флаги дискретных разрядов	Описание
SP53	Включен, когда значение операнда больше, чем значение, с которым может работать аккумулятор

ПРИМЕЧАНИЕ: **Флаги состояния действительны только до того момента,** когда будет выполнена другая команда, использующая те же самые флаги.

Указатель для этой команды начинается с 0 и помещается в аккумулятор

В следующем примере, когда X1 включен, константа (K6) загружается в аккумулятор командой Load. Это значение определяет длину таблицы и помещается в первую ячейку стека после выполнения команды Load Address. Восьмеричный адрес 1400 (V1400) - это стартовый адрес таблицы, и он загружается в аккумулятор. Определяется значение, большее, чем значение поиска, заданное в команде Find Greater Than. Если такое значение найдено, в аккумулятор записывается смещение (от стартового адреса таблицы до ячейки, где это значение находится). Если такое значение не найдено в таблице, в аккумулятор заносится нуль, а SP53 переводится в состоянии ВКЛ.



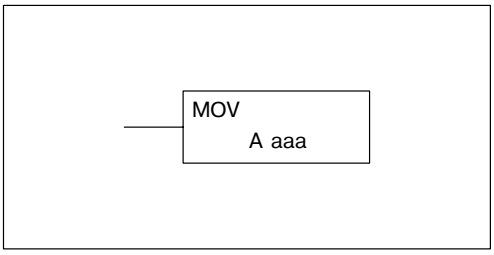
Набор на ручном программаторе

STR	X(IN)	1	↵								
LD	K(CON)	6	↵								
LD	SHFT	A	OCT	1	4	0	0	↵			
SHFT	F	D	G	T	SHFT	K(CON)	8	9	8	9	↵

Move (MOV)

X	√	√
430	440	450

Команда Move перемещает до 255 значений из одной таблицы V-памяти в другую таблицу V-памяти с такой же длиной. Функциональные параметры загружаются в первый уровень стека аккумулятора и в аккумулятор двумя дополнительными командами. Ниже показаны шаги, необходимые для программирования функции Move.



Шаг 1: Загрузить число перемещаемых ячеек V-памяти в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным значением.

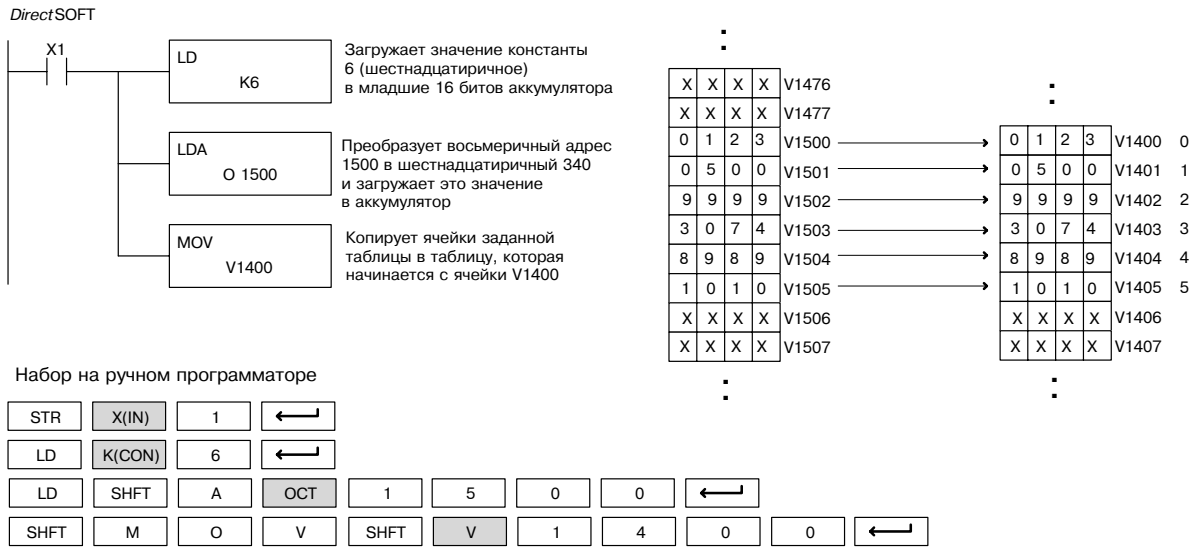
Шаг 2: Загрузить стартовую ячейку V-памяти для ячеек, перемещаемых в аккумулятор. Этот параметр должен быть шестнадцатиричным значением.

Шаг 3: Вставить команду Move, которая указывает стартовую ячейку V-памяти (Vaaa) для таблицы назначения.

Полезная подсказка: Для параметров, которые требуют шестнадцатиричных значений при ссылке на ячейки памяти, может быть использована команда LDA для преобразования восьмеричного адреса в шестнадцатиричный эквивалент и для загрузки значения в аккумулятор.

Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Указатель	P	Все (см. стр. 3-41)	Все (см. стр. 3-42)

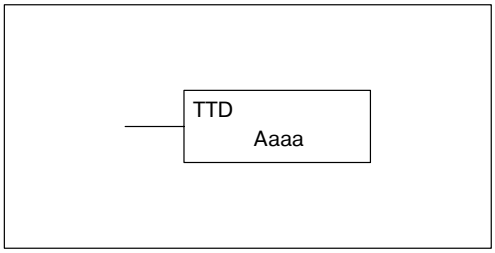
В следующем примере, когда X1 включен, константа (K6) загружается в аккумулятор командой Load. Это значение указывает длину таблицы и помещается в первой ячейке стека после выполнения команды Load Address. Восьмеричный адрес 1500 (V1500), который является стартовой ячейкой для исходной таблицы, загружается в аккумулятор. Ячейка таблицы назначения (V1400) указывается в команде Move.



Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	K(CON)	6	↵						
LD	SHFT	A	OCT	1	5	0	0	↵	
SHFT	M	O	V	SHFT	V	1	4	0	0	↵

Table To Destination (TTD)



Команда Table To Destination перемещает значение из таблицы V-памяти в ячейку V-памяти и увеличивает указатель таблицы на 1. Первая ячейка V-памяти в таблице содержит указатель на следующую ячейку в таблице, которую нужно переместить. Команда будет выполнена один раз в цикле сканирования, если ввод остается включенным. Указатель таблицы будет сброшен в 1, когда значение равняется последней ячейке в таблице. Функциональные параметры загружаются в первый уровень стека аккумулятора и аккумулятор двумя дополнительными командами. Ниже перечислены шаги, необходимые для программирования функция Table To Destination.

Шаг 1: - Загрузить длину таблицы (число ячеек V памяти) в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным значением.
Шаг 2: - Загрузить в аккумулятор стартовый адрес V-памяти для таблицы. (Помните, что стартовый адрес таблицы используется как указатель таблицы). Этот параметр должен быть шестнадцатиричным значением.
Шаг 3: - Вставить команду TTD, в которой задана ячейка V памяти адресата (Vaaa).
Полезная Подсказка: - Для параметров, которые требуют шестнадцатиричных значений при ссылке на ячейки памяти, для преобразования восьмеричного адреса в шестнадцатиричный эквивалент и загрузки значения в аккумулятор может использоваться команда LDA.
Полезная Подсказка: - Команда будет выполняться в каждом цикле сканирования, если включена входная логика. Если Вы не хотите, чтобы команда выполнилась для более чем одного цикла сканирования, во входной логике должен использоваться один однотактный режим (PD).
Полезная Подсказка: - Указатель должен быть установлен в значение, в котором необходимо начать операцию над таблицей. Необходимо использовать специальное реле SP0 или однотактный режим (PD), чтобы значение устанавливалось только в одном цикле сканирования и не влияло бы на работу команды.

Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42

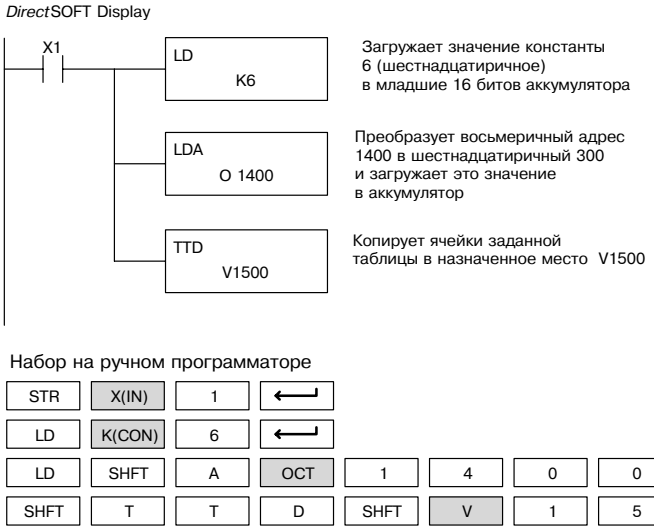
Флаги дискретных разрядов	Описание
SP56	Включен, когда указатель таблицы равен длине таблицы

X	√	√
430	440	450

ПРИМЕЧАНИЕ: Флаги состояния (SP) действительны только до:
- выполнения следующей команды, которая использует тот же самый флаг, или
- до конца цикла сканирования.

Указатель для этой команды начинается в 0 и сбрасывается, когда достигнута длина таблицы. На первый взгляд может показаться, что указатель должен быть сброшен в 0. Однако он сбрасывается в 1, а не в 0.

В следующем примере, когда X1 включен, константа (K6) загружается в аккумулятор командой Load. Это значение определяет длину таблицы и помещается в первую ячейку стека после выполнения команды Load Address. Восьмеричный адрес 1400 (V1400) (стартовый адрес для исходной таблицы) загружается в аккумулятор. Помните, V1400 используется как адрес указателя, а не часть источника данных таблицы. Адрес адресата (V1500) задан в команде Table to Destination. Указатель таблицы (в этом случае V1400) будет увеличиваться на "1" после каждого выполнения команды TTD.



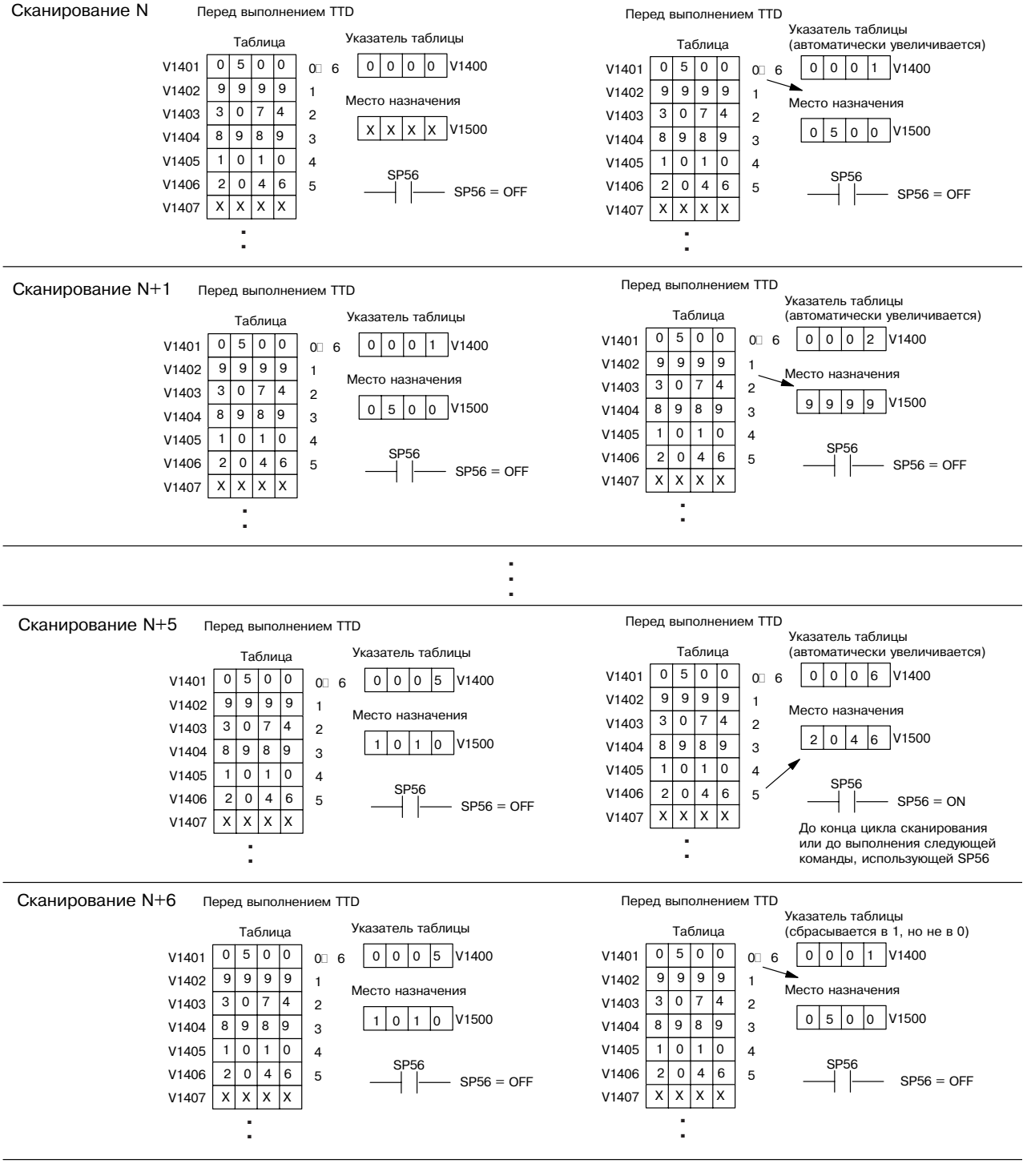
Важно понять, как пронумерованы ячейки таблицы. Если Вы исследуете пример таблицы, вы обратите внимание, что первая ячейка данных, V1401, будет использоваться, когда указатель равен нулю, а потом снова, когда указатель равен шести. Почему? Потому что указатель равен нулю только перед самым первым выполнением. В дальнейшем, он увеличивается от одного до шести, а затем сбрасывается к 1.

Наш пример использует также нормальный входной контакт (X1), чтобы управлять командой. Так как сканирование процессора чрезвычайно быстрый процесс, а приращение указателя происходит автоматически, цикл по ячейкам таблицы прошел бы очень быстро. Если это вызывает трудности, используйте вариант SP56 вместе с однотактным режимом (PD) и регистром-защелкой (например, C1), чтобы иметь возможность провести цикл по всем ячейкам таблицы один раз, затем остановиться. Никакой логики здесь не требуется, это - только вспомогательный метод.

На следующей схеме показаны результаты по циклам сканирования при выполнении программы нашего примера. Обратите внимание, что указатель автоматически циклически изменяется от 0 до 6, а затем начинается с 1 вместо 0. Также обратите внимание, что SP56 включен только до конца цикла сканирования.

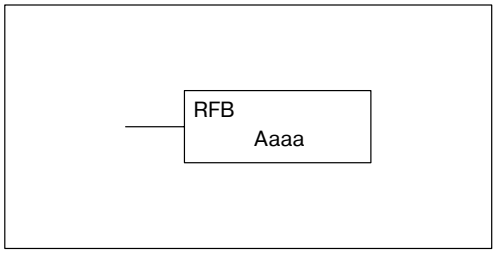
Пример выполнения

Пример выполнения



Remove From Bottom (RFB)

X	√	√
430	440	450



Команда Remove From Bottom перемещает значение из нижней части таблицы V-памяти в ячейку V - памяти и уменьшает указатель таблицы на 1. Первая ячейка V - памяти в таблице содержит указатель таблицы, который указывает следующую ячейку в таблице, которую нужно переместить. Команда выполняется один раз в цикле сканирования при условии, что ввод остается включенным. Команда прекратит работу, когда указатель будет равен 0. Функциональные параметры загружаются в первый уровень стека аккумулятора и аккумулятор двумя дополнительными командами. Ниже перечислены шаги, необходимые для программирования функция Remove From Bottom.

Шаг 1: - Загрузить в первый уровень стека аккумулятора длину таблицы (число ячеек V - памяти). Этот параметр должен быть шестнадцатиричным значением.

Шаг 2: - Загрузить в аккумулятор стартовый адрес V - памяти для таблицы. (Помните, что стартовый адрес таблицы используется как указатель таблицы.) Этот параметр должен быть шестнадцатиричным значением.

Шаг 3: - Вставить команду RFB, которая задает ячейку памяти назначения (Vaaa).

Полезная Подсказка: - Для параметров, которые требуют шестнадцатиричные значения при ссылке на ячейки памяти, для преобразования восьмеричного адреса в шестнадцатиричный эквивалент и для загрузки значения в аккумулятор может использоваться команда LDA.

Полезная Подсказка: - Команда будет выполняться в каждом цикле сканирования, если включена входная логика. Если Вы не хотите, чтобы команда выполнялась больше, чем в одном цикле сканирования, то во входной логике должен использоваться однотактный режим (PD).

Полезная Подсказка: - Указатель должен быть установлен в значение, в котором начинается работа с таблицей. Необходимо использовать специальное реле SP0 или однотактный режим (PD), чтобы значение устанавливалось только в одном цикле сканирования и не влияло бы на работу команды.

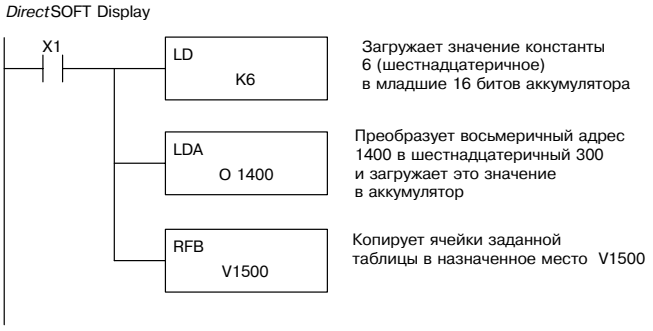
Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	aaa	aaa
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42

Флаги дискретных разрядов	Описание
SP56	Включен, когда указатель таблицы равен длине таблицы

ПРИМЕЧАНИЕ: Флаги состояния (SP) действительны только до:
- выполнения следующей команды, которая использует тот же самый флаг, или
- до конца цикла сканирования.

Указатель для этой команды можно устанавливать таким образом, чтобы начать операцию где-нибудь в таблице. Он не устанавливается автоматически. Вы должны загрузить значение в указатель где-нибудь в вашей программе.

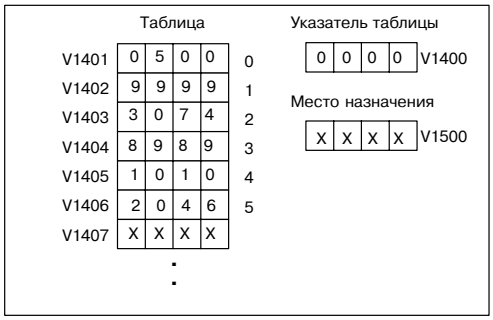
В следующем примере, когда X1 включен, константа (K6) загружается в аккумулятор командой Load. Это значение определяет длину таблицы и помещается в первую ячейку стека после выполнения команды Load Address. Восьмеричный адрес 1400 (V1400) (стартовый адрес для исходной таблицы) загружается в аккумулятор. Помните, V1400 используется как адрес указателя, а не как часть источника данных таблицы. Ячейка назначения (V1500) задается в команде Remove From Bottom. Указатель таблицы (в данном случае V1400) будет уменьшен на "1" после каждого выполнения команды RFB.



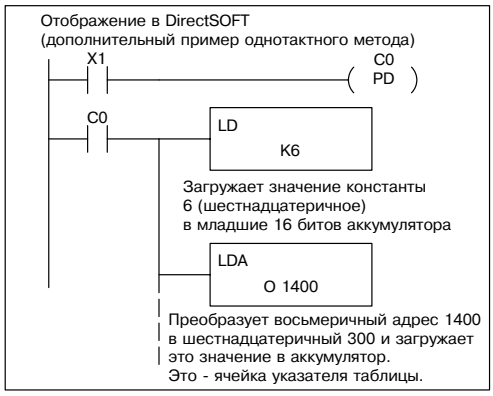
Набор на ручном программаторе

STR	X(IN)	1	↵						
LD	K(CON)	6	↵						
LD	SHFT	A	OCT	1	4	0	0	↵	
SHFT	R	F	B	SHFT	V	1	5	0	0	↵

Важно понять, как пронумерованы ячейки таблицы. Если Вы посмотрите пример таблицы, то обратите внимание, что первая ячейка данных, V1401, будет использоваться, когда указатель равен 1. Вторая ячейка данных, V1402, будет использоваться, когда указатель равен двум и т.д.

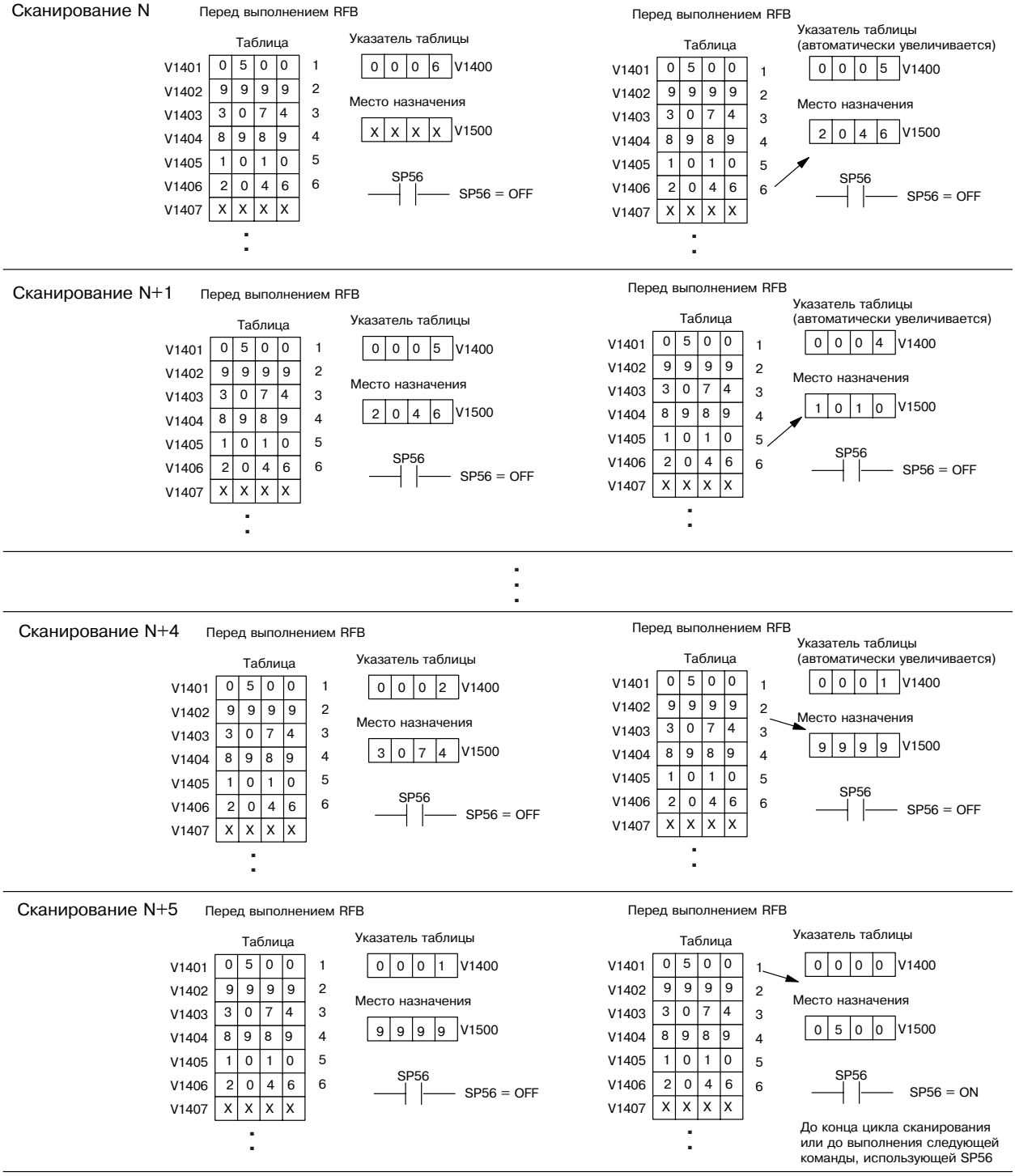


Наш пример использует также нормальный входной контакт (X1), чтобы управлять командой. Так как сканирование процессора чрезвычайно быстрый процесс, а уменьшение указателя происходит автоматически, цикл по ячейкам таблицы прошел бы очень быстро. Если это вызывает трудности, используйте вариант однотактного режима (PD), чтобы удалять одно значение каждый раз, когда входные контакты переходят от низкого уровня к высокому.



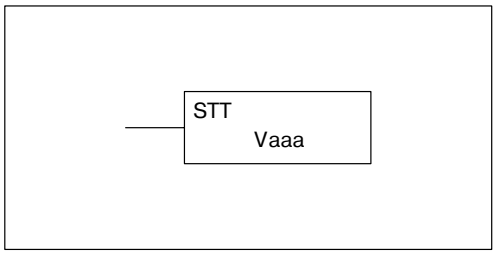
На следующей схеме показаны результаты по циклам сканирования при выполнении программы нашего примера. Обратите внимание, что указатель автоматически циклически уменьшается от 6 до 0. Также обратите внимание, что SP56 включен только до конца цикла сканирования.

Пример выполнения



Source to Table (STT)

X	√	√
430	440	450



Команда Source to Table перемещает значение из ячейки V - памяти в таблицу V - памяти и увеличивает указатель таблицы на 1. Когда указатель таблицы достигает конца таблицы, он сбрасывается в 1. Первая ячейка V - памяти в таблице содержит указатель таблицы, который указывает следующую ячейку в таблице для хранения значения. Команда выполняется один раз в цикле сканирования при условии, что ввод остается включенным. Функциональные параметры загружаются в первый уровень стека аккумулятора и аккумулятор двумя дополнительными командами. Ниже перечислены шаги, необходимые для программирования функции Source To Table.

Шаг 1: - Загрузить в первый уровень стека аккумулятора длину таблицы (число ячеек V - памяти). Этот параметр должен быть шестнадцатиричным значением.

Шаг 2: - Загрузить в аккумулятор стартовый адрес V-памяти для таблицы. (Помните, что стартовый адрес таблицы используется как указатель таблицы.) Этот параметр должен быть шестнадцатиричным значением.

Шаг 3: - Вставить команду STT, которая задает ячейку V-памяти источника (Vaaa). Это - ячейка, из которой будет перемещаться значение.

Полезная Подсказка: - Для параметров, которые требуют шестнадцатиричные значения при ссылке на ячейки памяти, для преобразования восьмеричного адреса в шестнадцатиричный эквивалент и для загрузки значения в аккумулятор может использоваться команда LDA.

Полезная Подсказка: - Команда будет выполнена в каждом цикле сканирования, если включена входная логика. Если Вы не хотите, чтобы команда выполнялась больше, чем в одном цикле сканирования, то во входной логике должен использоваться однотактный режим (PD).

Полезная Подсказка: - Должно быть установлено значение счетчика таблицы для указания начальной точки для операции. Он должен быть также установлен в значение, которое находится внутри таблицы. Например, если таблица имеет длину 6 слов, то допустимый диапазон значений указателя от 0 до 6. Если значение находится вне этого диапазона, то данные не будут перемещаться. Должен использоваться также однотактный режим (PD), чтобы значение устанавливалось только в одном цикле сканировании и не влияло на работу команды.

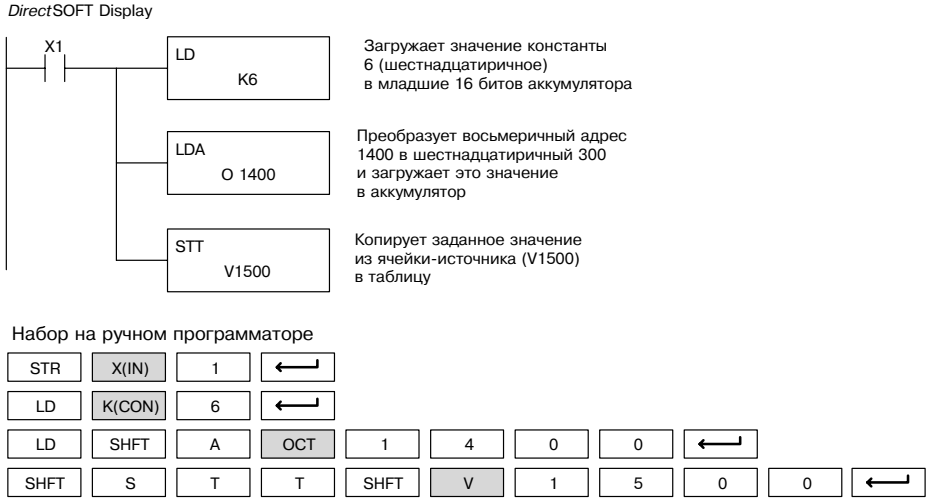
Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42

Флаги дискретных разрядов	Описание
SP56	Включен, когда указатель таблицы равен длине таблицы

ПРИМЕЧАНИЕ: Флаги состояния (SP) действительны только до:
- выполнения следующей команды, которая использует тот же самый флаг, или
- до конца цикла сканирования.

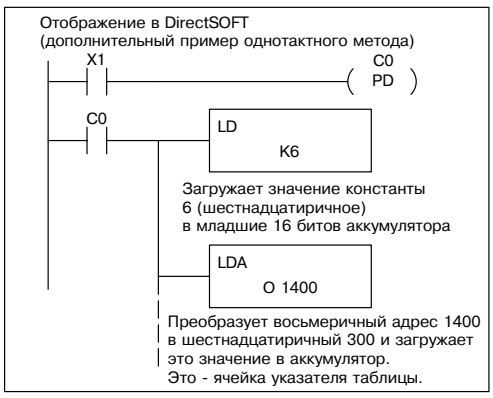
Указатель для этой команды начинается с 0 и сбрасывается в 1 автоматически, когда достигнута длина таблицы.

В следующем примере, когда X1 включен, константа (K6) загружается в аккумулятор командой Load. Это значение определяет длину таблицы и помещается в первую ячейку стека после выполнения команды Load Address. Восьмеричный адрес 1400 (V1400), который является начальной ячейкой для таблицы адресата и указателя таблицы, загружается в аккумулятор. Адрес источника данных (V1500) задается в команде Source to Table. Указатель таблицы будет увеличиваться на "1" каждый раз после выполнения команды.



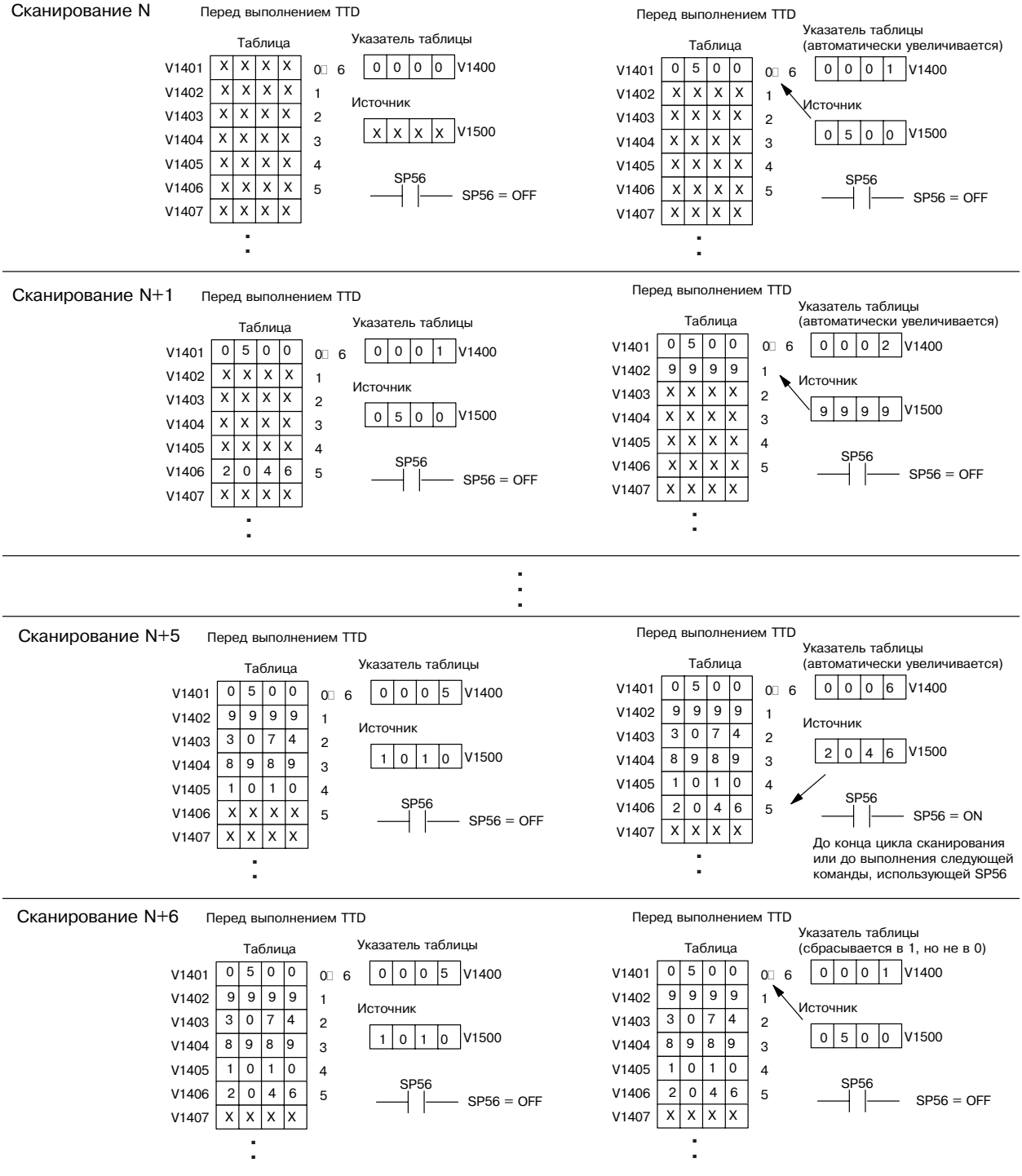
Важно понять, как пронумерованы ячейки таблицы. Если Вы посмотрите пример таблицы, то обратите внимание, что первая ячейка данных, V1401, будет использоваться, когда указатель равен нулю, и снова, когда указатель равен шести. Почему? Потому что указатель равен нулю только перед самим первым выполнением. В дальнейшем, он увеличивается от одного до шести, а затем сбрасывает в 1

Наш пример использует также нормальный входной контакт (X1), чтобы управлять командой. Так как сканирование процессора чрезвычайно быстрый процесс, а приращение указателя происходит автоматически, цикл по ячейкам таблицы прошел бы очень быстро. Если это вызывает трудности, используйте SP56 вместе с однотактным режимом (PD), чтобы перемещать одно значение каждый раз при переходе входных контактов от низкого уровня к высокому.



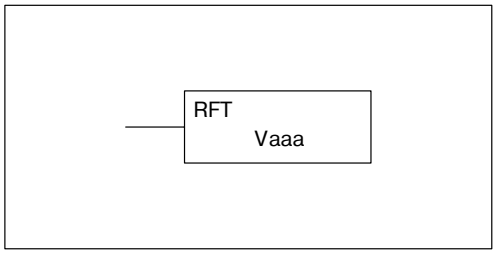
На следующей схеме показаны результаты по циклам сканирования при выполнении программы нашего примера. Обратите внимание на то, как указатель автоматически циклически повторяется от 0 до 6, а затем начинается с 1 вместо 0. Также обратите внимание, как выполнение влияет на SP56. Хотя наш пример не показывает этого, мы подразумеваем, что имеется другая часть программы, которая изменяет значение в V1500 (в источнике данных) до выполнения команды STT. Это не обязательно, но становится нагляднее, как источник данных копируется в таблицу.

Пример выполнения



Remove from Table (RTF)

X	√	√
430	440	450



Команда Remove from Table выталкивает значение из таблицы и сохраняет его в ячейке V - памяти. После удаления значения из таблицы, все остальные значения перемещаются вверх на 1 ячейку. Первая ячейка в таблице V - памяти содержит счетчик длины таблицы. При выполнении команды счетчик таблицы увеличивается каждый раз на 1. Если длина счетчика - нуль или больше максимальной длины таблицы (определенной в первом уровне стека аккумулятора), команда не будет выполнена, а SP56 будет включен.

Команда выполняется один раз в цикле сканирования при условии, что вход остается включенным. Функциональные параметры загружаются в первый уровень стека аккумулятора и в аккумулятор двумя дополнительными командами. Ниже перечислены шаги, необходимые для программирования функции Remove From Table.
Шаг 1: - Загрузить в первый уровень стека аккумулятора длину таблицы (число ячеек V - памяти). Этот параметр должен быть шестнадцатиричным значением.
Шаг 2: - Загрузить в аккумулятор стартовый адрес V-памяти для таблицы. (Помните, что стартовый адрес таблицы используется как указатель таблицы.) Этот параметр должен быть шестнадцатиричным значением.
Шаг 3: - Вставить команду RFT, которая задает ячейку V-памяти с назначением (Vaaa). Это ячейка, в которую значение будет перемещаться.
Полезная Подсказка: - Для параметров, которые требуют шестнадцатиричные значения при ссылке на ячейки памяти, для преобразования восьмеричного адреса в шестнадцатиричный эквивалент и для загрузки значения в аккумулятор может использоваться команда LDA.
Полезная Подсказка: - Команда выполняется в каждом цикле сканирования, если включена входная логика. Если Вы не хотите, чтобы команда выполнилась больше, чем в одном цикле сканирования, во входной логике должен использоваться однотактный режим (PD).
Полезная Подсказка: - Должно быть установлено значение счетчика таблицы для указания начальной точки для операции. Он должен быть также установлен в значение, которое находится внутри таблицы. Например, если таблица имеет длину 6 слов, то допустимый диапазон значений указателя от 0 до 6. Если значение находится вне этого диапазона, то данные не будут перемещаться. Должен использоваться также однотактный режим (PD), чтобы значение устанавливалось только в одном цикле сканировании и не влияло на работу команды.

Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42

Флаги дискретных разрядов	Описание
SP56	Включен, когда указатель таблицы равен длине таблицы

ПРИМЕЧАНИЕ: Флаги состояния (SP) действительны только до:
- выполнения следующей команды, которая использует тот же самый Флаг, или
- до конца цикла сканирования.
Указатель для этой команды можно устанавливать таким образом, чтобы начать операцию где-нибудь в таблице. Он не устанавливается автоматически. Вы должны загрузить значение в указатель где-нибудь в вашей программе.

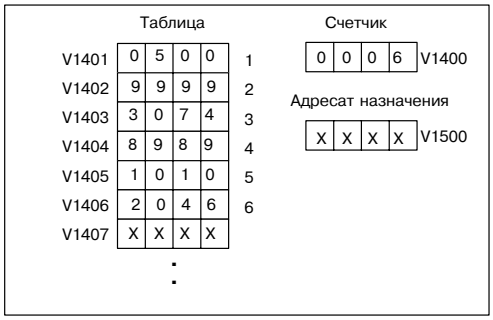
В следующем примере, когда X1 включен, константа (K6) загружается в аккумулятор командой Load. Это значение определяет длину таблицы и помещается в первую ячейку стека после выполнения команда Load Address. Восьмеричный адрес 1400 (V1400) - это стартовый адрес для исходной таблицы, он загружается в аккумулятор. Ячейка адресата (V1500) задается в команде Remove from Table. Счетчик таблицы будет уменьшаться на "1" после выполнения команды.

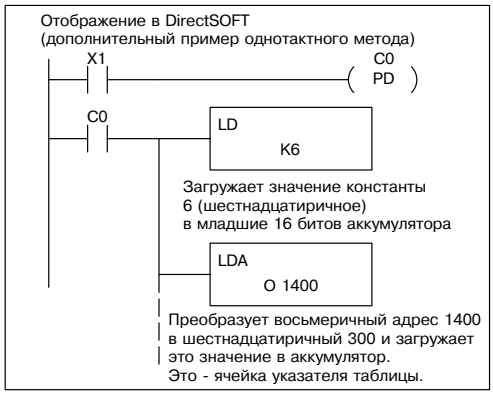
*Direct*SOFT Display

Набор на ручном программаторе

STR	X(IN)	1	↵							
LD	K(CON)	6	↵							
LD	SHFT	A	OCT	1	4	0	0	↵		
SHFT	R	F	T	SHFT	V	1	5	0	0	↵

Так как счетчик таблицы определяет диапазон данных, которые будут удалены из таблицы, важно понять, как пронумерованы ячейки таблицы. Если Вы рассмотрите таблицу примера, то обратите внимание на то, что ячейки данных нумеруются с верхней части таблицы. Например, если счетчик таблицы начинается с 6, то команда будет выполняться для всех шести ячеек.

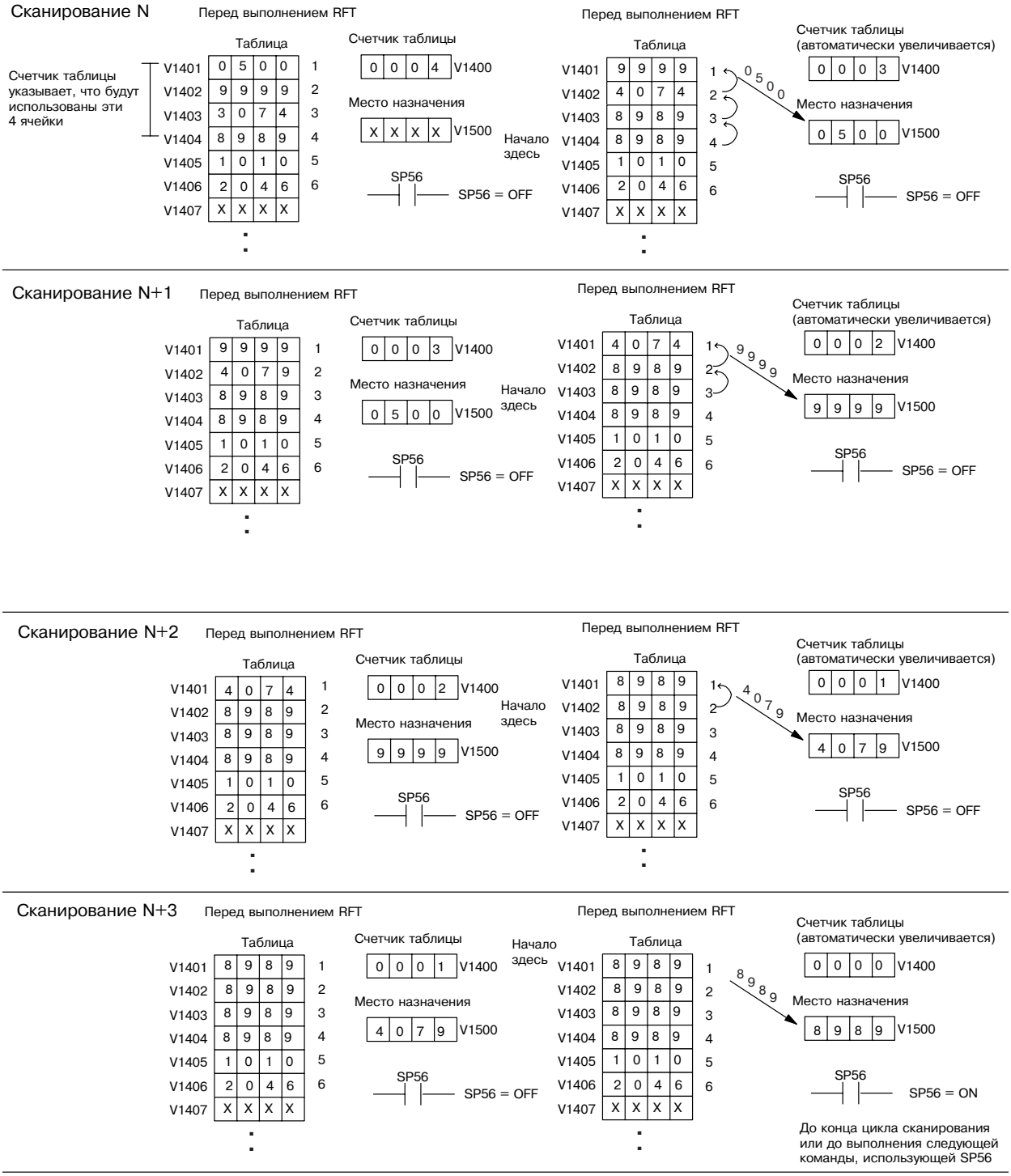




Наш пример использует также нормальный входной контакт (X1), чтобы управлять командой. Так как сканирование процессора чрезвычайно быстрый процесс, а уменьшение указателя происходит автоматически, данные были бы удалены из таблицы очень быстро. Если это вызывает трудности, используйте SP56 вместе с однотактным режимом (PD), чтобы перемещать одно значение каждый раз при переходе входных контактов от низкого уровня к высокому.

На следующей схеме показаны результаты по циклам сканирования при выполнении программы нашего примера. В этом примере мы показываем счетчик таблицы, установленный первоначально на 4. (Напоминаем, что Вы можете устанавливать счетчик таблицы в любое значение, которое будет лежать в диапазоне таблицы.) По мере выполнения команд счетчик таблицы автоматически уменьшается от 4 до 0. Обратите внимание, что последние две позиции таблицы, 5 и 6, не продвигаются вверх по таблице. Обратите также внимание на то, что SP56, который включается, когда счетчик таблицы равен нулю, остается включенным только до конца сканирования.

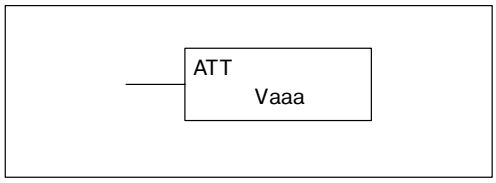
Пример выполнения



Add to Top (ATT)

X	√	√
430	440	450

Команда Add to Top помещает значение из ячейки V-памяти в таблицу V-памяти. Когда значение добавляется в таблицу, все остальные значения сдвигаются вниз на 1 ячейку.



Команда выполняется один раз в цикле сканирования при условии, что вход остается включенным. Функциональные параметры загружаются в первый уровень стека аккумулятора и в аккумулятор двумя дополнительными командами. Ниже перечислены шаги, необходимые для программирования функции Add to Top.

Шаг 1: - Загрузить в первый уровень стека аккумулятора длину таблицы (число ячеек V - памяти). Этот параметр должен быть шестнадцатиричным значением, от 0 до FF.

Шаг 2: - Загрузить в аккумулятор стартовый адрес V-памяти для таблицы. (Помните, что стартовый адрес таблицы используется как счетчик длины таблицы.) Этот параметр должен быть шестнадцатиричным значением.

Шаг 3: - Вставить команду ATT, которая задает исходную ячейку V-памяти (Vaaa). Это ячейка, из которой значение будет перемещаться.

Полезная Подсказка: - Для параметров, которые требуют шестнадцатиричные значения при ссылке на ячейки памяти, для преобразования восьмеричного адреса в шестнадцатиричный эквивалент и для загрузки значения в аккумулятор может использоваться команда LDA.

Полезная Подсказка: - Команда выполняется в каждом цикле сканирования, если включена входная логика. Если Вы не хотите, чтобы команда выполнилась больше, чем в одном цикле сканирования, во входной логике должен использоваться однотактный режим (PD).

Полезная Подсказка: - Должно быть установлено значение счетчика таблицы для указания начальной точки для операции. Он должен быть также установлен в значение, которое находится внутри таблицы. Например, если таблица имеет длину 6 слов, то допустимый диапазон значений, которые может иметь счетчик таблицы - от 1 до 6. Если значение находится вне этого диапазона или равно нулю, то данные в таблице не будут перемещаться. Должен использоваться также однотактный режим (PD), чтобы значение устанавливалось только в одном цикле сканировании и не влияло на работу команды

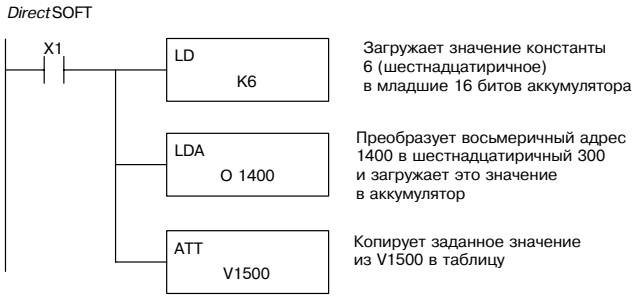
Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42

Флаги дискретных разрядов	Описание
SP56	Включен, когда указатель таблицы равен длине таблицы

ПРИМЕЧАНИЕ: Флаги состояния (SP) действительны только до:
- выполнения следующей команды, которая использует тот же самый флаг, или
- до конца цикла сканирования.

Указатель для этой команды можно устанавливать таким образом, чтобы начать операцию где-нибудь в таблице. Он не устанавливается автоматически. Вы должны загрузить значение в указатель где-нибудь в вашей программе.

В следующем примере, когда X1 включен, константа (K6) загружается в аккумулятор командой Load. Это значение определяет длину таблицы и помещается в первую ячейку стека после выполнения команды Load Address. Восьмеричный адрес 1400 (V1400), который является стартовым адресом для таблицы адресата и счетчика таблицы, загружается в аккумулятор. Исходный адрес (V1500) задан в команде Add to Top. Счетчик таблицы будет увеличиваться на "1" после выполнения команды.



Набор на ручном программаторе

STR	X(IN)	1	↵							
LD	K(CON)	6	↵							
LD	SHFT	A	OCT	1	4	0	0	↵		
SHFT	A	T	T	SHFT	V	1	5	0	0	↵

Для команды ATT счетчик таблицы определяет число добавлений, которые могут быть сделаны до окончания выполнения команды. Важно понять, как система использует этот счетчик, чтобы управлять выполнением.

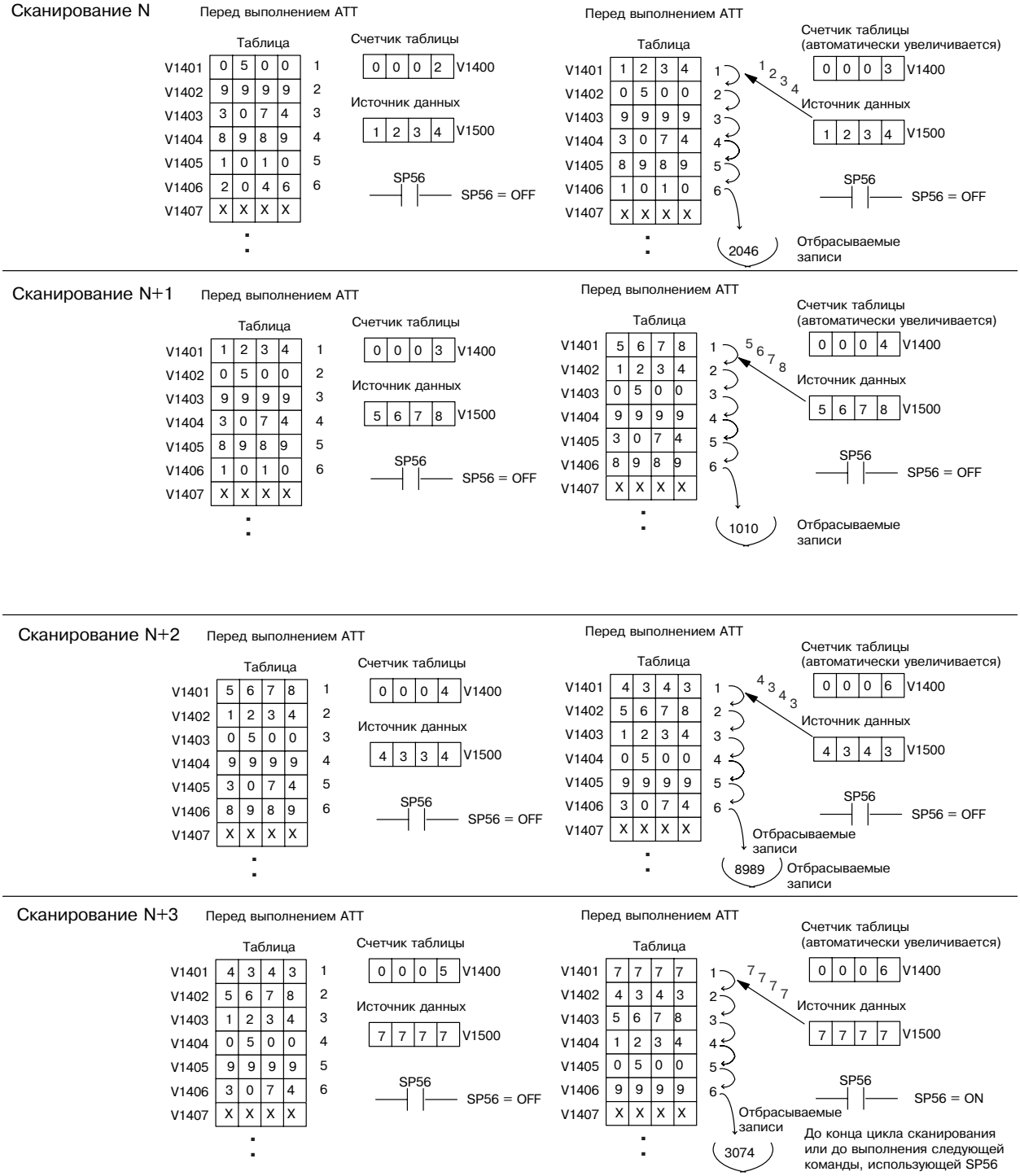
Например, если счетчик таблицы установлен равным 2, и длина таблицы составляет 6 слов, то может быть только 4 добавления данных до окончания выполнения. Это можно легко рассчитать:

Длина таблицы - счетчик таблицы = число циклов выполнения

В нашем примере используется также нормальный входной контакт (X1), чтобы управлять выполнением. Так как сканирование процессора чрезвычайно быстрый процесс, а приращение счетчика таблицы происходит автоматически, данные будут перемещаться в таблицу очень быстро. Если это вызывает трудности в вашем приложении, используйте вариант однотактного режима (PD), чтобы добавлять одно значение каждый раз при переходе входных контактов от низкого уровня к высокому.

На следующей схеме показаны результаты по циклам сканирования при выполнении программы нашего примера. Счетчик таблиц установлен первоначально в значение 2, и он автоматически увеличивается от 2 до 6, по мере выполнения команды. Обратите внимание, что SP56 включается, когда счетчик таблицы равен 6, т.е. равен длине таблицы. Кроме того, хотя в нашем примере это не показано, предполагается, что существует другая часть программы, которая изменяет значение в V1500 (источнике данных) до выполнения команды ATT.

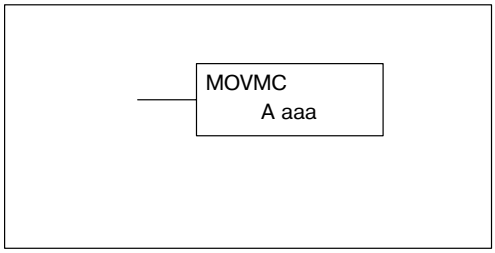
Пример выполнения



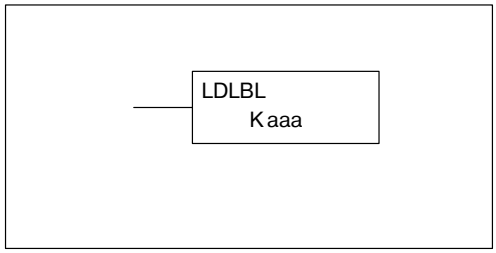
Move Memory Cartridge/ Load Label (MOVMC /LDLBL)

X	√	√
430	440	450

Команда Move Memory Cartridge используется для копирования данных между V - памятью и программной памятью. Команда Load Label используется только с командой MOVMC при копировании данных из программной памяти в V - память.



Для копирования данных между V-памятью и программной памятью функциональные параметры загружаются в первые два уровня стека аккумулятора и в аккумулятор двумя дополнительными командами. Ниже перечислены шаги, необходимые для программирования функций Move Memory Cartridge и Load Label.

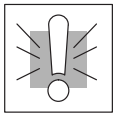


Шаг 1: - Загрузить число слов (максимум 255) для копирования во второй уровень стека аккумулятора Этот параметр должен быть шестнадцатиричным значением, от 0 до FF.
Шаг 2: - Загрузить смещение для области метки данных (в шестнадцатиричном формате) в программной памяти и начало блока V-памяти в первый уровень стека аккумулятора.
Шаг 3: - Загрузить исходную метку данных (LDLBL Kaaa) в аккумулятор для копирования данных из программной памяти в V память. Загрузить исходный адрес в аккумулятор, когда данные копируются из V-памяти в программную память. Это - адрес, откуда значение будет скопировано. Если исходный адрес - это ячейка V-памяти, то значение должно быть введено в шестнадцатеричном формате.
Шаг 4: - Вставить команду MOVMC, которая определяет адресат (Aaaa). Это адрес, куда значение будет скопировано.

Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	1-FFFF	1-FFFF

Флаги дискретных разрядов	Описание
SP53	Включен, когда указатель таблицы равен длине таблицы

ПРИМЕЧАНИЕ: Флаги состояния действительны только до:
- выполнения следующей команды, которая использует тот же самый флаг, или
- до конца цикла сканирования.



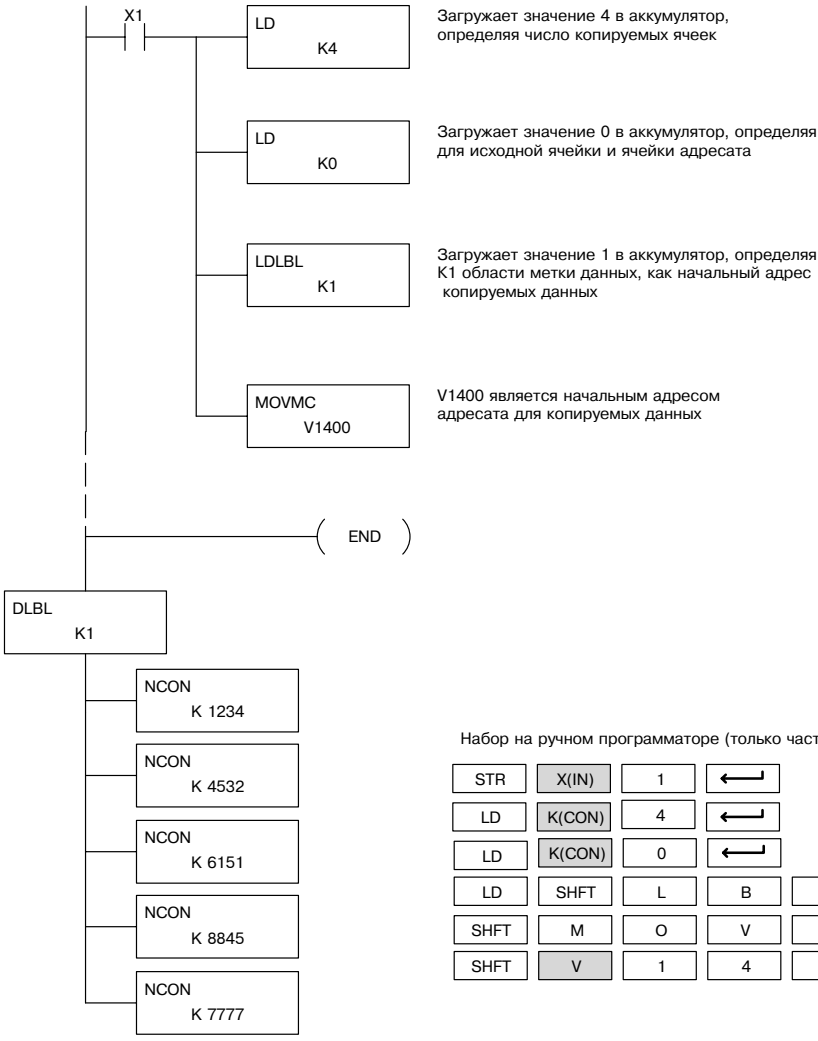
ПРЕДУПРЕЖДЕНИЕ: смещение для использования в этой команде начинается с 0, но можно использовать любое число, которое не приводит к тому, что данные будут вне исходной области данных, копируемых в таблицу адресата. Когда смещение - вне исходных информационных границ, в таблицу адресата будут перемещены неизвестные значения данных.

Copy Data From a Data Label Area to V Memory

X	√	√
430	440	450

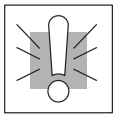
В следующем примере данные копируются из области метки данных (Data Label Area) в V-память. Когда X1 включен, константа (K4) загружается в аккумулятор с использованием команды Load. Это значение указывает длину таблицы назначения и помещается во второй ячейке стека после выполнения следующих команд Load и Load Label (LDLBL). Константа (K0) загружается в аккумулятор командой Load. Это значение указывает смещение для исходной таблицы и адресата, и помещается в первую ячейку стека после выполнения команды LDLBL. Исходный адрес, откуда копируются данные, загружается в аккумулятор командой LDLBL. Команда MOVMC указывает стартовую ячейку адресата и выполняет копирование данных из области метки данных в V-память.

*Direct*SOFT Display



Набор на ручном программаторе (только часть, относящаяся к MOVMC)

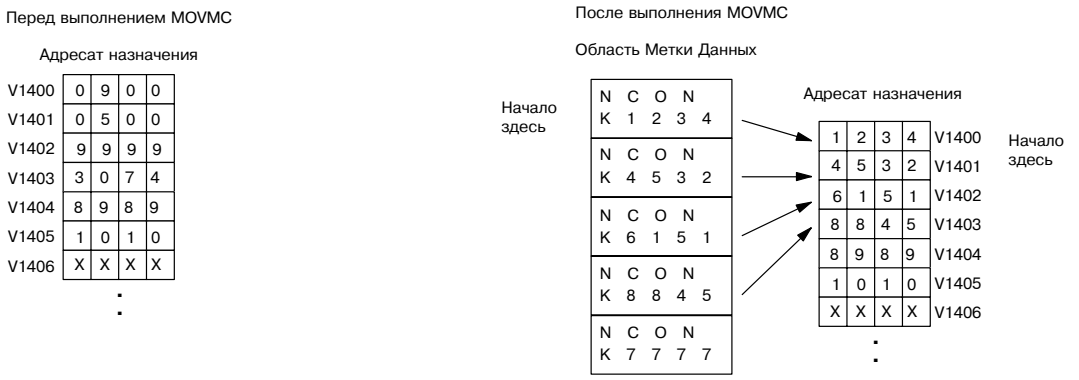
STR	X(IN)	1	↵					
LD	K(CON)	4	↵					
LD	K(CON)	0	↵					
LD	SHFT	L	B	L	SHFT	K(CON)	1	↵
SHFT	M	O	V	M	C			
SHFT	V	1	4	0	0	↵		



ПРЕДУПРЕЖДЕНИЕ: смещение для использования в этой команде начинается с 0, но можно использовать любое число, которое не приводит к тому, что данные будут вне исходной области данных, копируемых в таблицу адресата. Когда смещение находится вне исходных информационных границ, то в таблицу адресата будут перемещатся неизвестные значения данных.

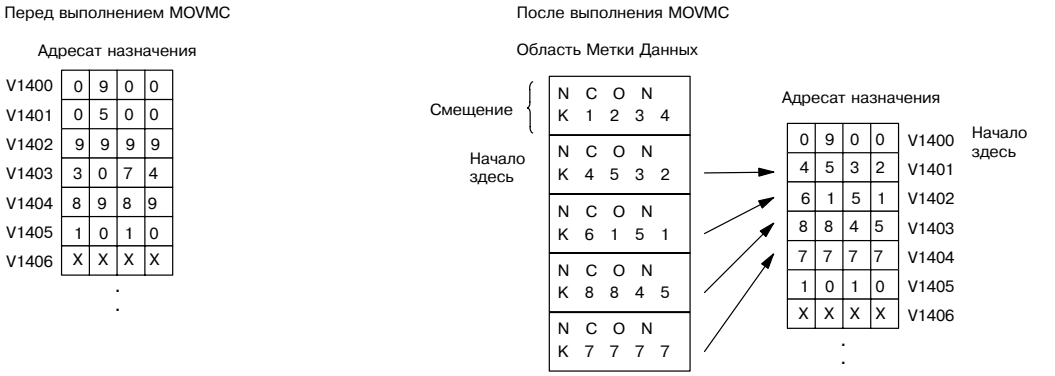
На следующей схеме показаны результаты нашего примера. Смещение равно нулю, и четыре слова будут скопированы в область метки данных.

Пример выполнения
Смещение = 0, перемещаются 4 слова

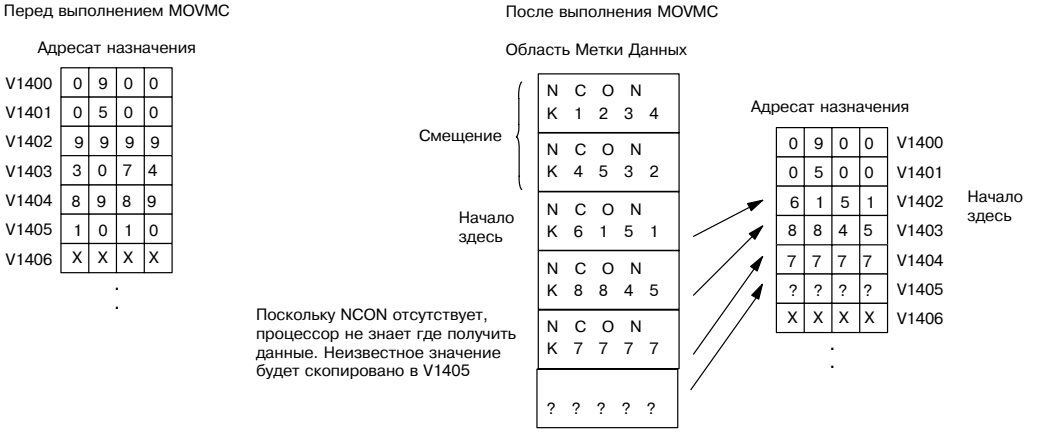


Это довольно простой пример, когда используется нулевое смещение. Однако, Вам необходимо понять результаты, которые были бы получены с другими значениями смещения (1 и 2). Обратите внимание, как используется смещение как для метки данных (источник), так и для таблицы адресата. Обратите внимание также на то, что неправильное смещение (в этом случае 2) может привести к неизвестным значениям, копируемым в таблицу адресата.

Смещение = 1, перемещается 4 слова



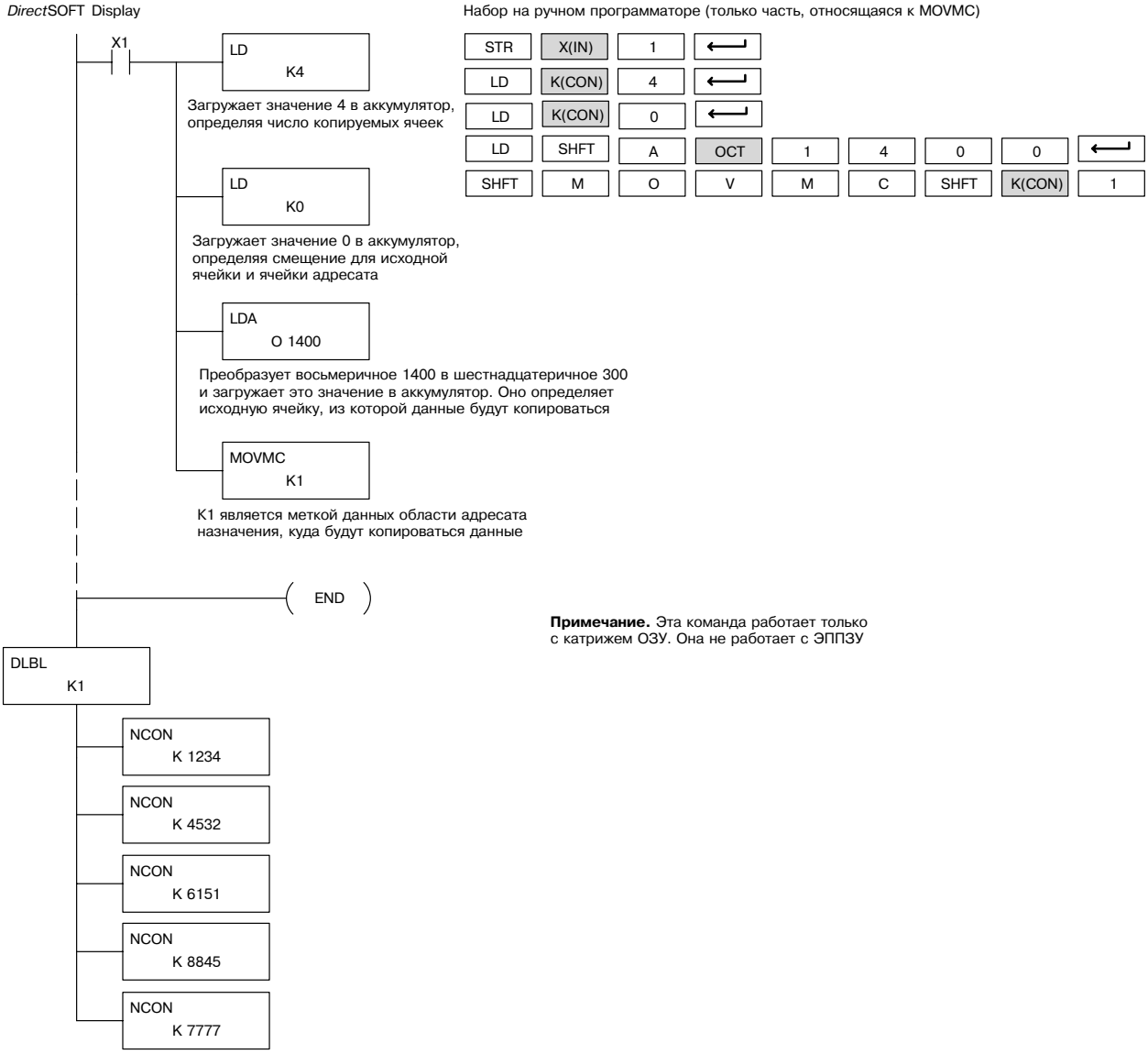
Смещение = 2, перемещается 4 слова



Copy Data From V Memory to a Data Label Area

√	√	√
430	440	450

В следующем примере данные копируются из V-памяти в область метки данных (Data Label Area). Когда X1 включен, константа (K4) загружается в аккумулятор командой Load. Это значение указывает длину таблицы и помещается во второй ячейке стека после выполнения последовательности команд Load и Load Address. Константа (K0) загружается в аккумулятор с использованием команды Load. Это значение указывает смещение для исходной таблицы и адресата, и помещается в первую ячейку стека после выполнения команды Load Address. Исходный адрес, откуда копируются данные, загружается в аккумулятор командой Load Address. Команда MOVMC указывает стартовую ячейку адресата и выполняет копирование данных из V-памяти в область метки данных.



Примечание. Эта команда работает только с катрижем ОЗУ. Она не работает с ЭППЗУ

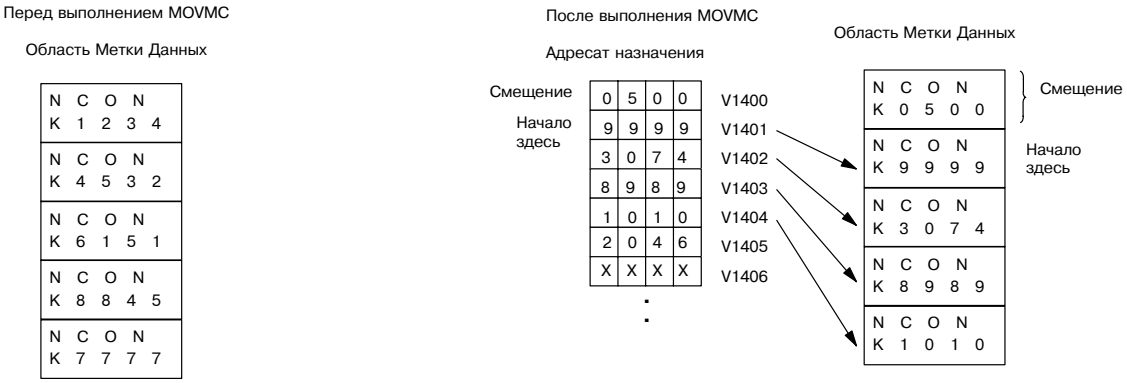
ПРЕДУПРЕЖДЕНИЕ: смещение для использования в этой команде начинается с 0, но можно использовать любое число, которое не приводит к тому, что данные будут вне исходной области данных, копируемых в таблицу адресата. Когда смещение находится вне исходных информационных границ, то в таблицу адресата будут перемещатся неизвестные значения данных.

На следующей схеме показаны результаты нашего примера. Смещение равно нулю, и четыре слова будут скопированы в область метки данных.

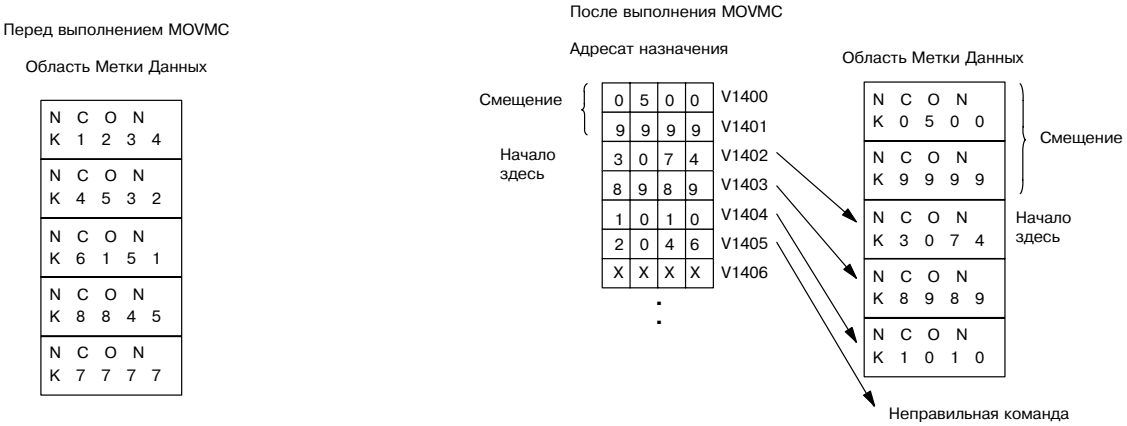
Пример выполнения
Смещение = 0, перемещаются 4 слова

Когда используется нулевое смещение, это довольно простой пример. Однако Вам необходимо понять результаты, которые были бы получены с другими значениями смещения (1 и 2). Обратите внимание на то, как используется смещение как для метки данных (источник), так и для таблицы адресата. Обратите внимание также на то, что неправильное смещение (в нашем случае 2) может привести к написанию неправильных команд, следующих после Метки Данных.

Смещение = 1, перемещаются 4 слова



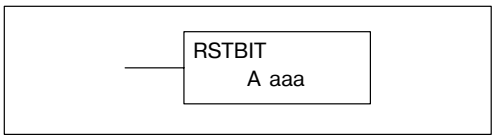
Смещение = 2, перемещаются 4 слова



Set Bit (SETBIT)

X	X	√
430	440	450

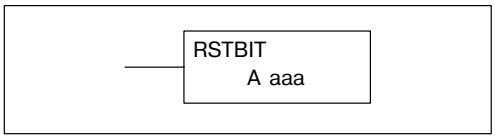
Команда Set Bit устанавливает один бит в значение ”1” в диапазоне ячеек V - памяти.



Reset Bit (RSTBIT)

X	X	√
430	440	450

Команда Reset Bit устанавливает один бит в значение ”0” в диапазоне ячеек V - памяти.



Следующее описание применимо к обеим табличным командам Set Bit и Reset Bit.

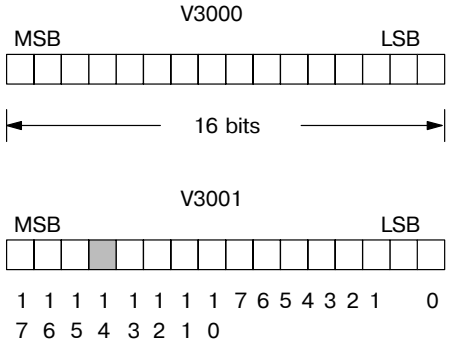
Шаг 1: - Загрузить в первый уровень стека аккумулятора длину таблицы (число ячеек V-памяти). Этот параметр должен быть шестнадцатиричным значением, от 0 до FF.
Шаг 2: - Загрузить в аккумулятор начальный адрес V-памяти для таблицы. Этот параметр должен быть шестнадцатиричным значением. Вы можете использовать команду LDA для преобразования восьмеричного адреса в шестнадцатиричный.
Шаг 3: - Вставить команду Set Bit или Reset Bit. В ней задается ссылка для номера бита, который Вы хотите установить или сбросить. Номер бита имеет восьмеричный формат, а первый номер в таблице имеет номер ”0”.
Полезная Подсказка: - Напоминаем, что каждая ячейка V - памяти содержит 16 бит. Поэтому биты первого слова таблицы нумеруются от 0 до 17 в восьмеричном формате. Например, если длина таблицы составляет 6 слов, то 6 слов = (6 x 16) битов, = 96 битов (десятичных) или 140 битов (восьмеричных). Допустимый диапазон ссылочных номеров на биты может быть от 0 до 137 восьмеричных. Флаг 53 устанавливается, если бит определен вне диапазона таблицы.

Тип данных операнда		Диапазон DL450
	A	**aaa**
V-память	V	Все (см. стр. 3-42)

Флаги дискретных разрядов	Описание
SP53	Включен, когда номер бита, используемый как ссылка в Set Bit или Reset Bit, выходит за пределы диапазона таблицы.

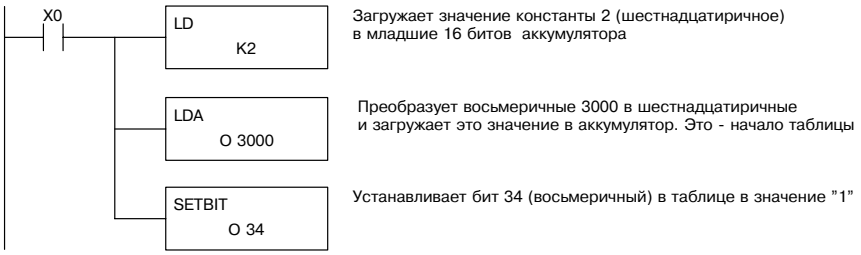
ПРИМЕЧАНИЕ: Флаги состояния действительны только до:
- выполнения следующей команды, которая использует тот же самый флаг, или
- до конца цикла сканирования.

Например, предположим, что мы имеем таблицу с началом в V3000 и длиной 2 слова, как показано на рисунке справа. Каждое слово в таблице содержит 16 битов, или от 0 до 17 в восьмеричном формате. Для установки бита 12 во втором слове мы используем его восьмеричную ссылку (бит 14). Затем мы вычисляем восьмеричный адрес бита от начала таблицы, 17 + 14 = 34 восьмеричных. В следующем примере программы показано, как установить бит в значение ”1” .



В следующем примере программы мы используем X0 для запуска операции Set Bit. Сначала мы загружает длину таблицы (2 слова) в стек аккумулятора. Затем мы загружаем в аккумулятор начальный адрес. Поскольку V3000 - это восьмеричный номер, мы должны преобразовать его в шестнадцатиричный с помощью команды LDA. Наконец, мы используем команду Set Bit (или Reset Bit) и определяем восьмеричный адрес бита (бит 34) при обращении к нему от начала таблицы.

*Direct*SOFT



Набор на ручном программаторе

STR	X(IN)	0	↵					
LD	K(CON)	2	↵					
LD	SHFT	A	OCT	3	0	0	0	↵
SET	SHFT	B	I	T	OCT	3	4	↵

Table Shift Left (TSHFL)

Команда Table Shift Left сдвигает все биты в таблице V - памяти влево на заданное число битовых разрядов.

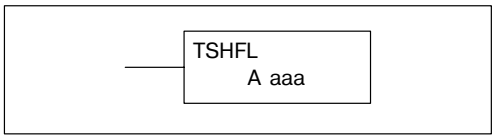
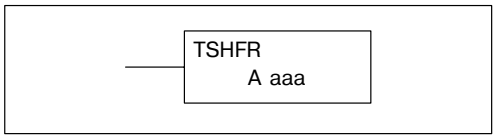


Table Shift Right (TSHFR)

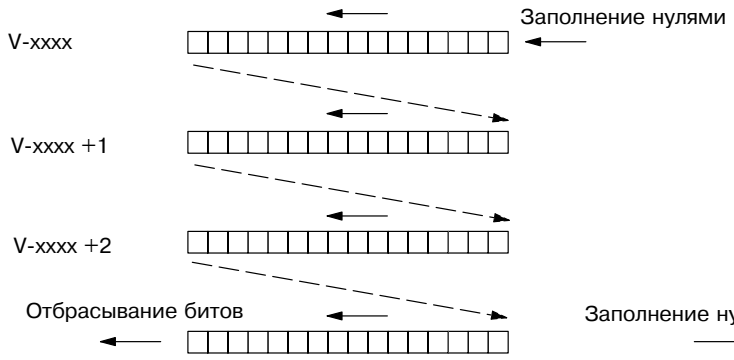
Команда Table Shift Right сдвигает все биты в таблице V - памяти вправо на заданное число битовых разрядов..



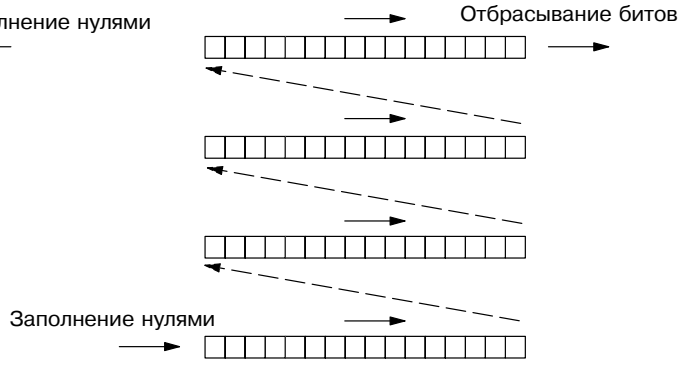
X	X	√
430	440	450

Следующее описание применимо к обеим табличным командам Table Shift Left и Table Shift Right. Таблицей является только некоторая область ячеек V - памяти. Команды Table Shift Left (Сдвиг таблицы влево) или Table Shift Right (Сдвиг таблицы вправо) сдвигают биты последовательно через всю таблицу. Биты сдвигаются к концу одного слова и к противоположному концу смежного слова. В конце таблицы биты либо сбрасываются, либо нули заносятся в таблицу. Ниже приведена произвольная таблица длиной 4 слова.

Сдвиг таблицы влево



Сдвиг таблицы вправо



Шаг 1: - Загрузить в первый уровень стека аккумулятора длину таблицы (число ячеек V-памяти). Этот параметр должен быть шестнадцатиричным значением, от 0 до FF.

Шаг 2: - Загрузить в аккумулятор начальный адрес V-памяти для таблицы. Этот параметр должен быть шестнадцатиричным значением. Вы можете использовать команду LDA для преобразования восьмеричного адреса в шестнадцатиричный.

Шаг 3: - Вставить команду Table Shift Left или Table Shift Right. В ней задается число битовых разрядов, на который Вы хотите сдвинуть всю таблицу. Число битовых разрядов должен иметь восьмеричный формат.

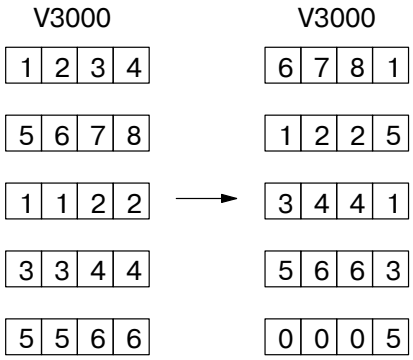
Полезная Подсказка: - Напоминаем, что каждая ячейка V-памяти содержит 16 битов. Поэтому биты первого слова таблицы нумеруются от 0 до 17 в восьмеричном формате. Если Вы хотите сдвинуть всю таблицу на 20 битов, это в восьмеричном формате - 24. Флаг 53 устанавливается, если число битов, на которое на которое делается сдвиг больше общего числа битов в таблице. Флаг 67 устанавливается, если последний сдвигаемый бит (непосредственно перед его отбрасыванием) равен ”1”.

Тип данных операнда		Диапазон DL450
	A	**aaa**
V-память	V	Все (см. стр. 3-42)

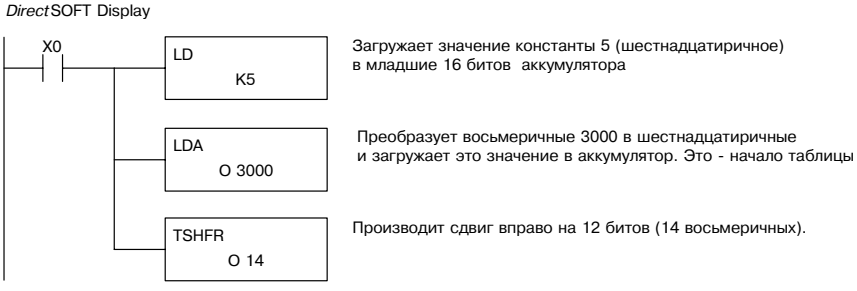
Флаги дискретных разрядов	Описание
SP53	Включен, когда номер бита, используемый как ссылка в Set Bit или Reset Bit, выходит за пределы диапазона таблицы.
SP67	Включен, когда последний сдвигаемый бит (непосредственно перед его отбрасыванием) равен "1".

ПРИМЕЧАНИЕ: Флаги состояния действительны только до:
- выполнения следующей команды, которая использует тот же самый флаг, или
- до конца цикла сканирования.

Справа показана таблица, содержащая двоично-десятичные данные (для демонстрационных целей) Предположим, что мы хотим сдвинуть таблицу вправо на 3 двоично-десятичные цифры (на 12 битов). Преобразуя 12 в восьмеричный формат получим 14 битов (восьмеричных). Используя команду Table Shift Right и определяя сдвиг на 14 восьмеричных, получим таблицу, показанную на рисунке правее. Отметим, что последовательность 2-3-4 отбрасывается, а последовательность 0-0-0 вдвигается в нижнюю часть таблицы.



В следующем примере программы предполагается, что данные в ячейках с V3000 по V3004 имеют указанные выше значения. Мы используем X0 для запуска операции Table Shift Right. Сначала мы загружает длину таблицы (5 слов) в стек аккумулятора. Затем мы загружаем в аккумулятор начальный адрес. Поскольку V3000 - это восьмеричный номер, мы должны преобразовать его в шестнадцатиричный с помощью команды LDA. Наконец, мы используем команду Table Shift Right и определяем число битов, на которое производится сдвиг (12 десятичных, 14 восьмеричных).



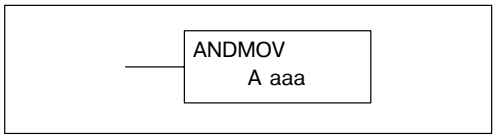
Набор на ручном программаторе

STR	X(IN)	0	↵					
LD	K(CON)	5	↵					
LD	SHFT	A	OCT	3	0	0	0	↵
SHFT	T	S	H	F	R	OCT	1	4	↵

AND Move (ANDMOV)

X	X	√
430	440	450

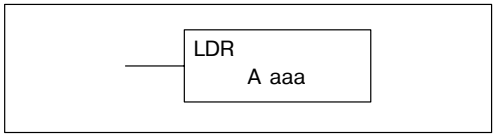
Команда AND Move копирует данные из таблицы в заданную ячейку памяти, проводя операцию логического И каждого слова с данными аккумулятора по мере их записи.



OR Move (ORMOV)

X	X	√
430	440	450

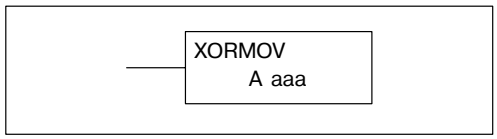
Команда OR Move копирует данные из таблицы в заданную ячейку памяти, проводя операцию логического ИЛИ каждого слова с данными аккумулятора по мере их записи.



Exclusive OR Move (XORMOV)

X	X	√
430	440	450

Команда Exclusive OR Move копирует данные из таблицы в заданную ячейку памяти, проводя операцию Исключающего ИЛИ каждого слова с данными аккумулятора по мере их записи.



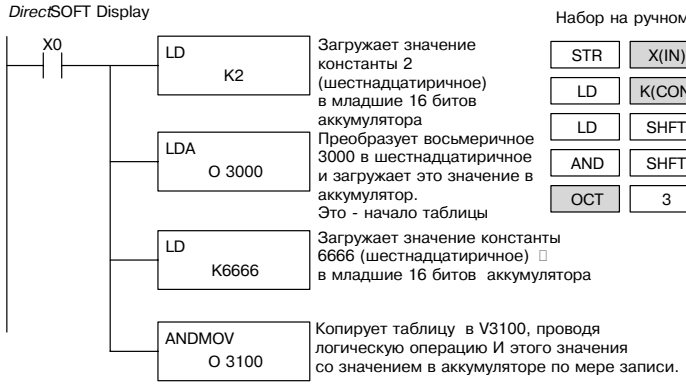
Следующее описание применимо к командам AND Move, OR Move и Exclusive OR Move. Таблицей является только некоторая область ячеек V-памяти. Эти команды копируют данные таблицы в другую заданную ячейку, выполняя логическую операцию по каждому слову с содержимым аккумулятора по мере того, как пишется новая таблица.

Шаг 1: - Загрузить в первый уровень стека аккумулятора длину таблицы (число ячеек V-памяти). Этот параметр должен быть шестнадцатиричным значением, от 0 до FF.
Шаг 2: - Загрузить в аккумулятор начальный адрес V-памяти для таблицы. Этот параметр должен быть шестнадцатиричным значением. Вы можете использовать команду LDA для преобразования восьмеричного адреса в шестнадцатиричный.
Шаг 3: - Загрузить в аккумулятор битовый набор в двоично-десятичном/шестнадцатиричном формате, который будет комбинироваться с содержимым таблицы по мере того, как она будет копироваться.
Шаг 4: - Вставить команду AND Move, OR Move или Exclusive OR Move. Они определит начальную ячейку копирования исходной таблицы. Эта новая таблица будет иметь ту же длину, что и исходная таблица.

Тип данных операнда		Диапазон DL450
	A	**aaa**
V-память	V	Все (см. стр. 3-42)

Справа показан пример таблицы, содержащей двоично-десятичные данные (для демонстрационных целей). Предположим, что мы хотим переместить таблицу из двух слов в V3000 и выполнить операцию логического И с ней и с K6666. Копия таблицы в V3100 показывает результат операции логического И для каждого слова.

Программа на следующей странице реализует операцию ANDMove В ней предполагается, что данные в таблице в V3000 - V3001 уже имеются. Сначала мы загружаем длину таблицы (два слова) в аккумулятор. Затем мы загружаем начальный адрес исходной таблицы, используя команду LDA. Далее мы загружаем данные в аккумулятор для логической операции И с таблицей. В команде ANDMove мы определяем адресат таблицы, V3100.

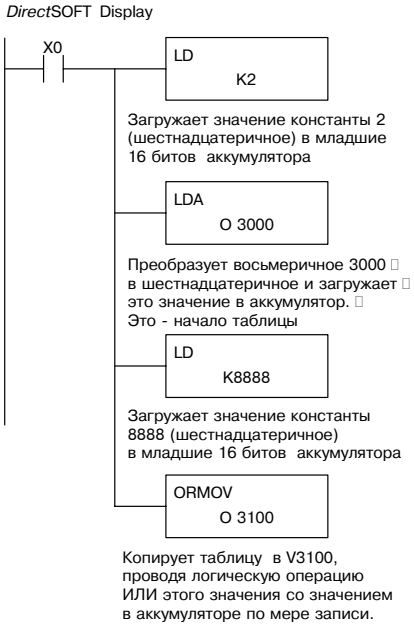


Набор на ручном программаторе

STR	X(IN)	0	↵					
LD	K(CON)	5	↵					
LD	SHFT	A	OCT	3	0	0	0	↵
AND	SHFT	M	O	V				
OCT	3	1	0	0	↵			

В примере справа показана таблица из двух слов в V3000 и логическое ИЛИ этого значения с K8888. Копия этой таблицы в V3100 показывает результат операции ИЛИ для каждого слова.

В программе справа выполняется приведенный выше пример команды ORMOVE. В нем предполагается, что данные в таблице уже заданы. Сначала мы загружаем длину таблицы (два слова) в аккумулятор. Затем мы загружаем начальный адрес исходной таблицы, используя команду LDA. Затем мы загружаем данные в аккумулятор для выполнения логической операции ИЛИ с таблицей. В команде ORMOVE мы определяем адресат таблицы, V3100.



Набор на ручном программаторе

STR	X(IN)	0	↵					
LD	K(CON)	5	↵					
LD	SHFT	A	OCT	3	0	0	0	↵
OR	SHFT	M	O	V				
OCT	3	1	0	0	↵			

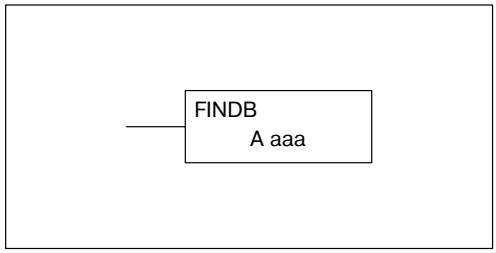
В примере справа показана таблица из двух слов в V3000 и логическое Исключающее ИЛИ (XOR) этого значения с K3333. Копия этой таблицы в V3100 показывает результат операции Исключающее ИЛИ (XOR) для каждого слова.

Пример программы для XORMOV аналогичен примеру для ORMOVE. Используется только команда XORMOV. На ручном программаторе Вы должны использовать клавишу SHFT и явно набрать " XORMOV ".

Find Block (FINDB)

X	X	√
430	440	450

Команда Find Block производит поиск наличия в таблице V - памяти заданного блока значений. Функциональные параметры загружаются в первый и второй уровни стека аккумулятора и в аккумулятор тремя дополнительными командами. Если блок найден, то его начальный адрес запоминается в аккумуляторе. Если блок не найден, то выставляется флаг SP53.



Тип данных операнда		Диапазон DL450
	A	**aaa**
V-память	V	Все (см. стр. 3-42)
V-память	P	Все (см. стр. 3-42)

Флаги дискретных разрядов	Описание
SP53	Включен, когда выполняется команда Find Block, но блок данных не найден в заданной таблице.

Чтобы запрограммировать функцию Find Block, необходимы перечисленные ниже шаги.

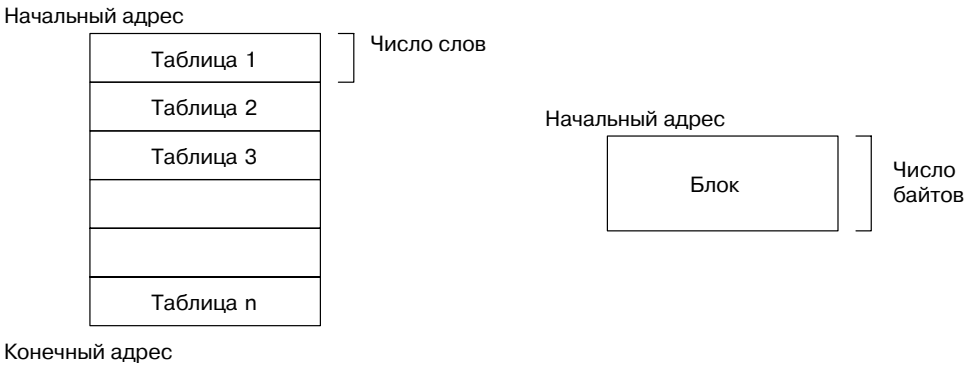
Шаг 1: - Загрузить число байтов в блоке, который должен быть найден. Этот параметр должен быть шестнадцатиричным значением, от 0 до FF.

Шаг 2: - Загрузить длину таблицы (число слов), в которой должен проводиться поиск. Команда Find Block будет проверять многие таблицы, которые находятся рядом в V-памяти. Данный параметр должен быть шестнадцатеричным значением, от 0 до FF.

Шаг 3: - Загрузить в аккумулятор конечный адрес для всех таблиц. Этот параметр должен быть шестнадцатиричным значением. Вы можете использовать команду LDA для преобразования восьмеричного адреса в шестнадцатиричный.

Шаг 4: - Загрузить в аккумулятор начальный адрес для всех таблиц. Этот параметр должен быть шестнадцатиричным значением. Вы можете использовать команду LDA для преобразования восьмеричного адреса в шестнадцатиричный.

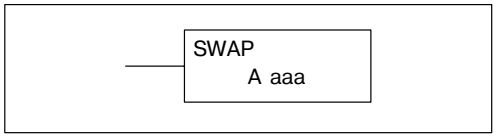
Шаг 5: - Вставить команду Find Block. Она определит начальную ячейку блока данных, который Вы пытаетесь найти.



Swap (SWAP)

X	X	√
430	440	450

Команда Swap обменивает данные в двух таблицах одинаковой длины. Устанавливает один бит в значение ”0” в диапазоне ячеек V - памяти.

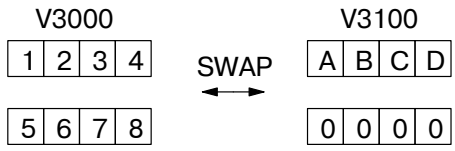


Следующее описание применимо к обеим табличным командам Set Bit и Reset Bit.

Шаг 1: - Загрузить в первый уровень стека аккумулятора длину таблиц (число ячеек V - памяти). Этот параметр должен быть шестнадцатиричным значением, от 0 до FF. Напоминаем, что таблицы должны быть одинаковой длины.
Шаг 2: - Загрузить в аккумулятор начальный адрес V-памяти для первой таблицы. Этот параметр должен быть шестнадцатеричным значением. Вы можете использовать команду LDA для преобразования восьмеричного адреса в шестнадцатиричный.
Шаг 3: - Вставить команду Swap. Она определяет начальный адрес второй таблицы. Этот параметр должен быть шестнадцатиричным значением. Вы можете использовать команду LDA для преобразования восьмеричного адреса в шестнадцатиричный.
Полезная Подсказка: - Перестановка данных происходит в одном цикле сканирования. Если команда выполняется в нескольких последовательных циклах сканирования, то трудно распознать фактическое содержание любой из таблиц в конкретный момент времени. Поэтому напоминаем, что выполнять перестановки необходимо только в одном цикле сканирования.

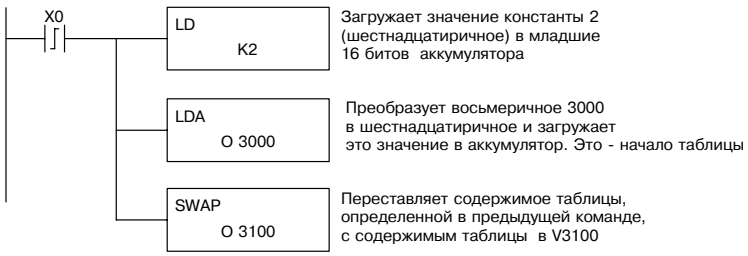
Тип данных операнда	Диапазон DL450
A	**aaa**
V-память	V Все (см. стр. 3-42)

В примере справа показана таблица из двух слов в V3000. Мы переставляем ее содержимое с другой таблицей из двух слов в V3100 , используя команду Swap. Необходимая программа приведена ниже.



В приведенной ниже программе примера используется контакт PD (срабатывает в одном цикле сканирования при переходе ВЫКЛ-ВКЛ). Сначала мы загружаем длину таблицы (два слова) в аккумулятор. Затем мы загружаем в аккумулятор адрес первой таблицы (V3000), используя команду LDA для преобразования восьмеричного адреса в шестнадцатеричный. Следует отметить, что не имеет никакого значения, какая таблица будет названа ”первой”, так как результаты перестановки будут одни и те же.

*Direct*SOFT



Набор на ручном программаторе

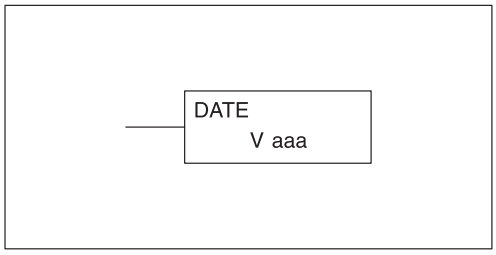
STR	X(IN)	SHFT	P	D	SHFT	0	⟵			
LD	K(CON)	2	⟵							
LD	SHFT	A	OCT	3	0	0	0	⟵		
SHFT	S	W	A	P	OCT	3	1	0	0	⟵

Команды времени / даты

Date (DATE)

X	√	√
430	440	450

Команда Date может быть использована для установки даты в процессоре. Для установки даты команде требуется две последовательные ячейки V-памяти (Vaaa). Если значения в указанных ячейках не допустимы, то дата не будет установлена. Текущую дату можно прочитать в 4 последовательных ячейках V-памяти (V7771-V7774).

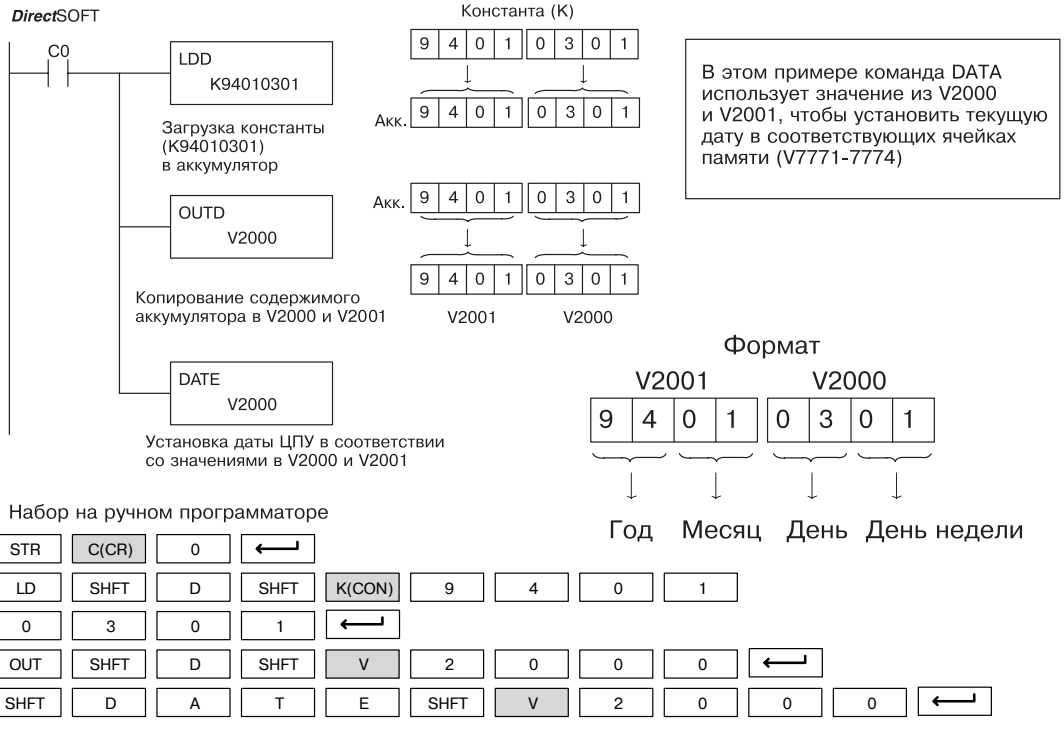


Дата	Диапазон	Ячейка V-памяти (BCD) (только для чтения)
Год	0-99	V7774
Месяц	1-12	V7773
Число	1-31	V7772
День недели	0-06	V7771

Значения, вводимые для дня недели: 0=воскресенье, 1=понедельник, 2=вторник, 3=среда, 4=четверг, 5=пятница, 6=суббота.

Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	aaa	aaa
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42

В следующем примере, когда C0 включен, константа (K94010301) загружается в аккумулятор командой Load Double (C0 должен быть контактом от однотактной команды PD). Значение в аккумуляторе выводится в V2000 командой Out Double. Команда Date использует значение в V2000 для установки даты в процессоре.



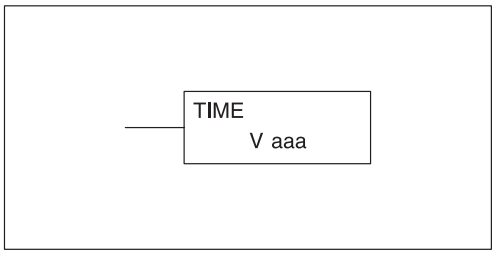
Набор на ручном программаторе

STR	C(CR)	0	↵							
LD	SHFT	D	SHFT	K(CON)	9	4	0	1		
0	3	0	1	↵						
OUT	SHFT	D	SHFT	V	2	0	0	0	↵	
SHFT	D	A	T	E	SHFT	V	2	0	0	0 ↵

Time (TIME)

X	√	√
430	440	450

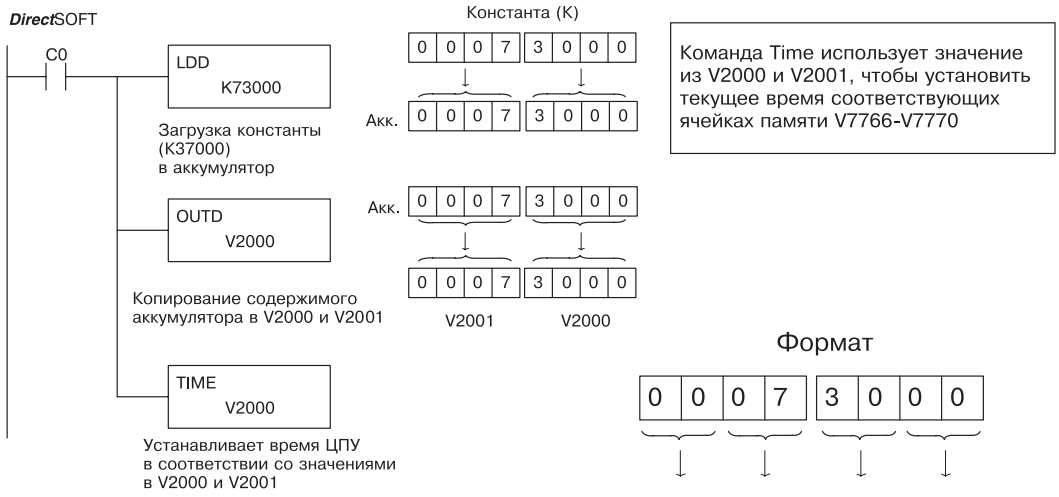
Команда Time может быть использована для установки времени (24 часа) в процессоре. Для установки времени команде требуется две последовательные ячейки V-памяти (Vaaa). Если значения в указанных ячейках не допустимы, то время не будет установлено. Текущее время можно прочитать в ячейках памяти V7747 и V7766-V7770.



Дата	Диапазон	Ячейка V-памяти (BCD) (только для чтения)
1/100 секунды (10 мс)	0-99	V7747
Секунды	0-59	V7766
Минуты	0-59	V7767
Часы	0-23	V7770

Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42

В следующем примере, когда C0 включен, константа (K73000) загружается в аккумулятор командой Load Double (C0 должен быть контактом от однотактной команды PD). Значение в аккумуляторе выводится в V2000 командой Out Double. Команда Time использует значение в V2000 для установки времени в процессоре.



Набор на ручном программаторе

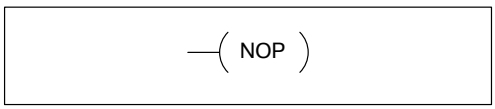
STR	C(CR)	0	↵								
LD	SHFT	D	SHFT	K(CON)	7	3	0	0	0	↵	
OUT	SHFT	D	SHFT	V	2	0	0	0	↵		
SHFT	T	I	M	E	SHFT	V	2	0	0	0	↵

Команды управления процессором

No Operation (NOP)

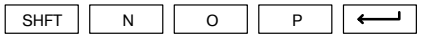
√	√	√
430	440	450

No Operation - пустая (непрограммируемая) ячейка памяти. Эти команды являются только символами-заполнителями в программе. Поэтому вам не нужно их программировать, поскольку они автоматически появляются после окончания программы.



*Direct*SOFT Display

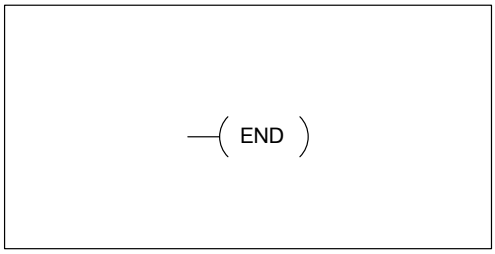
Набор на ручном программаторе



End (END)

√	√	√
430	440	450

Команда End отмечает точку окончания обычного сканирования программы. Команда End требуется в конце основного тела программы. Если команда End пропущена, то возникнет ошибка и процессор не перейдет в Рабочий режим. Метки данных, подпрограммы и программы прерывания находятся после команды End. Команда End - безусловная, поэтому для нее не требуется входной контакт.



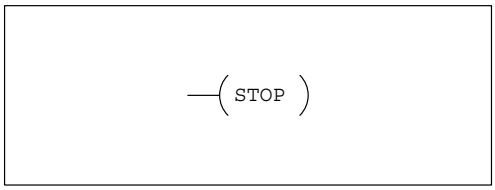
*Direct*SOFT Display

Набор на ручном программаторе

Stop (STOP)

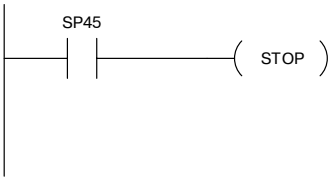
√	√	√
430	440	450

Команда Stop меняет режим работы процессора с Рабочего режима на Программный (Stop) режим. Эта команда обычно используется, чтобы остановить работу ПЛК при сбоях, таких как сбой модуля ввода/вывода.

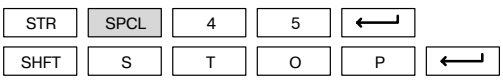


В следующем примере, когда SP45 включится, показывая сбой модуля ввода/вывода, процессор остановит работу и переключится в Программный режим.

*Direct*SOFT Display



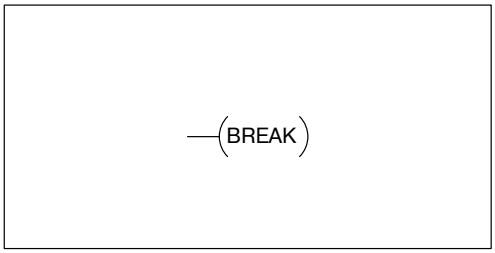
Набор на ручном программаторе



Break (BREAK)

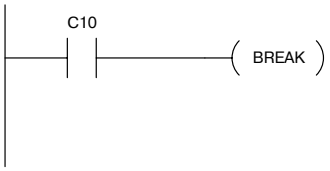
X	√	√
430	440	450

Команда Break изменяет рабочий режим процессора с Рабочего в режим Test Program (Тестирование в Программном режиме). Эта команда обычно используется для помощи при отладке прикладной программы. Команда Break позволяет сохранить данные V-памяти и регистров отображения, которые обычно стираются при команде Stop или при нормальном переходе из Рабочего режим в Программный.

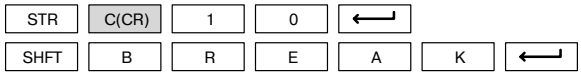


В следующем примере, когда C10 включен, процессор останавливает работу и переключается в режим Test Program.

*Direct*SOFT Display



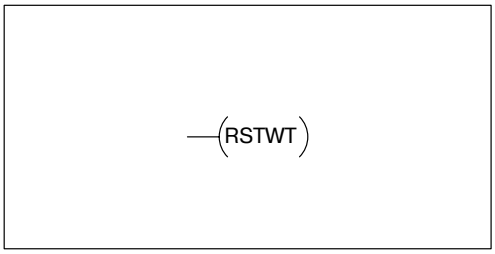
Набор на ручном программаторе



Reset Watch Dog Timer (RSTWT)

√	√	√
430	440	450

Команда Reset Watch Dog Timer сбрасывает таймер сканирования процессора. Настройка по умолчанию для сторожевого таймера - 200 мс. Время сканирования очень редко превышает 200 мс, но это возможно. Циклы For/Next, подпрограммы, программы прерывания и табличные команды могут быть запрограммированы так, что время сканирования станет больше 200 мс. Когда команды используются таким способом, что превышают установку сторожевого таймера, то эту команду можно использовать для сброса таймера.



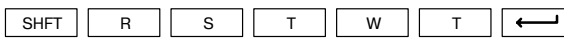
Если время сканирования превысит установку сторожевого таймера, то произойдет программная ошибка времени ожидания (E003) и процессор перейдет в Программный режим. Местоположение команды RSTWT в программе очень важно. Команда должна быть выполнена до того момента, когда время сканирования превысит установку сторожевого таймера.

Если время сканирования постоянно превышает установку сторожевого таймера, то значение времени ожидания может быть увеличено сверх установленных по умолчанию 200 мс с помощью AUX55 на ручном программаторе или соответствующей вспомогательной функции в программном пакете. Это устраняет потребность в команде RSTWT.

В следующем примере, таймер сканирования процессора будет сброшен в 0 при выполнении команды RSTWT. Подробный пример смотрите в команде For/Next.

*Direct*SOFT Display

Набор на ручном программаторе

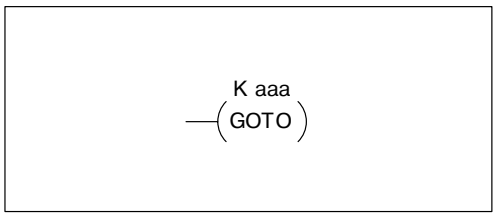


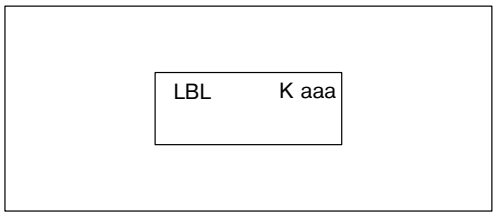
Команды управления программой

Goto /Label (GOTO/LBL)

X	√	√
430	440	450

Goto / Label пропускает все команды между Goto и соответствующей командой LBL. Значение операнда для Goto и соответствующей командой LBL то же самое. Логические команды между Goto и соответствующей командой LBL не выполняются, когда действует команда Goto. В программе может быть использовано до 128 команд Goto и до 64 команд LBL.

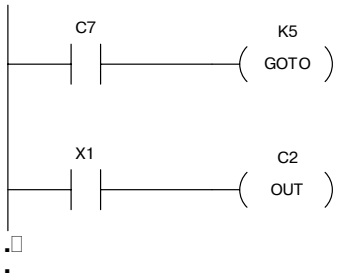


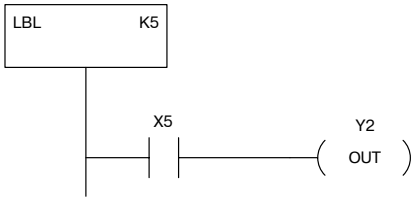


Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
Константа	K	1-FFFF	1-FFFF

В следующем примере, когда C7 включен, все логические команды между GOTO и соответствующей командой LBL (обозначенной с тем же значением константы Kaaa) будут пропущены. Пропущенные команды не будут выполнены процессором.

*Direct*SOFT Display





Набор на ручном программаторе

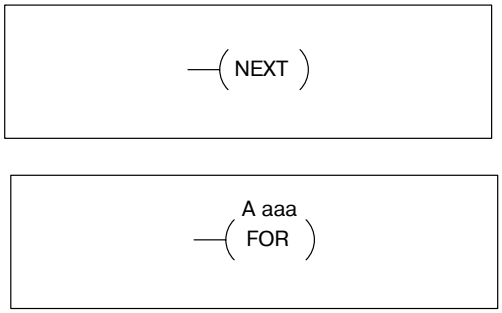
STR	C(CR)	7	↵					
SHFT	G	O	T	O	SHFT	K(CON)	5	↵
STR	X(IN)	1	↵					
OUT	C(CR)	2	↵					

SHFT	L	B	L	SHFT	K(CON)	5	↵
STR	X(IN)	5	↵				
OUT	Y(OUT)	2	↵				

For / Next (FOR/NEXT)

X	√	√
430	440	450

Команды For и Next используются для выполнения части релейной логики между командами For и Next указанное число раз. Когда включена команда For, программа будет циклически повторяться указанное число раз. Если команда For не включена, то часть релейной логики между командами For и Next не выполняется.

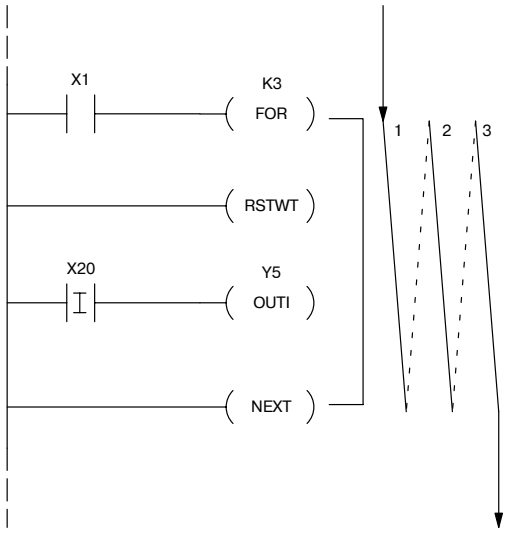


Команды For и Next не могут быть вложены. В программе можно использовать до 64 циклов For / Next. При превышении максимального числа циклов For / Next возникнет ошибка E413. Во время выполнения цикла For / Next обычное обновление входов/выходов и служебные действия процессора откладываются. Программа сканирования может значительно увеличиться в зависимости от числа повторений логических команд между командами For и Next. За исключением непосредственных команд ввода/вывода, входы/выходы не будут обновляться, пока не завершится выполнение программы для этого цикла сканирования. В зависимости от отрезка времени, требуемого для завершения выполнения программы, может возникнуть необходимость сбросить сторожевой таймер внутри цикла For / Next, используя команду RSTWT.

Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	1-9999	1-9999

В следующем примере, когда X1 включен, прикладная программа внутри цикла For / Next будет выполнена три раза. Если X1 выключен, программа внутри цикла не будет выполнена. Непосредственные команды могут быть нужны или не нужны в зависимости от вашей прикладной программы. Также команда RSTWT не требуется, если цикл For / Next не делает время сканирования больше, чем установка сторожевого таймера. Для более подробной информации по сторожевому таймеру обратитесь к команде RSTWT.

*Direct*SOFT Display



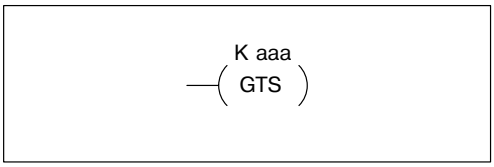
Набор на ручном программаторе

STR	X(IN)	1	↵	
SHFT	F	O	R	SHFT
SHFT	R	S	T	W
STR	SHFT	I	SHFT	X(IN)
Y(OUT)	SHFT	I	SHFT	Y(OUT)
SHFT	N	E	X	T

Goto Subroutine (GTS)

X	√	√
430	440	450

Команда Goto Subroutine позволяет помещать часть программы релейной логики вне основного тела программы для того, чтобы выполнять ее только при необходимости. В программе может использоваться максимум 128 команд GTS.



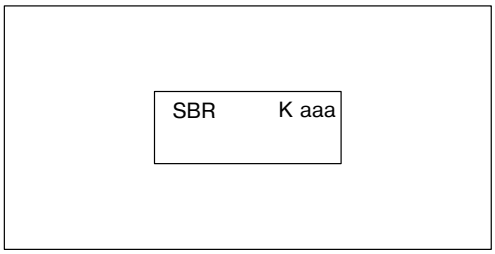
После завершения подпрограммы немедленно после команды GTS происходит возврат к выполнению главной программы. Команды могут быть вложены вплоть до 8 уровней. Если этот предел превышен, то возникает ошибка E412. Обычно эти команды используются в прикладных программах, в которых блок программной логики может выполняться медленно и не требовать выполнения при каждом сканировании.

Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	aaa	aaa
Константа	K	1-FFFF	1-FFFF

Subroutine (SBR)

X	√	√
430	440	450

Метка подпрограммы и вся связанная логика помещается после оператора End в программе. В программе может использоваться максимум 128 команд GTS и 64 команды SBR. Когда подпрограмма вызывается из главной программы, процессор будет выполнять подпрограмму (SBR) с той же самой константой (K), что и команда GTS, которая вызвала подпрограмму..



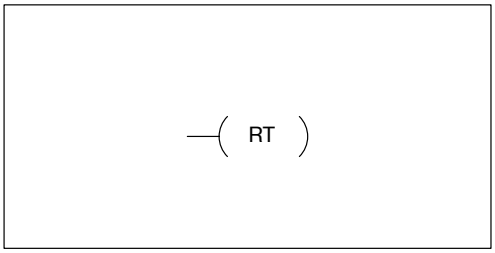
Помещенный в подпрограмму код позволяет сканировать и выполнять подпрограмму только при необходимости, так как она находится после команды End. Код, который не сканируется, не влияет на общее время цикла сканирования программы.

Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	aaa	aaa
Константа	K	1-FFFF	1-FFFF

Subroutine Return (RT)

X	√	√
430	440	450

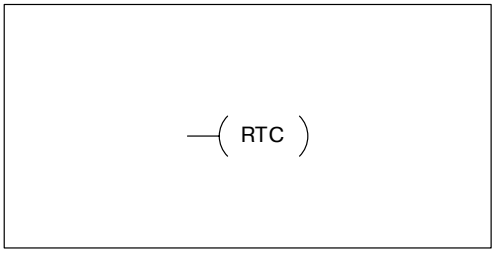
Когда в подпрограмме выполняется команда Subroutine Return, процессор вернется к той точке в главном теле программы, с которой эта подпрограмма была вызвана. Subroutine Return используется для завершения подпрограммы, и она должна быть последней командой в подпрограмме и являться автономной (в цепи не должно быть входного контакта).



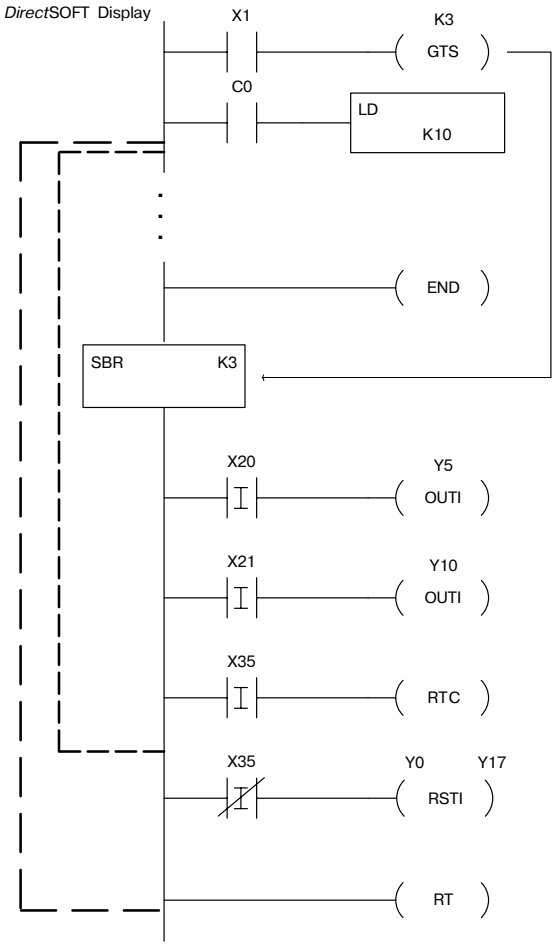
Subroutine Return Conditional (RTC)

X	√	√
430	440	450

Команда Subroutine Return Conditional - дополнительная команда, используемая с входным контактом для выполнения условного возврата из подпрограммы. Команда Subroutine Return (RT) требуется также для завершения подпрограммы.



В следующем примере, когда X1 включен, будет вызвана подпрограмма K3. Процессор перейдет к метке подпрограммы K3, и будет выполняться логика в подпрограмме. Если X35 включен, то процессор вернется в главную программу по команде RTC. Если X35 не включен, то Y0-Y17 будут сброшены, и затем процессор вернется в главное тело программы.



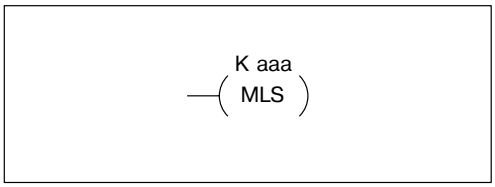
Набор на ручном программаторе

STR	X(IN)	1	↵				
SHFT	G	T	S	SHFT	K(CON)	3	↵
.							
.							
END	↵						
SHFT	S	B	R	SHFT	K(CON)	3	↵
STR	SHFT	I	SHFT	X(IN)	2	0	↵
OUT	SHFT	I	SHFT	Y(OUT)	5	↵	
STR	SHFT	I	SHFT	X(IN)	2	1	↵
OUT	SHFT	I	SHFT	Y(OUT)	1	0	↵
STR	SHFT	I	SHFT	X(IN)	3	5	↵
SHFT	R	T	C	↵			
STR	NOT	SHFT	I	SHFT	X(IN)	3	5	↵
RST	SHFT	I	SHFT	Y(OUT)	0	Y(OUT)	1	7	↵
SHFT	R	T	↵					

Master Line Set (MLS)

√	√	√
430	440	450

Команда Master Line Set позволяет программе управлять разделами релейной логики, формируя новую шину питания, управляемую главной левой шиной питания. Главная левая шина всегда ведущей линией 0. Когда используется команда MLS K1, на уровне 1 создается новая шина. Команды Master Line Set и Master Line Reset могут быть использованы для того, чтобы вложить шины глубиной до семи уровней.

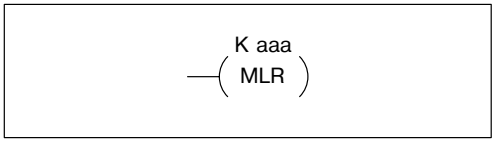


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	aaa	aaa	aaa
Константа	K	1-7	1-7	1-7

Master Line Reset (MLR)

√	√	√
430	440	450

Команда Master Line Reset отмечает конец управления для соответствующей команды MLS. Команда MLR имеет ссылку на единицу меньше, чем соответствующая команда MLS.



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	aaa	aaa	aaa
Константа	K	0-6	0-6	1-6

Понимание ведущих управляющих реле

Команды Master Line Set (MLS) и Master Line Reset (MLR) позволяют Вам быстро управлять потоками "питания" для разделов программы RLL. Это обеспечивает гибкость управления программой. Следующий пример показывает, как работают команды MLS и MLR, создавая суб-шины "питания" для управляющей логики.

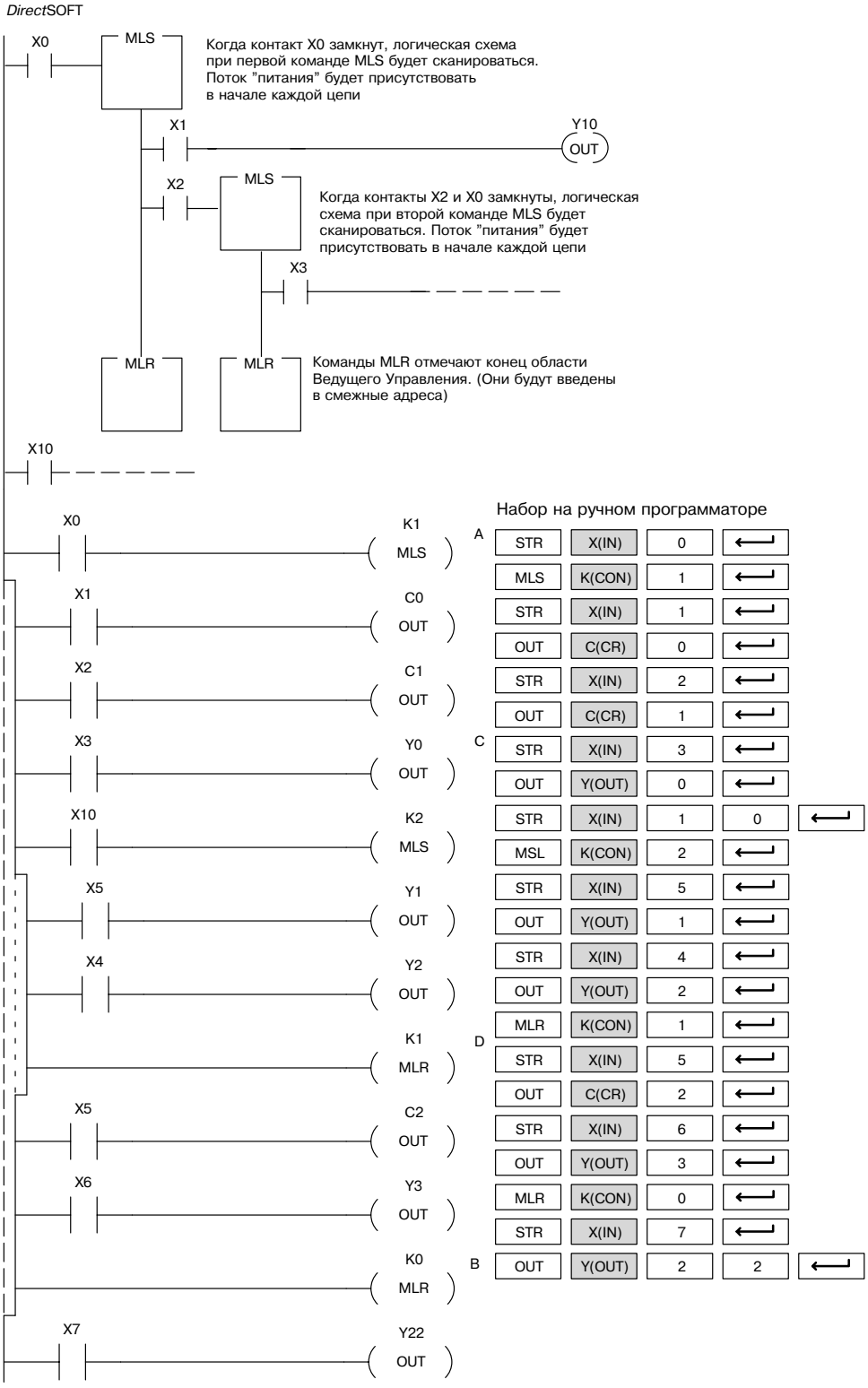
Пример MLS/MLR

В следующем примере MLS/MLR логическая схема между первым MLS K1 (A) и MLR KO (B) будет иметь "питание" в шине "питания", если вход X0 включен. (Следует отметить, что если X0 буде отключен, логическая схема будет сканироваться, но поток "питания" будет отсутствовать.) Логическая схема между MLS K2 (C) и MLR K1 (D) будет иметь "питание" в шине "питания", только если X10 и X0 включены. Последняя цепь не управляется ни одной из обмоток MLS, она всегда имеет "питание" в начале цепи.

Напоминаем, что команды MLS / MLR управляют потоками "питания" между шинами "питания". Они не управляют выполнением команд. Команды еще выполняются, но поскольку отсутствует поток "питания", логическая схема не будет включать обмотки. Рассмотрим следующий случай для нашего примера.

1. X0 выключен, и это означает, что нет никакого потока "питания" во второй цепи.
2. Вы используете устройство программирования, чтобы включить C0.

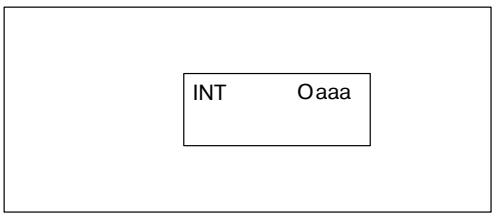
Поскольку поток "питания" отсутствует во второй цепи (STR X1, OUT C0), то Вы ожидаете, что C0 останется включенным, так как Вы включили его с устройства для программирования. Однако команда MLS не предполагает, что команды внутри зоны управления не выполняются. Фактически они выполняются, но без потока "питания". Поэтому в нашем случае C0 будет включено при выполнении программной цепи. Это происходит потому, что процессор "видит", что в этой цепи нет потока "питания". Когда процессор выполнит эту цепочку, он выключит C0.


DirectSOFT
X0
MLS
Когда контакт X0 замкнут, логическая схема при первой команде MLS будет сканироваться. Поток "питания" будет присутствовать в начале каждой цепи
X1
Y10
OUT
X2
MLS
Когда контакты X2 и X0 замкнуты, логическая схема при второй команде MLS будет сканироваться. Поток "питания" будет присутствовать в начале каждой цепи
X3
MLR
MLR
Команды MLR отмечают конец области Ведущего Управления. (Они будут введены в смежные адреса)
X10
Набор на ручном программаторе
X0
K1
MLS
X1
C0
OUT
X2
C1
OUT
X3
Y0
OUT
X10
K2
MLS
X5
Y1
OUT
X4
Y2
OUT
K1
MLR
X5
C2
OUT
X6
Y3
OUT
K0
MLR
X7
Y22
OUT
A
STR X(IN) 0
MLS K(CON) 1
STR X(IN) 1
OUT C(CR) 0
STR X(IN) 2
OUT C(CR) 1
C
STR X(IN) 3
OUT Y(OUT) 0
STR X(IN) 1 0
MSL K(CON) 2
STR X(IN) 5
OUT Y(OUT) 1
STR X(IN) 4
OUT Y(OUT) 2
D
MLR K(CON) 1
STR X(IN) 5
OUT C(CR) 2
STR X(IN) 6
OUT Y(OUT) 3
MLR K(CON) 0
STR X(IN) 7
B
OUT Y(OUT) 2 2

Команды прерывания

Interrupt (INT)

√	√	√
430	440	450



Команда Interrupt позволяет поместить раздел релейной логики вне главного тела программы и выполнять его при необходимости. Прерывания могут быть вызваны из программы или посредством модуля прерываний, который устанавливается в слоте 0 и обеспечивает 8 входов прерывания.

Один модуль прерывания может быть установлен в системе DL430 (X0 - X7) и два модуля прерывания - в системе DL440 или DL450 (X0 - X7 и X20-X27). Напоминаем, модули прерывания занимают 16 точек.

Программное прерывание использует прерывание #17, что означает, что аппаратное прерывание #17 и программное прерывание не могут использоваться вместе.

Обычно прерывания используются в прикладной программе, где требуется быстрый отклик на вход, или раздел программы должен выполняться быстрее, чем обычный цикл сканирования процессора. Метка прерывания и вся связанная логика должны помещаться в программе после оператора End. Когда программа прерывания вызвана из модуля прерывания или из программы прерывания, то процессор заканчивает выполнение текущей команды, а затем будет выполнять указанную программу прерывания. Имеется только одно программное прерывание, оно имеет метку INT17. Выполнение программы продолжится с того места, где оно было прервано сразу после обработки прерывания.

Программное прерывание устанавливается при программировании времени прерывания в V737. Допустимый диапазон - от 3 до 1000 мс. Значение должно быть в двоично-десятичном формате. Прерывание не выполняется, если значение выходит из этого диапазона.

См. пример программы с программным прерыванием.

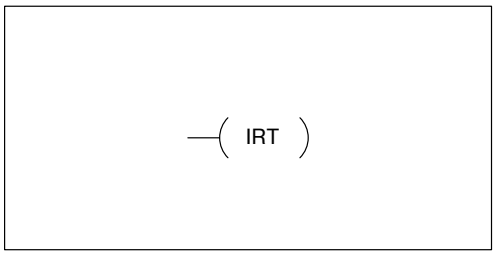
Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
Константа□	O□	0-7□	0-17□	0-17

	Программное обеспечение		DL430		DL440/450	
	Вход прерывания	Программа прерывания	Вход прерывания	Программа прерывания	Вход прерывания	Программа прерывания
	–	–	X0	INT 0	X0	INT 0
	–	–	X1	INT 1	X1	INT 1
	–	–	X2	INT 2	X2	INT 2
	–	–	X3	INT 3	X3	INT 3
	–	–	X4	INT 4	X4	INT 4
	–	–	X5	INT 5	X5	INT 5
	–	–	X6	INT 6	X6	INT 6
	–	–	X7	INT 7	X7	INT 7
2 модуль	–	–	–	–	X20	INT 10
	–	–	–	–	X21	INT 11
	–	–	–	–	X22	INT 12
	–	–	–	–	X23	INT 13
	–	–	–	–	X24	INT 14
	–	–	–	–	X25	INT 15
	–	–	–	–	X26	INT 16
	В V737 установлено время прерывания	INT 17	–	–	X27 (не может использоваться вместе с программным прерыванием)	INT 17

Interrupt Return (IRT)

√	√	√
430	440	450

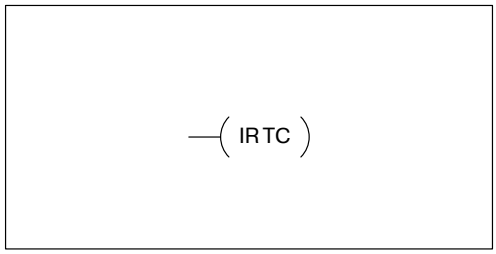
При выполнении команды Interrupt Return в программе прерывания процессор вернется в ту точку главного тела программы, из которой программа прерывания была вызвана. Interrupt Return программируется как последняя команда в программе прерывания и является отдельно стоящей командой (в цепи нет входного контакта).



Interrupt Return Conditional (IRTC)

√	√	√
430	440	450

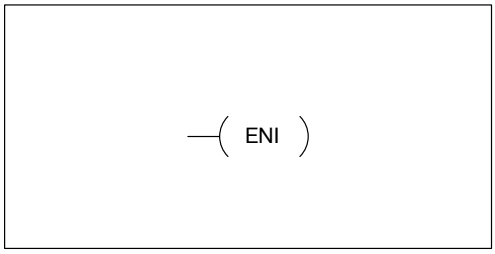
Команда Interrupt Return Conditional - дополнительная команда, используемая с входным контактом, для выполнения условного возврата из программы прерывания. Для завершения программы прерывания требуется команда Interrupt Return.



Enable Interrupts (ENI)

√	√	√
430	440	450

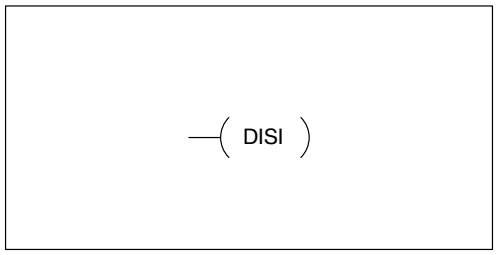
Команда Enable Interrupt применяется в главном теле прикладной программы (перед командой End), чтобы разрешить аппаратные или программные прерывания. Как только обмотка будет возбуждена, прерывания будут разрешены до тех пор, пока они не будут запрещены командой Disable Interrupt.



Disable Interrupts (DISI)

√	√	√
430	440	450

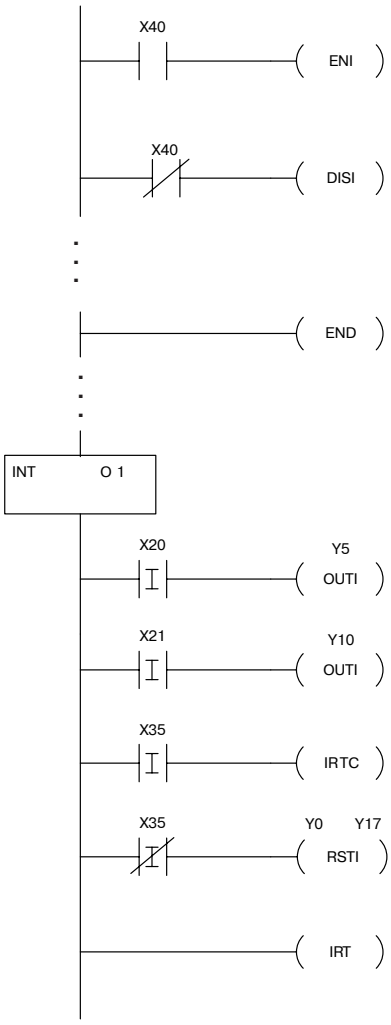
Команда Disable Interrupt применяется в главном теле прикладной программы (перед командой End), чтобы запретить аппаратные и программные прерывания. Как только обмотка будет возбуждена, прерывания будут запрещены до тех пор, пока они не будут разрешены командой Enable Interrupt.



Пример прерывания для модуля прерывания

В следующем примере, когда X40 включен, прерывания разрешены. Когда X40 выключен, прерывания запрещены. После получении сигнала прерывания X1, процессор перейдет к метке прерывания INT O1. В программе прерывания будет выполняться прикладная релейная логика. Если X35 включен, процессор вернется к главной программе по команде IRTC. Если X35 не включен, то Y0 - Y17 будут сброшены, а процессор вернется к главному телу программы.

*Direct*SOFTDisplay

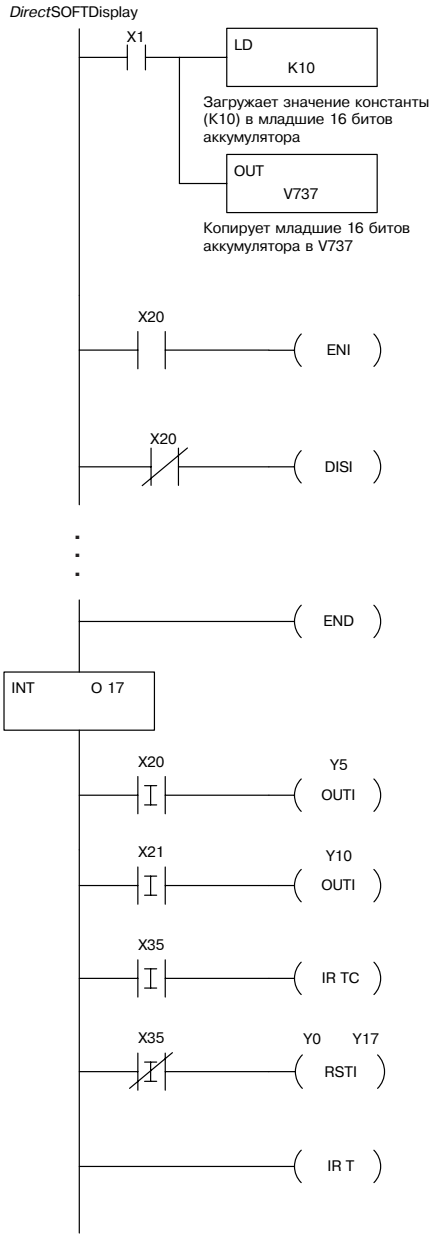


Набор на ручном программаторе

STR	X(IN)	4	0	↵				
SHFT	E	N	I	↵				
STR	NOT	X(IN)	1	7	↵			
SHFT	D	I	S	I	↵			
.								
.								
.								
END	↵							
SHFT	I	N	T	OCT	1	↵		
STR	SHFT	I	SHFT	X(IN)	2	0	↵	
OUT	SHFT	I	SHFT	Y(OUT)	5	↵		
STR	SHFT	I	SHFT	X(IN)	2	1	↵	
OUT	SHFT	I	SHFT	Y(OUT)	1	0	↵	
STR	SHFT	I	SHFT	X(IN)	3	5	↵	
SHFT	I	R	T	C	↵			
STR	NOT	SHFT	I	SHFT	X(IN)	3	5	↵
RST	SHFT	I	SHFT	Y(OUT)	0	Y(OUT)	1	7 ↵
SHFT	I	R	T	↵				

Пример прерывания для программного прерывания

В следующем примере, когда X1 включен, значение 10 копируется в V7634. Это значение устанавливает программное прерывание 10 мс. Когда X20 включен, прерывания разрешены. Когда X20 выключен, прерывания запрещены. Каждые 10 мс процессор будет переходить к метке прерывания INT O17. В программе прерывания будет выполняться прикладная релейная логика. Если X35 включен, процессор вернется к главной программе по команде IRTC. Если X35 не включен, то Y0-Y17 будут сброшены, и затем процессор вернется к главному телу программы, когда команда IRT выполняется Программное прерывание ограничено диапазоном 3 - 999 мс. Ввод 0, 1 или 2 даст время прерывания 3 миллисекунды.



Набор на ручном программаторе

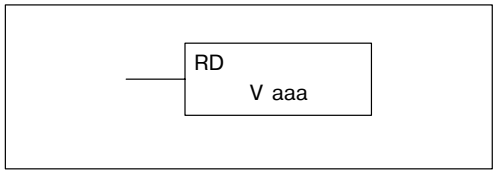
STR	X(IN)	1	↵				
LD	K(CON)	1	0				
OUT	V	7	3	7	↵		
STR	X(IN)	2	0	↵			
SHFT	E	N	I	↵			
STR	NOT	X(IN)	2	0	↵		
SHFT	D	I	S	I	↵		
END	↵						
SHFT	I	N	T	OCT	1	7	↵
STR	SHFT	I	SHFT	X(IN)	2	0	↵
OUT	SHFT	I	SHFT	Y(OUT)	5	↵	
STR	SHFT	I	SHFT	X(IN)	2	1	↵
OUT	SHFT	I	SHFT	Y(OUT)	1	0	↵
STR	SHFT	I	SHFT	X(IN)	3	5	↵
SHFT	I	R	T	C	↵		
STR	NOT	SHFT	I	SHFT	X(IN)	3	5	↵
RST	SHFT	I	SHFT	Y(OUT)	0	Y(OUT)	1	7	↵
SHFT	I	R	T	↵					

Команды интеллектуального ввода / вывода

Read from Intelligent Module (RD)

√	√	√
430	440	450

Команда Read from Intelligent Module считывает блок данных (1-128 байт максимум) с интеллектуального модуля ввода/вывода в V-память процессора. Функциональные параметры загружаются в первый и второй уровень стека аккумулятора и в аккумулятор тремя дополнительными командами. Ниже перечислены шаги, необходимые для программирования функции Read from Intelligent Module.



Шаг 1: Загрузить номер каркаса (0-3) в первый байт и номер слота (0-7) во второй байт второго уровня стека аккумулятора.
Шаг 2: Загрузить число передаваемых байтов в первый уровень стека аккумулятора (до 128 байт максимум).
Шаг 3: Загрузить адрес, с которого данные будут считываться в аккумулятор. Этот параметр должен быть шестнадцатиричным значением.
Шаг 4: Вставить команду RD, которая указывает стартовую ячейку V-памяти (Vaaa), куда будут считываться данные.
Полезная подсказка: Используйте команду LDA для преобразования восьмеричного адреса в его шестнадцатиричный эквивалент и для загрузки его в аккумулятор.

Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)

Флаги дискретных разрядов	Описание
SP54	Включен, когда команды RX, WX, RD, WT выполняются с неправильными параметрами.

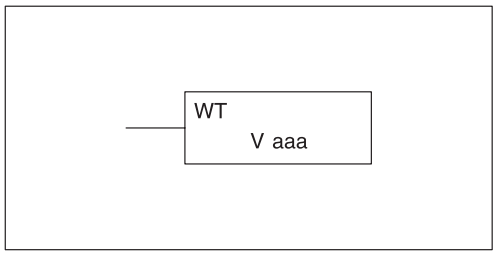
ПРИМЕЧАНИЕ: Флаги состояния действуют только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, команда RD считывает шесть байт данных с интеллектуального модуля в каркасе 1, слоте 2, начиная с адреса 0 в интеллектуальном модуле, и копирует информацию в ячейки V-памяти V1400-V1402.

Write to Intelligent Module (WT)

√	√	√
430	440	450

Команда Write to Intelligent Module записывает блок данных (1-128 байт максимум) в интеллектуальный модуль ввода/вывода из блока V-памяти в процессоре. Функциональные параметры загружаются в первый и второй уровень стека аккумулятора и в аккумулятор тремя дополнительными командами. Ниже перечислены шаги, необходимые для программирования функции Write to Intelligent Module.



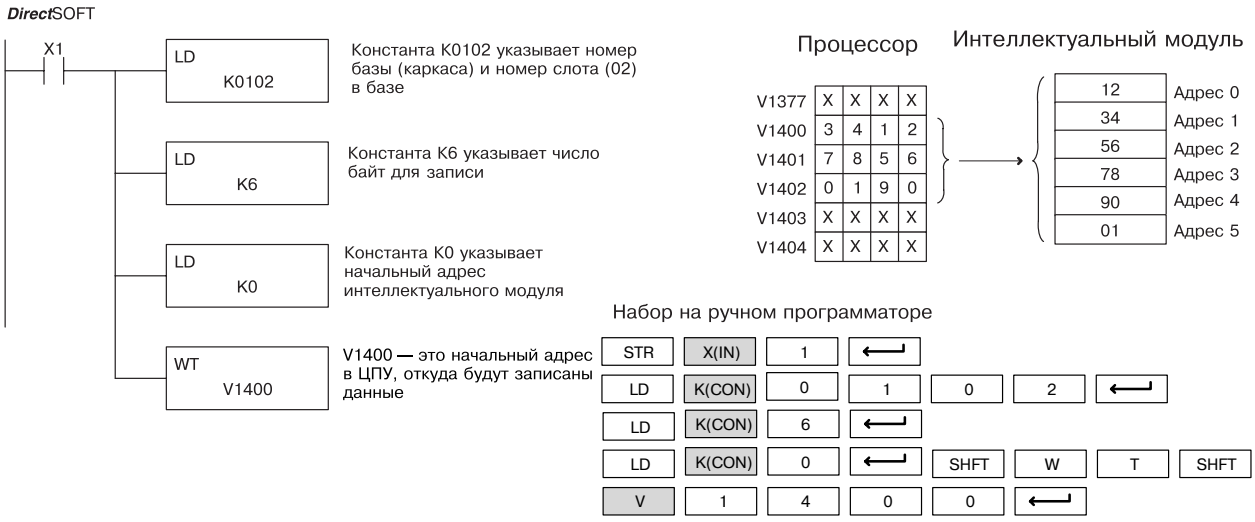
Шаг 1: Загрузить номер каркаса (0-3) в первый байт и номер слота (0-7) во второй байт второго уровня стека аккумулятора.
Шаг 2: Загрузить число передаваемых байтов в первый уровень стека аккумулятора (до 128 байт максимум).
Шаг 3: Загрузить адрес интеллектуального модуля, который будет получать данные в аккумулятор. Этот параметр должен быть шестнадцатиричным значением.
Шаг 4: Вставьте команду WT, которая указывает стартовую ячейку V-памяти (Vaaa), откуда данные будут записаны в процессор.
Полезная подсказка: Используйте команду LDA для преобразования восьмеричного адреса в его шестнадцатеричный эквивалент и для загрузки его в аккумулятор.

Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)

Флаги дискретных разрядов	Описание
SP54	Включен, когда команды RX, WX, RD, WT выполняются с неправильными параметрами.

ПРИМЕЧАНИЕ: Флаги состояния действуют только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

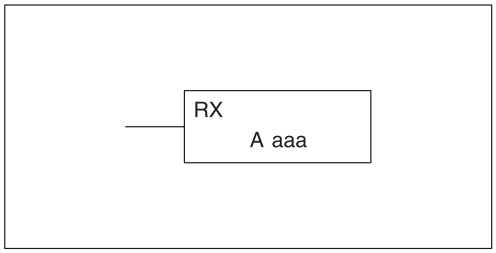
В следующем примере, когда X1 включен, команда WT записывает шесть байт данных в интеллектуальный модуль в каркасе 1, слоте 2, начиная с адреса 0 в интеллектуальном модуле, и копирует информацию из ячеек V-памяти V1400-V1402.



Сетевые команды

Read from Network (RX)

√	√	√
430	440	450



Команда Read from Network используется ведущим (master) устройством (процессором с DCM) в сети для чтения блока данных с другого процессора. Функциональные параметры загружаются в первый и второй уровень стека аккумулятора и в аккумулятор тремя дополнительными командами. Ниже перечислены шаги, необходимые для программирования функции Read from Network.

Шаг 1: Загрузить адрес ведомого (slave) устройства (0-90 двоично-десятичных) в первый байт и номер слота ведущего DCM (0-7) во второй байт второго уровня стека аккумулятора.

Шаг 2: Загрузить число передаваемых байтов (2 -128 двоично-десятичные, кратные 2) в первый уровень стека аккумулятора.

Шаг 3: Загрузить адрес, в котором Вы хотите хранить данные в ведущей станции. Этот параметр должен быть шестнадцатиричным значением.

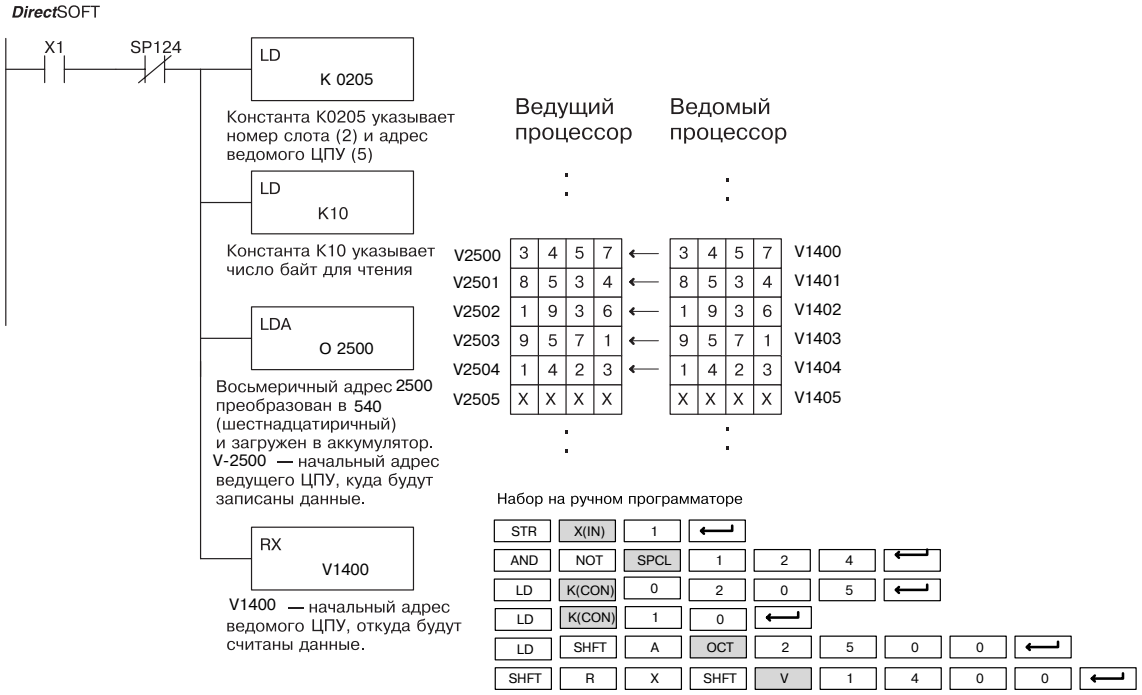
Шаг 4: Вставить команду RX, которая указывает стартовую ячейку V-памяти (Vaaa), в ведомом процессоре, куда данные будут получаться.

Полезная подсказка: Используйте команду LDA для преобразования восьмеричного адреса в его шестнадцатеричный эквивалент и для загрузки его в аккумулятор.

Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Входы	X	0–477	0–477	0–1777
Выходы	Y	0–477	0–477	0–1777
Реле управления	C	0–737	0–1777	0–3777
Стадии	S	0–577	0–1777	0–1777
Таймер	T	0–177	0–377	0–377
Счетчик	CT	0–177	0–177	0–377
Специальные реле	SP	0–137, 320–617	0–137 320–717	0–137 320–717
Глобальный ввод/вывод	GX	0–777	0–1777	0–2777
Программная память	L$	0–3583	0-7679 (7.5K программной памяти) 0-15871 (15.5K программной памяти	0-7679 (7.5K программной памяти) 0-15871 (15.5K программной памяти
Оперативная память	Z	0–FFFF	0–FFFF	0–FFFF

ПРИМЕЧАНИЕ: Если Вы используете сопроцессор CoProcessorTM, Share Data Network или модули DCM, обратитесь к соответствующему Руководству для получения информации о том, как передавать данные в сети DirectNET.

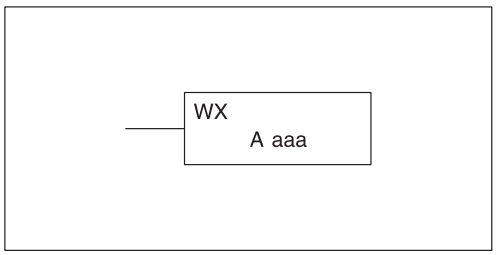
В следующем примере, когда X1 включен, а реле занятости модуля SP124 (см. "Специальные реле") не включено, команда RX обратится к DCM, работающему как ведущее устройство в слоте 2. Десять последовательных байтов данных (V1400-V1404) будут считываться из процессора с адресом станции 5 и копироваться в ячейки V-памяти V2500-V2504 в процессоре с ведущим DCM.



Write to Network (WX)

√	√	√
430	440	450

Команда Write to Network используется для записи блока данных с ведущего (master) устройства на ведомое (slave) устройство в той же самой сети. Функциональные параметры загружаются в первый и второй уровень стека аккумулятора и в аккумулятор тремя дополнительными командами. Ниже перечислены шаги, необходимые для программирования функции Write to Network.



Шаг 1: Загрузить адрес ведомого устройства (0-90 шестнадцатиричные) в первый байт и номер слота ведущего DCM (0-7) во второй байт второго уровня стека аккумулятора. (Замечание: адрес 0 действует только при равноправных станциях).

Шаг 2: Загрузить число передаваемых байтов (2 -128 двоично-десятичные, кратные 2) в первый уровень стека аккумулятора.

Шаг 3: Загрузить адрес, с которого Вы хотите получить данные в ведущую станцию. Этот адрес должен быть шестнадцатиричным значением.

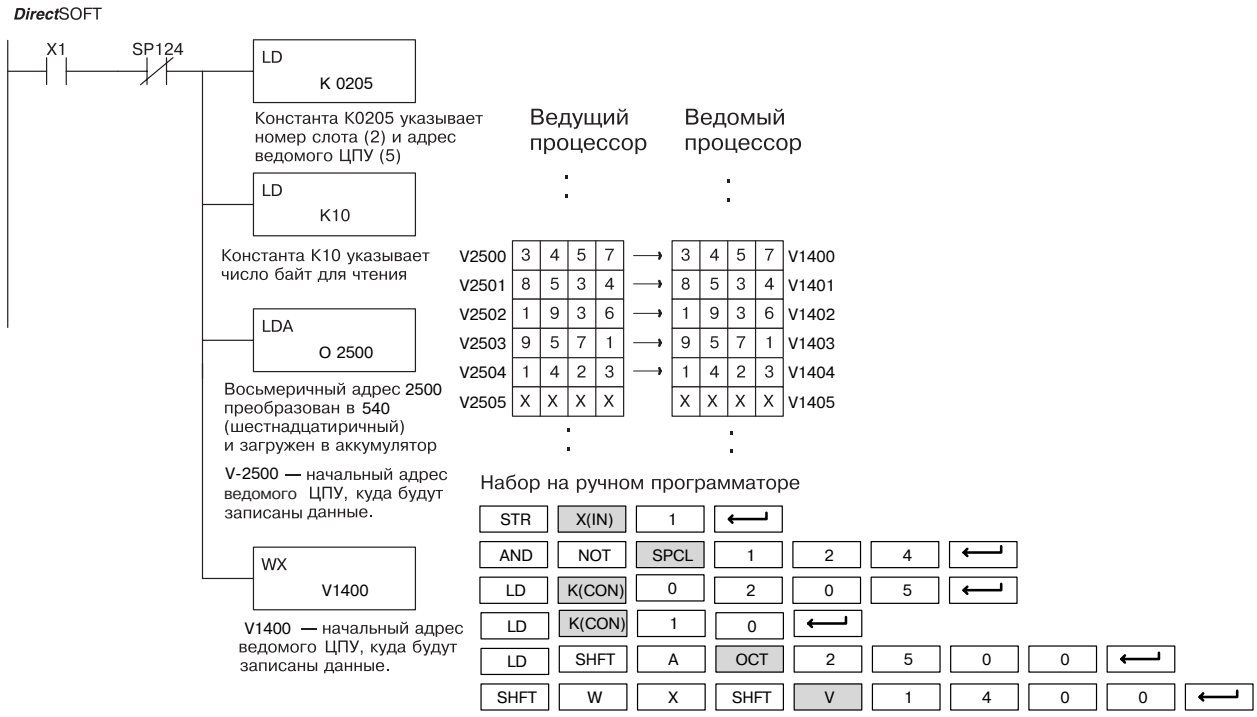
Шаг 4: Вставить команду WX и определить стартовую ячейку V-памяти (Vaaa) в ведомом процессоре, где данные будут храниться.

Полезная подсказка: Используйте команду LDA для преобразования восьмеричного адреса в его шестнадцатиричный эквивалент и для загрузки его в аккумулятор.

Тип данных операнда	A	Диапазон DL430	Диапазон DL440	Диапазон DL450
		ааа	ааа	ааа
V-память	V	Все (см. стр. 3-40)	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Входы	X	0–477	0–477	0–1777
Выходы	Y	0–477	0–477	0–1777
Реле управления	C	0–737	0–1777	0–3777
Стадии	S	0–577	0–1777	0–1777
Таймер	T	0–177	0–377	0–377
Счетчик	CT	0–177	0–177	0–377
Специальные реле	SP	0–137, 320–617	0–137 320–717	0–137 320–717
Глобальный ввод/вывод	GX	0–777	0–1777	0–2777
Программная память	L$	0–3583	0-7679 (7.5K программной памяти) 0-15873 (15.5K программной памяти)	0-7679 (7.5K программной памяти) 0-15873(15.5K программной памяти)
Оперативная память	Z	0–FFFF	0–FFFF	0–FFFF

ПРИМЕЧАНИЕ: Если Вы используете сопроцессор CoProcessorTM, Share Data Network или модули DCM, обратитесь к соответствующему Руководству для получения информации о том, как передавать данные в сети DirectNET.

В следующем примере, когда X1 включен, а реле занятости модуля SP124 (См. ”Специальные реле”) не включено, команда RX обратится к DCM, работающему как ведущее устройство в слоте 2. 10 последовательных байтов данных считываются из процессора с адреса станции 5 и копируются в ячейки V-памяти V400-V1404 в ведомом процессоре.



Команды работы с сообщениями

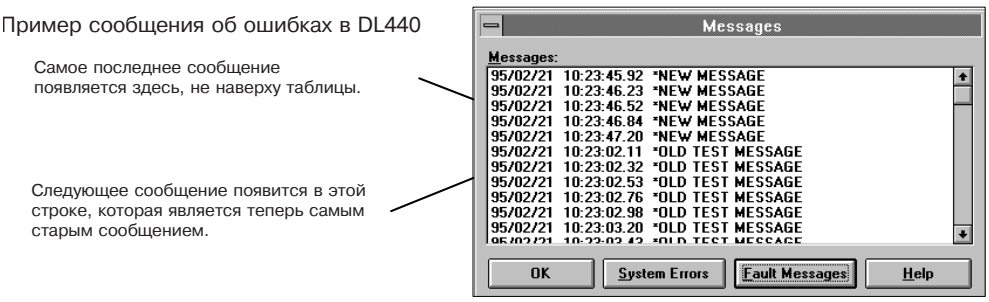
Системные ошибки и сообщения об ошибках

Процессоры DL405 обеспечивают возможность регистрации ошибок. Имеются некоторые заранее определенные сообщения об ошибках и коды системных ошибок, но Вы также можете использовать команду Fault, чтобы создать ваши собственные специфические сообщения. Процессор регистрирует ошибку, дату и время ошибки. Имеются две отдельных таблицы, которые хранят эту информацию.

- **Таблица системных ошибок (Error Code Table)** - и DL430, и DL440 имеет несколько предопределенных кодов системных ошибок. DL430 может фиксировать одновременно одну ошибку. DL440 может хранить до 32 ошибок в таблице ошибок. Когда происходит ошибка, она загружается в первую свободную ячейку. Следовательно, самая новая ошибка не может появиться в верхней строке таблицы. Если таблица заполнена и возникла новая ошибка, самая старая ошибка выталкивается (стирается) из таблицы, а новая ошибка вставляется в строку.
- **Таблица сообщений об ошибках (Error Message Table)** - DL430 и DL440 также позволяют формировать ваши собственные коды ошибок и сообщения. Они называются сообщениями об ошибках. Если у вас DL430, Вы можете формировать только цифровые коды ошибок. При DL440 Вы можете формировать коды ошибок или до 16 сообщений об ошибках, которые могут содержать до 23 алфавитно-цифровых символов. В любом случае Вы можете иметь до 16 сообщений или кодов, показанных в таблице. Когда сообщение возникло, оно помещается в первую свободную ячейку таблицы. Следовательно, самая новая ошибка не может появиться в верхней строке таблицы. Если таблица заполнена и возникла новая ошибка, самая старая ошибка выталкивается (стирается) из таблицы, а новая ошибка вставляется в эту строку.

На следующем рисунке показан пример таблицы сообщений об ошибках в DirectSOFT. Вы не сможете просмотреть всю таблицу одновременно на ручном программаторе. Сообщения автоматически появляются на дисплее ручного программатора в тот момент, когда они происходят. Сообщение будет отображаться на дисплее, пока выполняется команда Fault. Вы можете также использовать вспомогательную функцию AUX (5C), чтобы просмотреть все сообщения одновременно. (Более подробно об отображении на ручном программаторе см. далее)

Пример сообщения об ошибках в DL440



Существует несколько команд, которые могут использоваться в комбинации для создания этих кодов ошибок и сообщений.

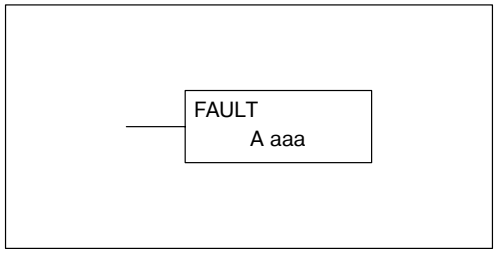
- FAULT- Fault (Ошибка)
- DLBL - Data Label (Метка данных)
- ACON - ASCII Constant (Константа ASCII)
- NCON - Numeric Constant (Цифровая константа)

На следующих нескольких страницах приводится более подробная информация об этих командах. В конце этого раздела имеются два примера, которые показывают, как использовать команды вместе.

Fault (FAULT)

X	√	√
430	440	450

В DL440 и DL450 команда Fault используется для отображения сообщения или цифрового кода на ручном программаторе, на экране DirectSOFT или на дисплее Интерфейса оператора DV-1000. Сообщение может иметь 23 знака и быть в виде данных V-памяти, цифровой константы или текста ASCII.



Чтобы отобразить значение в ячейке V-памяти, укажите в команде ячейку V-памяти. Чтобы отобразить данные в ASCII или цифровые данные Вы должны использовать команду DLBL (Метку данных) и команды ACON (ASCII константа) или NCON (цифровая константа) вместе с командой Fault. В этом случае вам необходимо определить значение константы (K) в команде Fault для соответствующей области метки данных, которая содержит команды ACON и/или NCON.

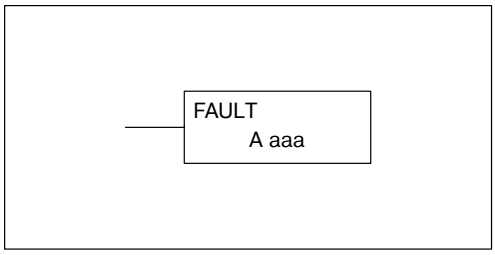
Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
V-память	V	Все (см. стр. 3-41)	Все (см. стр. 3-42)
Константа	K	1-FFFF	1-FFFF

Fault (FAULT)

X	√	√
430	440	450

В DL430 команда Fault используется для отображения цифрового кода на ручном программаторе, на экране DirectSOFT или на дисплее Интерфейса оператора DV-1000.

Код ошибки можно получить из ячейки V-памяти или можно его сконструировать, определяя константу в команде Fault.



Чтобы отобразить значение в ячейке V-памяти, укажите в команде ячейку V-памяти. Чтобы отобразить цифровую константу, определите значение константы (K) в команде.

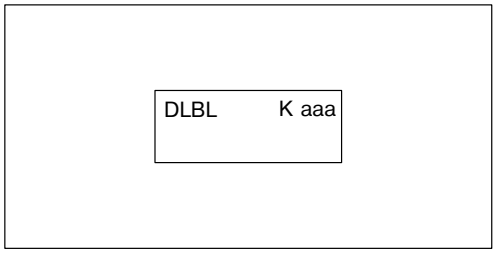
Тип данных операнда		Диапазон DL430
	A	**aaa**
V-память	V	Все (см. стр. 3-40)
Константа	K	1-FFFF

ПРИМЕЧАНИЕ: DL430 не поддерживает необходимых команд для создания алфавитно-цифровых сообщений об ошибках. Вы можете только создавать коды ошибок, получая код из ячейки V-памяти или определяя значение константы (K) в команде Fault.

Data Label (DLBL)

X	√	√
430	440	450

Команда Data Label отмечает начало ASCII / числовой области данных и обычно используется с командами NCON и ACON. Команды DLBL располагаются в программе после оператора End. В программе может использоваться максимум 64 команд DLBL. Команды NCON и ACON могут многократно использоваться в области DLBL. Примеры приводятся далее в данном разделе.



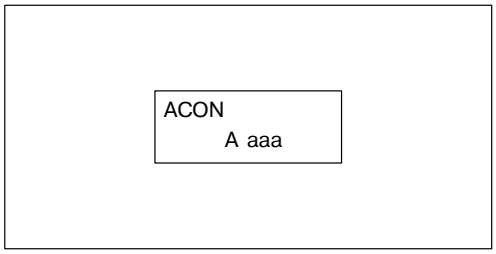
Тип данных операнда	Диапазон DL430	Диапазон DL440	Диапазон DL450
A	aaa	aaa	aaa
Константа K	—	1-FFFF	1-FFFF

ASCII Constant (ACON)

X	√	√
430	440	450

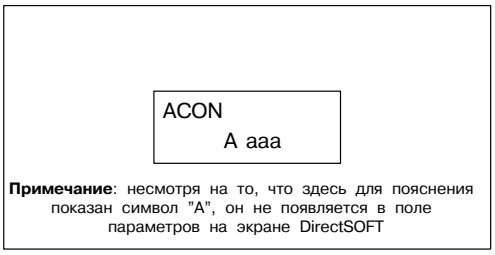
Команда ASCII Constant используется с командой DLBL, чтобы хранить ASCII текст для использования с другими командами. Команда используется по разному в ручном программаторе и в нашем программном обеспечении DirectSOFT.

Ручной программатор: В ручном программаторе могут храниться два символа ASCII в команде ACON. Если в команде ACON хранится только один символ, то в сообщении Fault первым знаком будет напечатан пробел.



Тип данных операнда	Диапазон DL430	Диапазон DL440	Диапазон DL450
A	aaa	aaa	aaa
ASCII A	—	0-9 A-Z	0-9 A-Z

Программное обеспечение DirectSOFT: Если Вы используете DirectSOFT, то Вы можете хранить в команде ACON до 40 символов. Вы можете также иметь более широкий диапазон поддерживаемых символов. (См. Руководство по DirectSOFT с полным списком доступных символов ASCII)



Примечание: несмотря на то, что здесь для пояснения показан символ "A", он не появляется в поле параметров на экране DirectSOFT

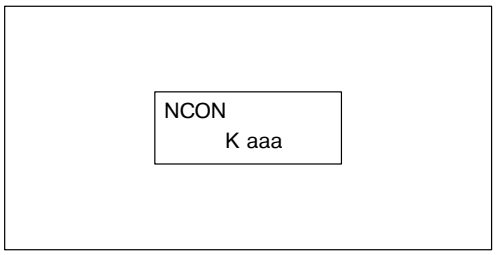
Тип данных операнда	Диапазон DL430	Диапазон DL440	Диапазон DL450
A	aaa	aaa	aaa
ASCII A	—	(См. Руководство по DirectSOFT)	(См. Руководство по DirectSOFT)

Вы можете задаться вопросом, почему же команды работают по-разному в двух различных инструментальных средствах программирования. Фактически команды не работают по-разному. В DirectSOFT 40-символов фактически разбиты на множители ACON, которые содержат по 2 символа каждый, когда они загружаются в процессор. Так, если Вы создаете программу и смотрите мнемонику программы на ручном программаторе (или даже с DirectSOFT), Вы можете увидеть множители ACON, которые содержат по 2 символа каждый. Примеры показаны на следующих страницах.

Numerical Constant (NCON)

X	√	√
430	440	450

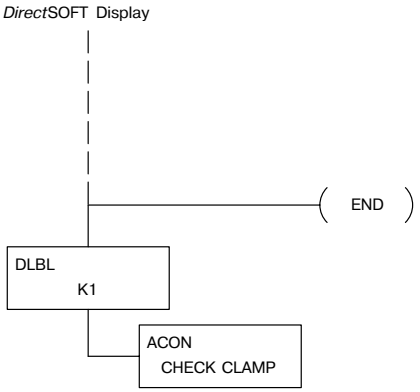
Команда Numerical Constant используется с командой DLBL, чтобы хранить шестнадцатиричный ASCII эквивалент цифровых данных для использования с другими командами. Две цифры могут быть сохранены в команде NCON.



Тип данных операнда		Диапазон DL440	Диапазон DL450
	A	**aaa**	**aaa**
Константа	K	0-FFFF	0-FFFF

Использование команд для формирования сообщения

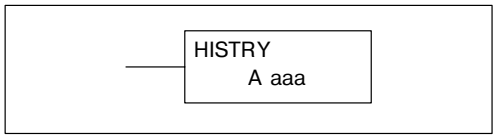
Команда Fault фактически копирует сообщения в соответствующую таблицу ошибок. Однако важно понять, как комбинируются команды DLBL, ACON, и NCON при формировании сообщения. DLBL помещается после команды END, которая является концом главной программы. Команды ACON и NCON помещаются внутри области DLBL. На рисунке справа показан пример.



History (HISTRY)

X	X	√
430	440	450

Команда History сохраняет информацию по истории событий в памяти ПЛК.

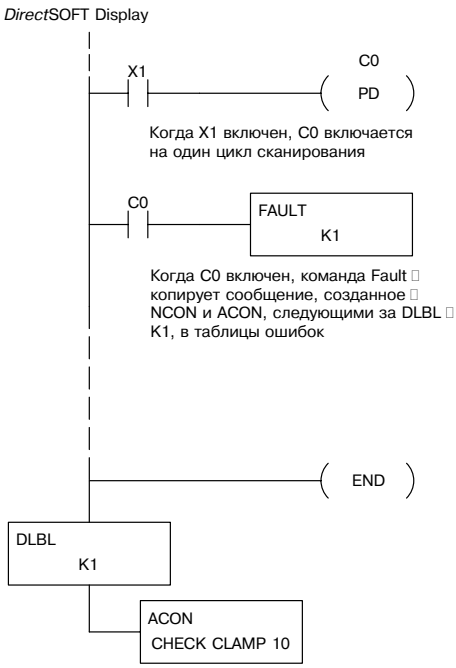


Важная информация по выполнению команды FAULT

Важно понять, как работают таблицы сообщений об ошибках и кодов ошибок, когда выполняется команда Fault. Каждый раз при выполнении команды код ошибки или сообщение вставляется в таблицу. Так как сканирование процессора - процесс быстрый, то есть возможность заполнения таблицы одинаковыми сообщениями. Во многих случаях это не желательно, так как лучше поддерживать хронологию ошибок. (Именно поэтому имеются 32 позиции в таблице кодов ошибок и 16 позиций в таблице сообщения об ошибках.).

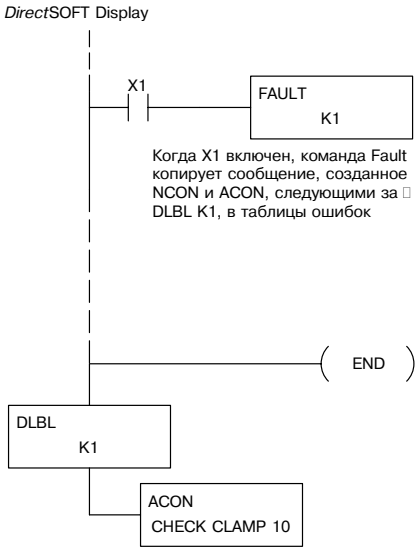
Например, пусть Вы исследуете переключатель ограничения (X1). Когда переключатель закрыт, Вы хотите фиксировать сообщение об ошибке (CHECK CLAMP 10) в таблице сообщений об ошибках. Реально переключатель ограничения может быть закрыт в течение нескольких секунд (или минут, или часов...). За это время процессор выполнит команду Fault неоднократно, так что вся таблица сообщений об ошибках будет заполнена одним и тем же сообщением.

Вместо того, чтобы использовать концевой переключатель для запуска команды Fault, используйте концевой переключатель, чтобы запускать реле управления, используемое в однотактном режиме (команда обмотки PD). Затем используйте реле управления как вход к команде Fault. Так как Fault теперь запускает PD, который включен только в одном цикле сканирования, сообщение будет скопировано в таблицу только один раз. Это сохранит хронологию старых сообщений.



ПРИМЕЧАНИЕ: Этот метод не будет работать корректно с ручным программатором. Вы должны использовать входной контакт (в этом примере X1), чтобы непосредственно вызвать команду Fault.

ПРИМЕЧАНИЕ: Этот метод не будет работать корректно с ручным программатором. Сообщение никогда не появится на экране. Вы должны использовать входной контакт (в этом примере X1), чтобы вызвать непосредственно команду Fault. См. пример на рисунке.

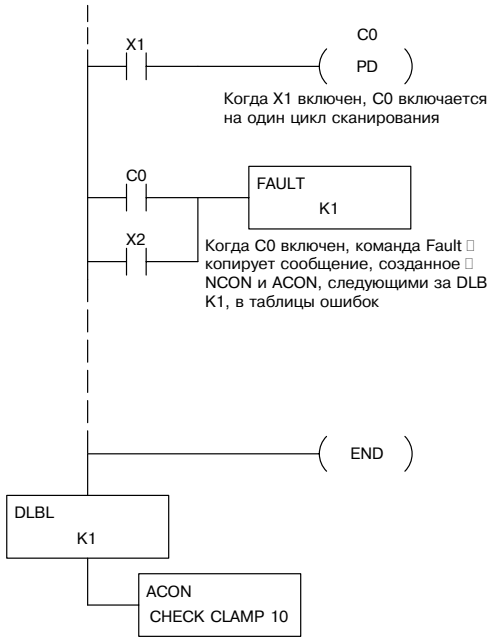


Очевидно, Вы могли бы комбинировать другие команды с использованием команды PD, чтобы разработать логику, которая работала бы и для ручного программатора и для DirectSOFT. Например, если это возможно в вашей прикладной программе, Вы можете иметь два входных контакта, которые могут вызывать команду FAULT.

Рассмотрите показанный пример справа. Если X1 включается, то сообщение будет только скопировано в таблицу один раз. Вы можете увидеть сообщение, используя DirectSOFT, но Вы не увидите его на дисплее ручного программатора.

Если включается X2, то одно сообщение вероятно заполнит всю таблицу. Также оно автоматически появится на экране ручного программатора.

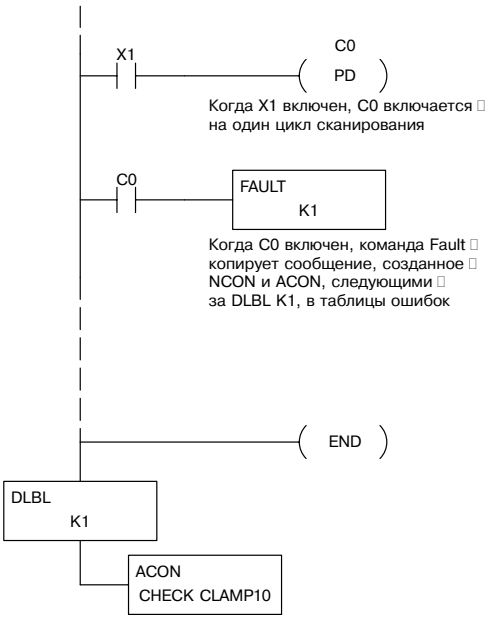
*Direct*SOFT Display



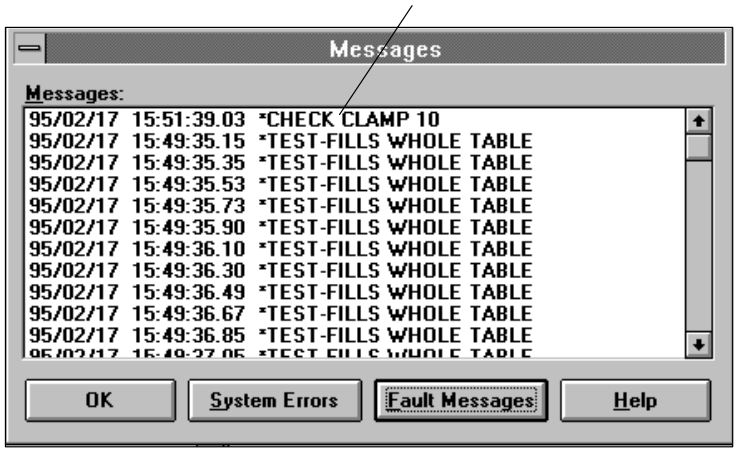
Пример для DirectSOFT

Так как в DirectSOFT Вы можете вводить большее число символов в команду ACON, то очень просто сформировать все сообщение в одной команде ACON.

*Direct*SOFT Display



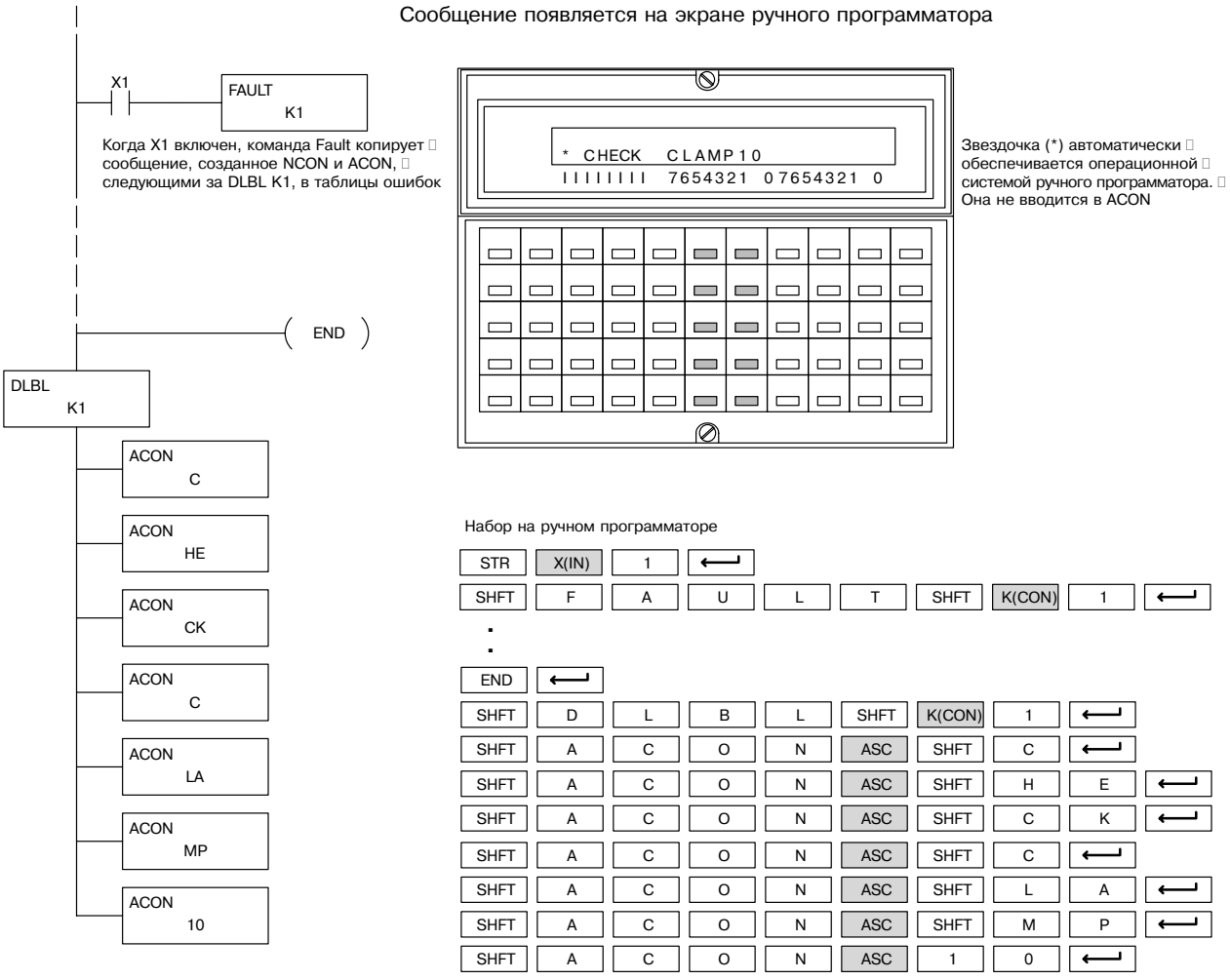
PD C0 допускает один ввод в таблицу



Пример для ручного программатора

Так как Вы можете вводить командой ACON только два символа, используя ручной программатор, то для получения тех же самых результатов программа будет длиннее. Обратите внимание также на то, что мы организовали содержимое команды ACON немного по-другому. Например, мы включили в первый ACON только символ "C". Мы должны придерживаться этого способа, чтобы получить корректное сообщение. (Напоминаем, что одиночный символ, введенный в ACON с ручного программатора, будет иметь перед собой пробел. Поэтому, если ваши сообщения не содержат четного числа символов, вам, вероятно, придется немного подумать, чтобы получить интервал только справа.) Также мы должны использовать X1, чтобы непосредственно запускать команду FAULT. Если мы не сделаем этого, сообщение автоматически никогда не появится на экране ручного программатора.

Очистка сообщений



Используйте разные методы очистки системных ошибок и сообщений об ошибках.

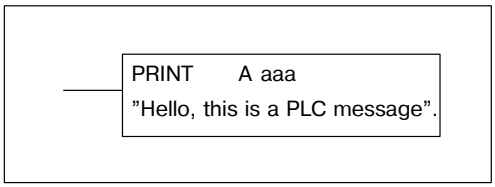
- Системные ошибки - инициализацией системной памяти процессора (scratchpad).
- Таблица сообщений о неисправностях - очисткой программной памяти процессора.

ПРЕДУПРЕЖДЕНИЕ: При инициализации оперативной памяти Вы также можете удалить какие- либо области сохранения памяти, которые Вы могли изменить.

Print Message (PRINT)

X	X	√
430	440	450

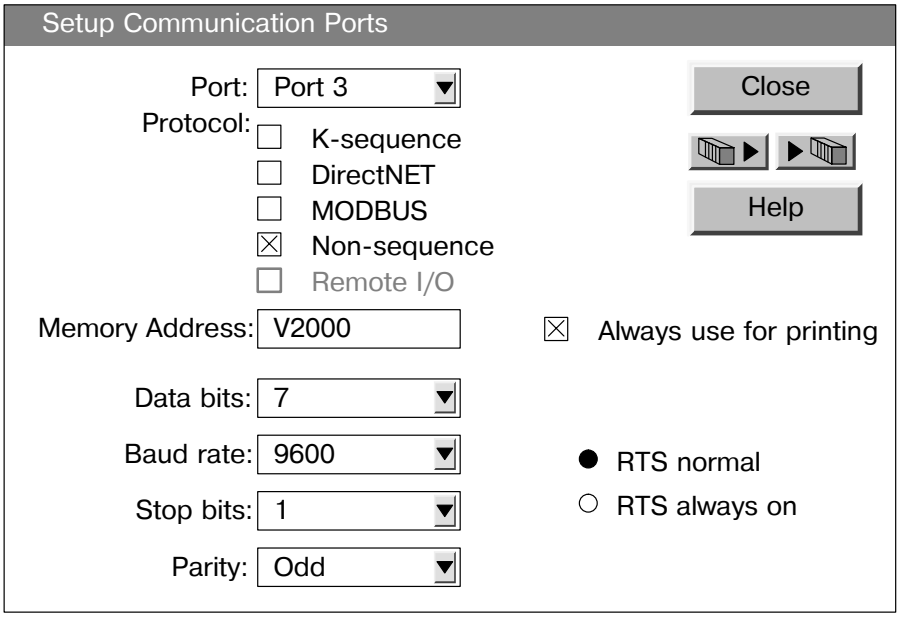
Команда Print Message печатает вложенный текст или сообщение с переменными текст / данные в заданный коммуникационный порт (1, 2, или 3 в процессоре DL450), который должен быть сконфигурирован.



Тип данных операнда		Диапазон DL430
	A	aaa
Константа	K	1,2 или 3

В спецификации процессора в главе 3 показано, что порты DL450 способны работать с несколькими протоколами. Чтобы сконфигурировать порт с помощью ручного программатора, используйте AUX 56 и следуйте подсказки, приведенные ниже на этой странице. Чтобы сконфигурировать порта в DirectSOFT, войдите в меню PLC, затем в Setup, затем в Setup Secondary Comm Port.

- Port (Порт): В списке номеров портов выберите "Port 2".
- Protocol (Протокол): Щелкните по метке слева от "Non-sequence", и затем Вы увидите диалоговое окно, показанное ниже.



- Memory Address (Адрес памяти): Выберите адрес V-памяти для DirectSOFT для хранения информации об установках порта. Вам понадобится зарезервировать 9 слов в V-памяти для этой цели. Выберите "Always use for printing" ("Всегда использовать для печати"), если это понадобится.
- Baud Rate (Скорость передачи в бодах): Выберите скорость передачи, которая соответствует вашему принтеру.
- Stop Bits, Parity (Стоповые биты, проверка четности): Выберите число стоп-битов и четность в соответствии с вашим принтером.
- RTS (Готовность к передаче): Выберите соответствующую опцию готовности к передаче.

Затем щелкните показанную клавишу, чтобы послать установки порта 3 в процессор, и щелкните Close. Потом посмотрите главу 3 с информацией по монтажным схемам, чтобы соединить ваш принтер с DL450.

Порты 1 и 2 в DL450 имеет стандартные RS232 сигналы и должены работать с большинством принтеров по последовательному входу.

Текстовый элемент - используется для печати символьных строк. Символьные строки определяются как последовательности символов, заключенных в двойные кавычки. Двузначное шестнадцатиричное число, стоящее после знака $, означает 8-битовый ASCII символ. Также, два знака, стоящие после знака $ интерпретируются в соответствии со следующей таблицей:

#	Знаковый код	Описание
1	$$	Знак доллара ($)
2	$є	Двойные кавычки («)
3	$L или $l	Перевод строки (LF)
4	$N или $n	Возврат каретки перевода строки (CRLF)
5	$P или $p	Перевод страницы
6	$R или $r	Возврат каретки (CR)
7	$T или $t	Табуляция

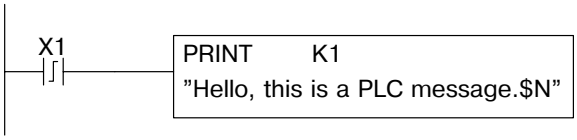
Следующие примеры показывают различные синтаксические соглашения и длину строки, выводимой на принтер.

Пример:

” ”	Длина 0 без символа
”A”	Длина 1 с символом ”A”
” ”	Длина 1 с ”пробелом”
” $” ”	Длина 1 с двойными кавычками
” $ R $ L ”	Длина 2 с одним CR и одним LF
” $ 0 D $ 0 A ”	Длина 2 с одним CR и одним LF
” $ $ ”	Длина 1 с одним символом $

При печати обычной строки текста, вам требуется включить двойные кавычки перед и после текстовой строки. Отметьте, что DirectSOFT не показывает синтаксические ошибки в команде PRINT. Поэтому важно проверять ваши данные во время разработки приложения до команды PRINT.

В следующем примере печатается сообщение через порт 1 Мы используем PD контакт, который выполняет команду Message только в одном цикле сканирования. Отметьте знак $N в конце сообщения, который производит возврат каретки / перевод строки на принтере. При этом принтер готовится к печати следующей строки, стартуя с левого края.



Печатает сообщение в порту 1, когда X1 делает переход ВЫКЛ-ВКЛ

Элемент V-памяти - используется для печати содержимого V-памяти в формате целых или вещественных чисел. Используйте номер V-памяти или номер V-памяти с символом ":" и типом данных. Типы данных показаны в таблице ниже. Коды символов должны писаться прописными буквами.

#	Знаковый код	Описание
1	нет	16-битовый двоичный (десятичное число)
2	: B	4-знаковый BCD
3	: D	32-битовый двоичный (десятичное число)
4	: D B	8-знаковый BCD
5	: R	Число с плавающей точкой (вещественное число)
6	: E	Число с плавающей точкой (вещественное число с экспонентой)

Пример:

V2000 Печать двоичных данных в V2000 для десятичного числа

V2000 : B Печать двоично-десятичных данных в V2000

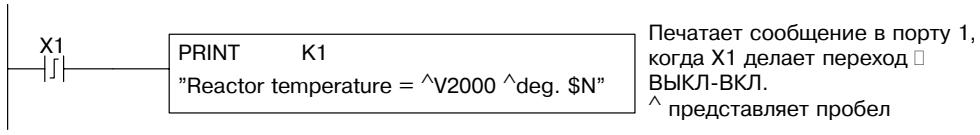
V2000 : D Печать двоичного числа в V2000 и V2001 для десятичного числа

V2000 : D B Печать двоично-десятичных данных в V2000 и V2001

V2000 : R Печать числа с плавающей точкой в V2000/V2001 как вещественного числа

V2000 : E Печать числа с плавающей точкой в V2000/V2001 как вещественного числа с порядком

Пример: В следующем примере печатается сообщение, содержащее текст и переменную. "Температура реактора" обозначает данные, которые находятся в V2000. Вы можете использовать квалификатор : B после V2000, если, например, данные в двоично-десятичном формате. Финальная строка добавляет единицы градусов в строку текста, а $N добавляет возврат каретки / перевод строки.



Текстовый элемент V-памяти - используется для печати текста, хранимого в V-памяти. Используйте знак %, за которым следует число символов после номера V-памяти для представления текста. Если Вы присвоили "0" числу символов, функция печати начнет отсчет знаков с первой ячейки. Затем она начнет следующую ячейку V-памяти и будет читать это число кодов ASCII для текста из памяти.

Пример:

V2000 % 16 Будет печататься 16 символов в V2000-V2007

V2000 % 0 Будут печататься символы от V2001 до Vxxxx (определяется числом в V2000)

Битовый элемент - используется для печати состояния указанных битов в V-памяти или битов реле. Битовый элемент может быть назначен указанной точкой (.) и номер бита предшествует номеру V-памяти или номеру реле. Описание типа выхода показано в таблице ниже.

#	Формат данных	Описание
1	нет	Печатает 1 при состоянии ВКЛ., и 0 – при состоянии ВЫКЛ.
2	: BOOL	Печатает «TRUE» при состоянии ВКЛ., и «FALSE» – при состоянии ВЫКЛ.
3	: ONOFF	Печатает «ON» при состоянии ВКЛ., и «OFF» – при состоянии ВЫКЛ.

Пример:

V2000 . 15	Печать состояния бита 15 в V2000 в формате 1/0
C100	Печать состояния C100 в формате 1/0
C100 : BOOL	Печать состояния C100 в формате TRUE/FALSE
C100 : ON/OFF	Печать состояния C100 в формате ON/OFF
V2000 . 15 : BOOL	Печать состояния бита 15 в V2000 в формате TRUE/FALSE

Максимальное число символов, которое Вы можете напечатать - 128. Число символов для каждого элемента показано в таблице ниже.

Тип элемента	Максимальное число знаков
Текст, 1 знак	1
16-битовый двоичный	6
32-битовый двоичный	11
4-знаковый BCD	4
8-знаковый BCD	8
Плавающая точка (вещественное число)	12
Плавающая точка (вещественное число с экспонентой)	12
V-память/текст	2
Бит (1/0 формат)	1
Бит (TRUE/FALSE формат)	5
Бит (ON/OFF формат)	3

Мнемоника на ручном программаторе - ”PRINT”, следом поле DEF.

Флаги специальных реле SP112 и SP117 показывают состояние портов процессор DL450 (занят, или ошибка связи). Для подробного описания смотрите приложение по специальным реле.

ПРИМЕЧАНИЕ: Вы должны использовать соответствующие специальные реле в соединении с командой PRINT, чтобы гарантировать, что программа не будет пытаться печатать в порт, который все еще занят предшествующими командами PRINT, WX или RX.

Программирование с помощью барабанного командоаппарата (только для DL450)

6

В этой главе……

Введение

Назначение

X	X	√
430	240	250

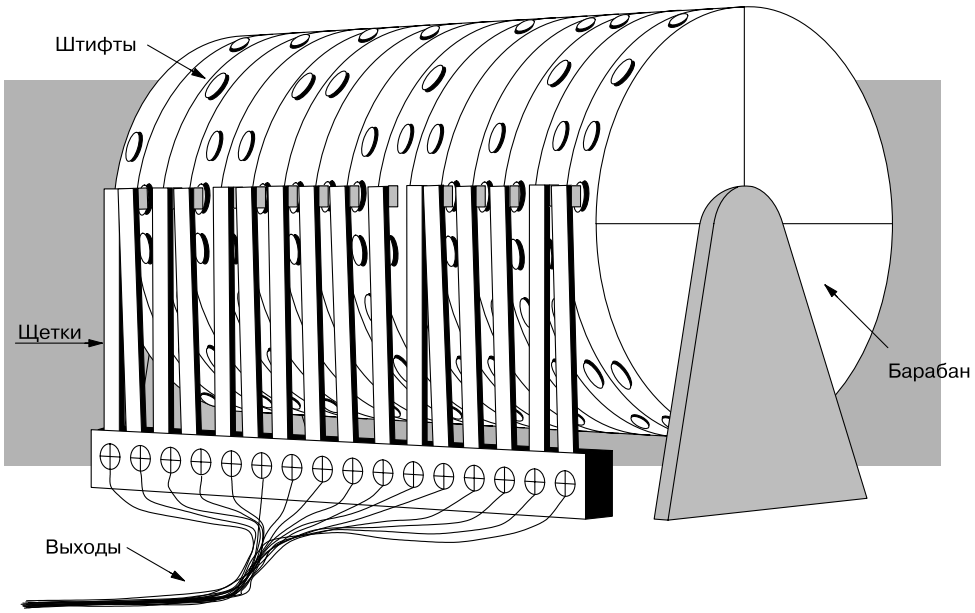
Четыре команды процессора DL250 электронным образом имитируют электромеханический барабанный командоаппарат. Команды предполагают небольшие отличия от принципа работы электромеханического устройства.

Команды барабанного командоаппарата наилучшим образом подходят для циклических процессов, состоящих из конечного числа шагов. Они могут простым образом выполнить несколько цепей релейной логики. Поэтому барабанные командоаппараты могут сэкономить время при программировании и отладке.

Терминология

Мы вводим некоторую терминологию, связанную с командами барабанного командоаппарата, описывая изображенный ниже оригинальный электромеханический барабан. На изогнутой поверхности механического барабана обычно располагаются штифты. Штифты размещены в соответствии с **конфигурацией**, представляющей набор действий, нужных для управления машиной. Мотор или соленоид в заданные моменты времени точно поворачивает барабан. При вращении стационарные щетки касаются штифтов (есть — вкл, нет — выкл). Это взаимодействие создает или разрывает электрический контакт со щетками, создавая электрические выходы барабанного командоаппарата. Выходы передаются к устройству машины для управления включением/выключением.

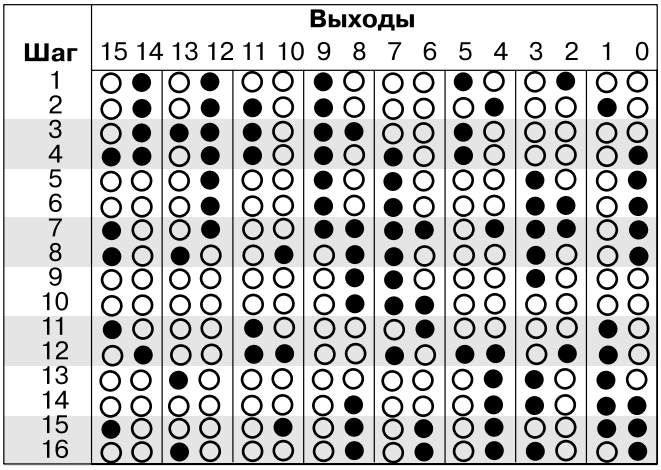
За один оборот барабан проходит конечное число положений, называемых **шагами**. Каждый шаг представляет собой один шаг процесса. При включении питания барабан **возвращается** к определенному шагу. Барабан вращается от одного шага к другому в зависимости от **таймера** или некоторого внешнего **события**. При определенных условиях оператор машины может вручную увеличивать шаг барабана, используя механизм барабана **управление скачками механизмом барабана**. Контакт каждой щетки создает уникальную конфигурацию включения/выключения, называемую **последовательностью** и предназначенную для управления конкретной машиной. Так как барабан является круглым, последовательность автоматически повторяется с каждым оборотом. Прикладные задачи могут существенно отличаться друг от друга, поэтому конкретный барабан может делать один оборот за секунду или за целую неделю.



Электронные барабанные командоаппараты реализуют возможности механических барабанов, дополняя их. Например, они обладают возможностью предварительной установки, что невозможно для механических барабанов: функция **предварительной установки** обеспечивает переход по команде от текущего шага к любому другому!

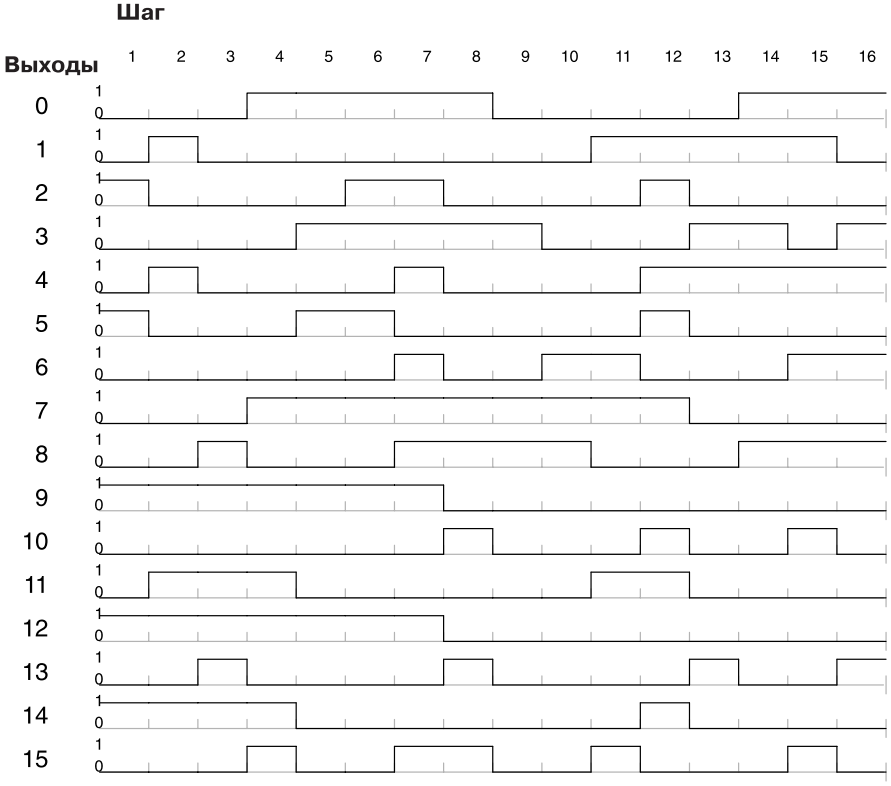
Схематическое представление барабанного командоаппарата

При редактировании электронный барабанный командоаппарат представляется в **Direct**SOFT (и в данном справочном руководстве) в виде схемы. Представьте, что полый цилиндр барабана разрезали между двумя рядами штифтов, а затем развернули. Тогда барабан будет выглядеть, как показано ниже. Каждый ряд с номером от 1 до 16 представляет собой шаг. Каждый столбец представляет выход с номером от 0 до 15 (соответствуя 16-битовому слову). Сплошные кружки обозначают штифты (состояние вкл.) механического барабана, а пустые кружки — отсутствие штифтов (состояние выкл.).



Последовательности выходов

Механический барабанный командоаппарат реализует последовательности управляющих изменений своих электрических выходов. На следующем рисунке показана последовательность управляющих воздействий вкл./выкл., создаваемая приведенной выше конфигурацией барабана. Сравните их, и вы увидите, что они эквивалентны! Если вы заметили эту эквивалентность, то вы начали понимать работу команд барабанного командоаппарата.



Шаговые переходы

Типы команд барабанного командоаппарата

Для процессора DL450 существует четыре типа команд барабанного командоаппарата:

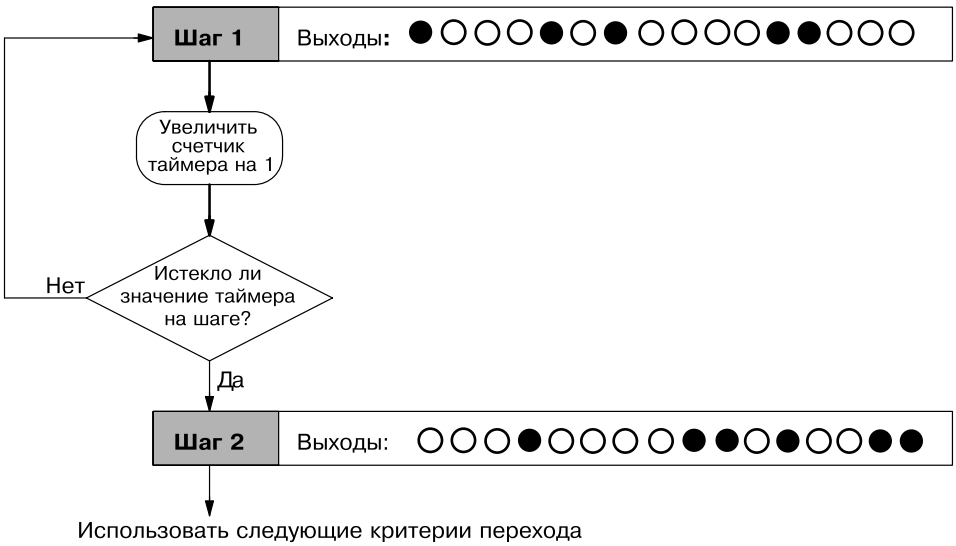
- Барабаны с дискретными выходами и переходом по времени (DRUM)
- Барабаны с дискретными выходами и переходом по времени и по событию (EDRUM)
- Маскированные барабаны с дискретными выходами (MDRUMD)
- Маскированные барабаны с выходами по словам и переходом по событию (MDRUMW)

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

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

Переходы только по таймеру

Барабанные командоаппараты переходят от шага к шагу в зависимости от времени и/или внешнего события (входа). Все четыре типа барабанных командоаппаратов обеспечивают шаговые переходы по таймеру, и три из них также обеспечивают переходы по событию. На приведенном ниже рисунке показано, как работают переходы только по таймеру.



В каждом из шагов барабан остается в течение конкретного времени (программируемого пользователем). Временной масштаб таймера задается программно от 0.01 с до 99.99 с. Это определяет разрешение или длительность каждого «тика» часов (единицы счета). Для всех шагов используется общий масштаб, но для каждого шага программно задается свое уникальное количество единиц счета. Конкретное время, в течение которого командоаппарат остается на конкретном шаге, определяется по формуле:

Время на шаг = 0.01 секунд x Временной масштаб x Единиц счета на шаг

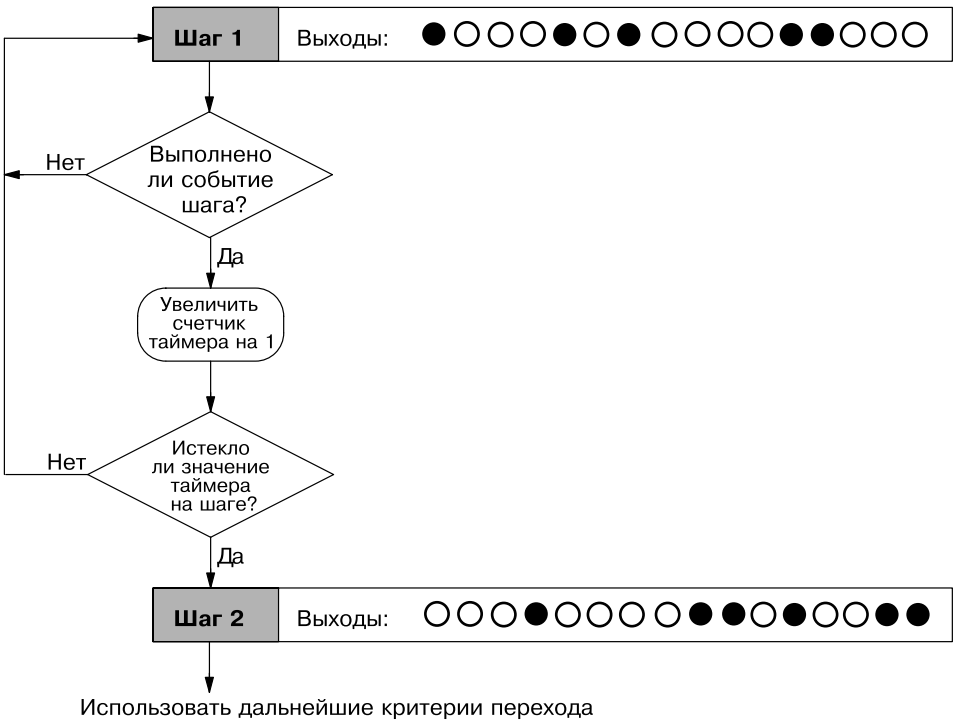
Например, если вы задаете для шага 1 5-секундный временной масштаб и 12 единиц счета, то барабанный командоаппарат будет находиться 60 секунд в состоянии Шага 1. Максимальная длительность каждого шага определяется формулой:

Максимальное время на шаг = 0.01 секунд x 9999 x 9999 = 999800 секунд = 277.7 часа = 11.6 дня

ПРИМЕЧАНИЕ. При первоначальном выборе разрешения временного масштаба хорошим значением на практике является 1/10 длительности самого короткого шага вашего барабанного командоаппарата. Это позволит оптимизировать длительность шага 10-процентными приращениями. Другие шаги с большей длительностью можно оптимизировать даже с меньшими (в процентном отношении) приращениями. Кроме того, обратите внимание, что команда барабанного командоаппарата выполняется при одном сканировании процессора только один раз. Следовательно, бессмысленно задавать временной масштаб барабана намного меньше времени сканирования процессора.

Переходы по таймеру и событию

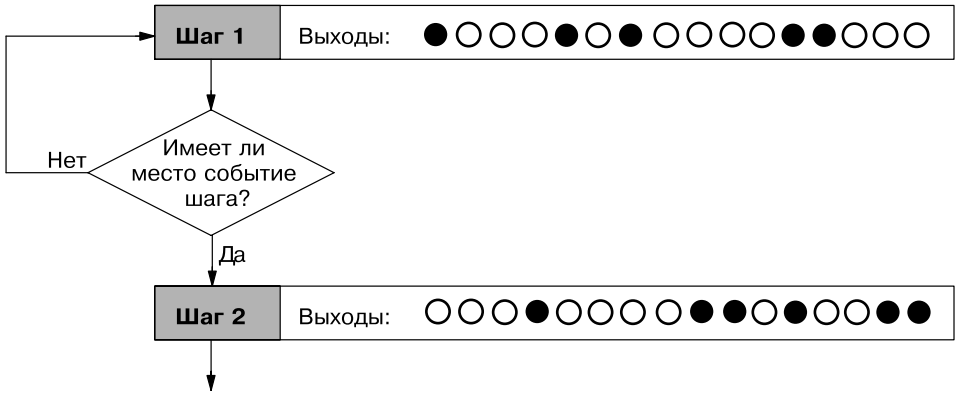
Барабанные командоаппараты с переходами по времени и по событию зависят от времени и/или от внешних событий. На приведенном рисунке показано выполнение шагов для таких командоаппаратов.



Когда барабанный командоаппарат переходит к шагу 1, устанавливается показанная конфигурация выходов. Начинается опрос внешнего входа, заданного для данного шага. Можно определить входы событий как типы дискретных точек X, Y или C. Предположим, мы выбрали X0 в качестве входа событий для шага 1. Когда X0 находится в состоянии ВКЛ, выполнен критерий события, и счетчик таймера увеличивается на 1. Счетчик таймера увеличивается, пока событие остается истинным. При истечении единиц счета для шага 1 командоаппарат переходит к шагу 2. Выходы немедленно изменяются в соответствии с новой конфигурацией, соответствующей шагу 2.

Переходы только по событию

Барабанные командоаппараты, работающие по времени и по событию, не обязательно должны использовать на каждом шаге одновременно и критерий события и критерий таймера. Можно выбрать один вариант и даже имея смешанные типы переходов в различных шагах командоаппарата. Например, вы можете захотеть, чтобы на шаге 1 выполнялся переход только по событию, на шаге 2 - только по времени, а на шаге 3 — и по времени, и по событию. Более того, вы можете выбрать для использования только часть из 16 шагов и только часть из 16 выходов.



Задание счетчиков

Каждая команда барабанного командоаппарата использует ресурсы четырех счетчиков в процессоре. При программировании команды барабанного командоаппарата сначала выбирается значение первого счетчика.Командоаппарат автоматически использует следующие три счетчика. Бит счетчика, связанный с первым счетчиком, включается, когда командоаппарат завершает свой цикл, и сбрасывается при сбросе командоаппарата. Эти значения счетчиков и бит счетчика точно показывают ход выполнения команды барабанного командоаппарата и могут быть отслежены вашей программой.

Предположим, вы программируете таймер на 8 шагов, и это число счетчика находится в CT10 (не забудьте, нумерация счетчика является восьмеричной). Использование счетчика показано справа. Правый столбец содержит типичные значения, поясняемые ниже.

Задание счетчиков

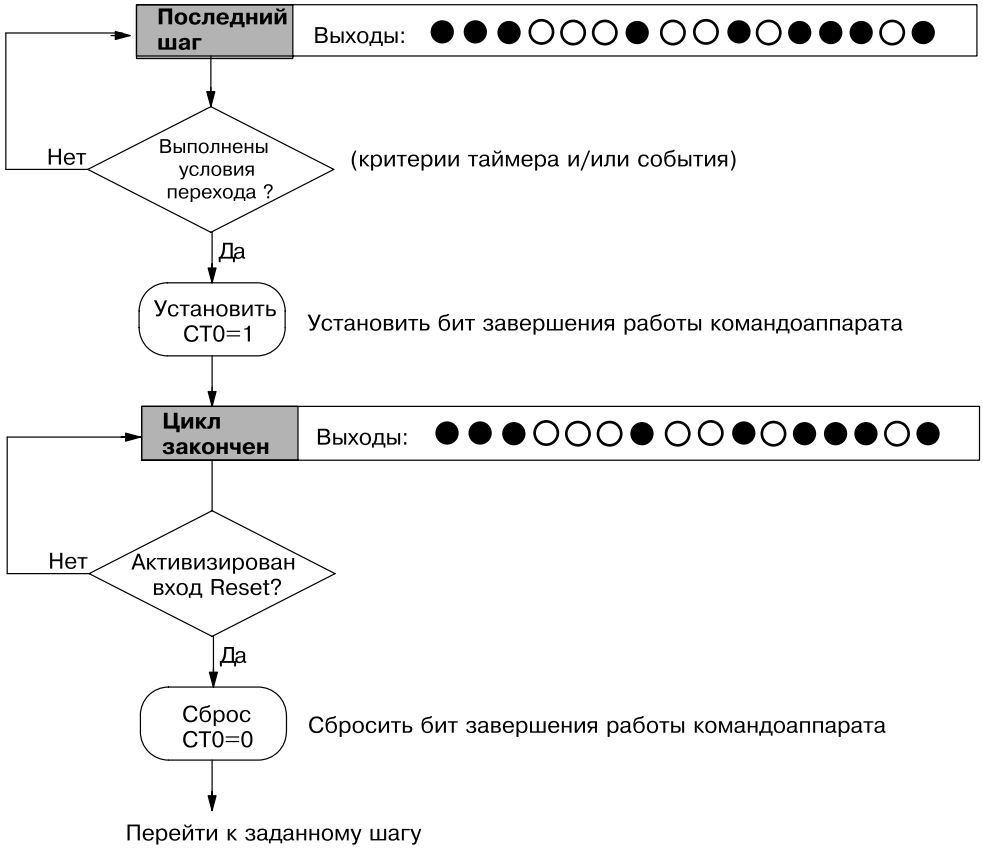
CT10	Единиц счета в шаге	V1010	1528
CT11	Счетчик таймера	V1011	0200
CT12	Заданный шаг	V1012	0001
CT13	Текущий шаг	V1013	0004

CT10 показывает, что вы находитесь на 1528-й единице счета текущего шага (шаг 4, что показано в CT13). Если мы задали, что шаг 4 содержит 3000 единиц счета, шаг выполнен более чем наполовину. CT11 — это счетчик таймера, единица которого соответствует 0.01 с. Следовательно, каждое изменение младшего значащего бита соответствует 0.01 с. Значение 200 означает, что вы находитесь на текущей единице счета (1528) 2 секунды (0.01 x 200). Наконец, CT12 содержит номер предварительно установленного шага, заданный для команды барабанного командоаппарата. В данном случае при активизации сброса командоаппарат переходит на шаг 1. Значение CT12 нельзя изменить без редактирования программы. Бит счетчика CT10 устанавливается при завершении цикла барабанного командоаппарата и сбрасывается при сбросе командоаппарата.

Завершение последнего шага

Номер последнего шага в последовательности барабанного командоаппарата может быть любым, так как допустимы неполные барабанные командоаппараты (см. следующий рисунок). Когда выполнены условия перехода для последнего шага, командоаппарат устанавливает бит счетчика, соответствующий счетчику, заданному в команде барабанного командоаппарата (например, CT0). Затем командоаппарат переходит в завершающее состояние «цикл закончен». Выходы командоаппарата остаются в конфигурации, определенной для последнего шага (включая любую логику маски выходов). После окончания цикла барабанного командоаппарата входы Start (Старт) и Jog (Скачок) перестают действовать.

Командоаппарат выходит из состояния «цикл закончен» при активизации входа Reset (Сброс) (или при переходе в рабочий режим программы). При этом снова устанавливается бит завершения (например, CT0), а затем барабан переходит прямо к заданному номеру шагу.



Обзор работы барабанного командоаппарата

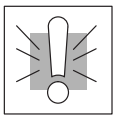
Блок-схема команд барабанного командоаппарата

Помимо конфигурации барабанного командоаппарата, команда использует различные входы и выходы (см. приведенный рисунок).

В команде барабанного командоаппарата используется ряд входов для управления шагами, основного управления барабаном. Ниже перечислены входы и их функции:

- **Start (Старт)** — вход Start действует, только если отключен Reset. Когда Start включен, для перехода по времени используется таймер барабанного командоаппарата, а для переходов по событию командоаппарат выполняет поиск входного события. Когда Start выключен, командоаппарат фиксируется в своем текущем состоянии (Reset должен оставаться выключенным), а его выходы сохраняют свою текущую конфигурацию ВКЛ/ВЫКЛ.
- **Jog (Скачок)** — вход Jog действует, только если отключен Reset (Start может быть включен или выключен). При каждом переходе Выкл/Вкл вход Jog переводит барабанный командоаппарат к следующему шагу. Обратите внимание, что вход Jog отсутствует только у основного командоаппарата, работающего по таймеру.
- **Reset (Сброс)** — приоритет входа Reset выше, чем у входа Start. Когда Reset включен, барабанный командоаппарат возвращается к заданному шагу. Когда Reset выключен, вход Start работает как обычно.
- **Preset step (Предварительно установленный шаг)** — определяемый вами номер шага от 1 до 16 (обычно шаг 1). Барабан переходит к этому шагу при каждом включении Reset и в начале работы процессора.

- **Counts/step (Единиц счета/шаг)** — количество единиц счета таймера, в течение которых барабанный командоаппарат находится на каждом шаге. Для каждого шага задается свое количество единиц счета. Однако для командоаппаратов с переходом по времени/событию, программирование этого параметра необязательно.
- **Timer Value (Значение таймера)** — текущее значение таймера единиц счета/шаг.
- **Counter # (Номер счетчика)** — номер счетчика задает первый из четырех последовательных счетчиков, используемых барабанным командоаппаратом для управления шагами. Они позволяют следить, как командоаппарат проходит свой цикл управления.
- **Events (События)** — дискретная точка типа X, Y, C, GX, GY, S, C, CT или SP служит в качестве входа для перехода шага. Однако для барабанных командоаппаратов с переходом по времени/событию программирование события необязательно.



Предупреждение. Выходы барабана доступны в любой момент, пока процессор находится в рабочем режиме. Вход Start не должен быть включенным, а вход Reset не отключает выходы. При переходе в рабочий режим выходы барабанного командоаппарата автоматически включаются и отключаются в соответствии с конфигурацией предварительно установленного шага. При этом учитывается влияние используемой маски выходов.

Состояние регистров барабанного командоаппарата при включении питания

Для понимания вашего приложения важен выбор шага, который будет стартовым при включении питания и при переходе в рабочий режим. Пожалуйста, взгляните на следующую схему. Если память счетчика настроена как несохраняемая, барабанный командоаппарат при каждом включении или при переходе в рабочий режим из программного инициализируется одинаковым образом. Однако если память счетчика настроена как сохраняющая значение (cretentive), командоаппарат остается в своем прежнем состоянии.

Номер счетчика	Функция	Инициализация при включении питания	
		Без сохранения значения	С сохранением значения
CT(n)	Текущая единица счета шага	Инициализация = 0	Используется предыдущее состояние (без изменений)
CT(n + 1)	Значение счетчика таймера	Инициализация = 0	Используется предыдущее состояние (без изменений)
CT(n + 2)	Предварительно установленный шаг	Инициализация = № предварительно установленного шага	Используется предыдущее состояние (без изменений)
CT(n + 3)	№ текущего шага	Инициализация = № предварительно установленного шага	Используется предыдущее состояние (без изменений)

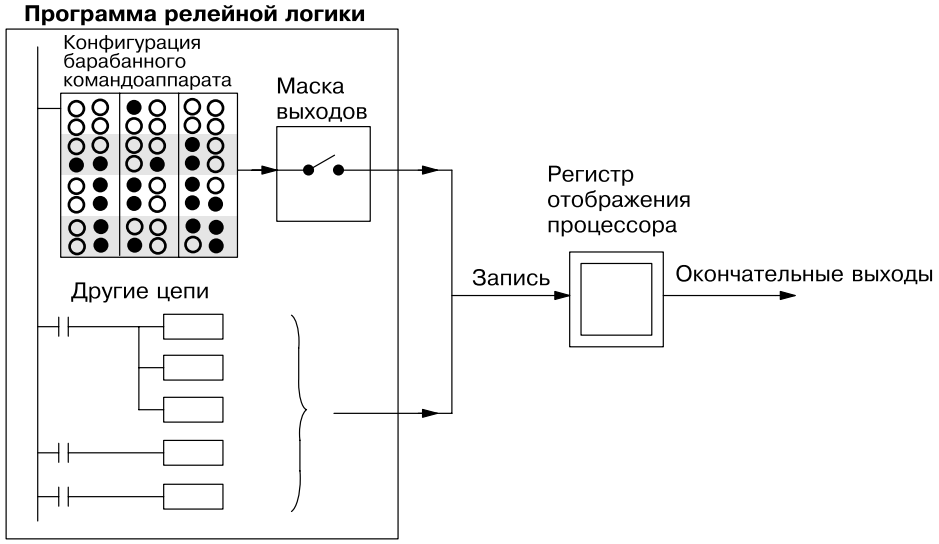
При использовании варианта без сохранения при включении обычно должны перезапускаться приложения с относительно короткими длительностями циклов барабанного командоаппарата. Для приложений с относительно длинными циклами командоаппарата может понадобиться восстанавливать предыдущую точку остановки операций, используя вариант с сохранением. Значением по умолчанию является вариант с сохранением. Это означает, что оперативная V-память будет сохранять свое значение при инициализации системной памяти (scratchpad).

Работа маски выходов

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

- **MDRUMD** — маскированные барабаны с дискретными выходами
- **MDRUMW** — маскированные барабаны с выходами по словам, меняющимися по событию

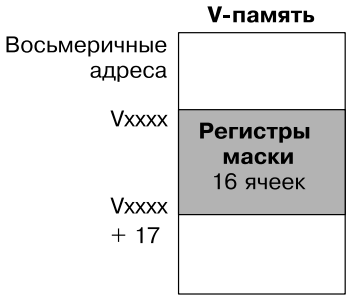
Маска выходов — это просто управление барабанным командоаппаратом с помощью побитового включения/выключения, записываемого в регистр отображения шестнадцати выходов (см. приведенный рисунок). Регистр отображения содержит формальный текущий статус всех точек ввода/вывода. В конце каждого сканирования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



Практическое использование масок выходов для барабанного командоаппарата включает:

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

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

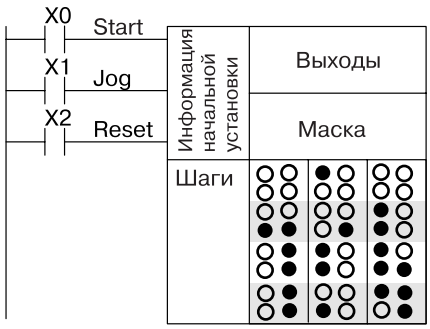


Если бит маски = 1, барабанный командоаппарат управляет точкой выхода. Если бит маски=0, командоаппарат не может записывать в регистр отображения, поэтому выход остается в своем текущем состоянии.

Техника управления барабанным командоаппаратом

Управляющие входы барабанного командоаппарата

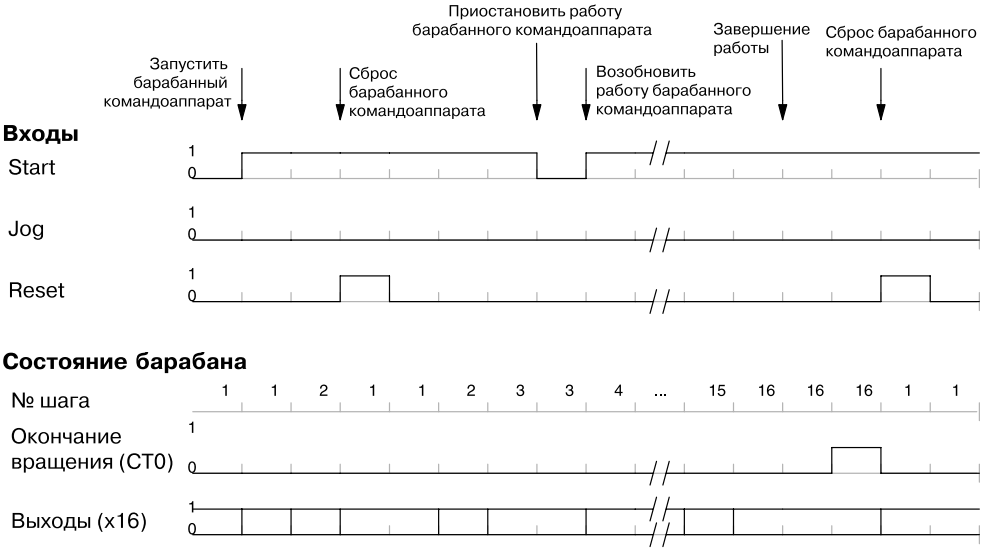
Теперь мы готовы собрать воедино предыдущие понятия и продемонстрировать управление с помощью команд барабанного командоаппарата. На рисунке справа показана упрощенная общая команда. Программа управляет входами Start, Jog и Reset. Бит первого счетчика командоаппарата (CT0, например) сообщает об окончании цикла командоаппарата.



На приведенной временной диаграмме показана некоторая входная последовательность барабанного командоаппарата с переходами по времени и реакция командоаппарата. После перехода в рабочий режим процессор записывает в регистр номера шага номер предварительного установленного шага (обычно шаг 1). Командоаппарат начинает работать при высоком уровне входа Start, проводя поиск события и/или запуская таймер единиц счета (в зависимости от типа командоаппарата и начальных значений).

После того как барабанный командоаппарат переходит к шагу 2, при включенном состоянии входа Start включается Reset. Так как Reset обладает приоритетом по сравнению с входом Start, командоаппарат возвращается к предварительно установленному шагу (шаг 1). Обратите внимание, что командоаппарат сохраняет состояние предварительно установленного шага, пока включен Reset, и что этот шаг не выполняется (в ответ на событие или по таймеру), пока Reset не будет выключен.

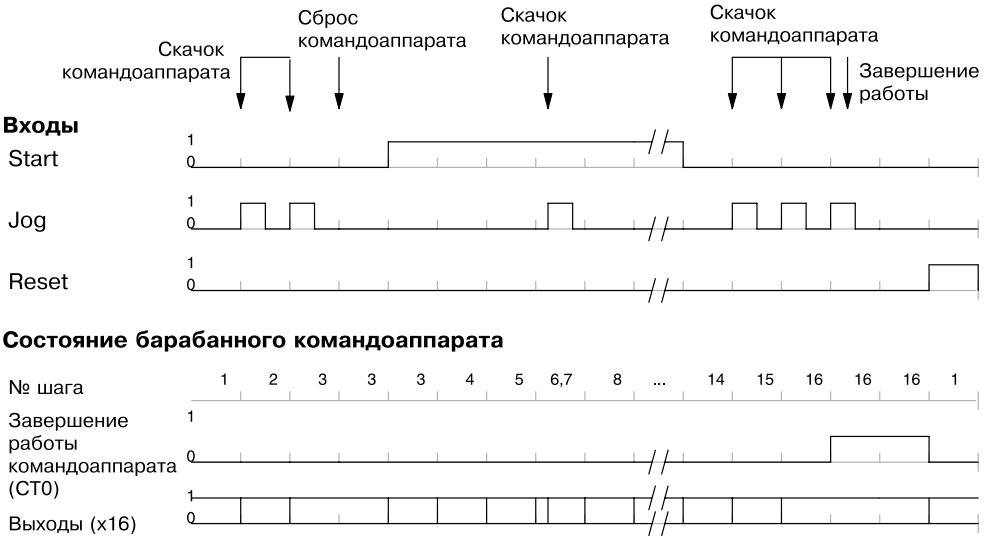
После того как барабан переходит к шагу 3, вход Start мгновенно отключается, останавливая таймер барабанного командоаппарата до нового включения Start.



Когда заканчивается последний шаг барабанного командоаппарата (в данном примере шаг 16), устанавливается бит окончания вращения (CT0), и номер шага остается равным 16. При включении входа Reset бит окончания вращения (CT0) сбрасывается, и командоаппарат принудительно переводится в предварительно установленное состояние.

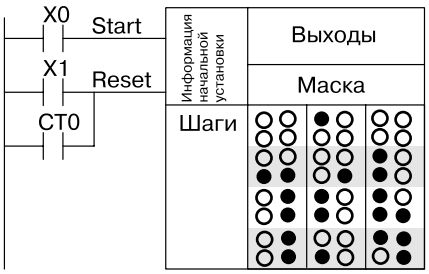
ПРИМЕЧАНИЕ. На временной диаграмме длительность всех шагов показана одинаковой. Длительность шагов может заметно меняться в зависимости от заданного числа единиц счета/шаг.

На приведенном ниже рисунке показано, как работает вход Jog с барабанными командоаппаратами с переходом по событиям. Обратите внимание на то, как в левой части диаграммы переходы ВЫКЛ/ВКЛ входа Jog увеличивают номер шага. Вход Start может быть включен или отключен (однако вход Reset должен быть выключен). Два скачка переводят командоаппарат на Шаг 3. Затем включается вход Start, и командоаппарат начинает работать как обычно. В течение шага 6 возникает другой входной сигнал Jog. Он переводит командоаппарат на шаг 7, сбрасывая таймер в 0. Командоаппарат начинает немедленно выполнять шаг 7, так как вход Start уже включен. К шагу 8 командоаппарат переходит обычным образом.
Когда барабанный командоаппарат переходит к шагу 14, вход Start отключается. Два дополнительных сигнала Jog переводят командоаппарат к шагу 16. Однако обратите внимание, что для перевода командоаппарата от шага 16 к состоянию «окончания цикла» требуется третий сигнал Jog. Наконец, поступает входной сигнал Reset, который возвращает командоаппарат в предварительно установленное состояние и сбрасывает бит завершения работы командоаппарата.



Барабанные командоаппараты с самовозвратом

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



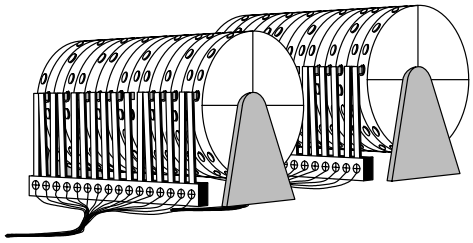
Инициализация выходов барабана

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

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

Каскадные барабанные командоаппараты, обеспечивающие более 16 шагов

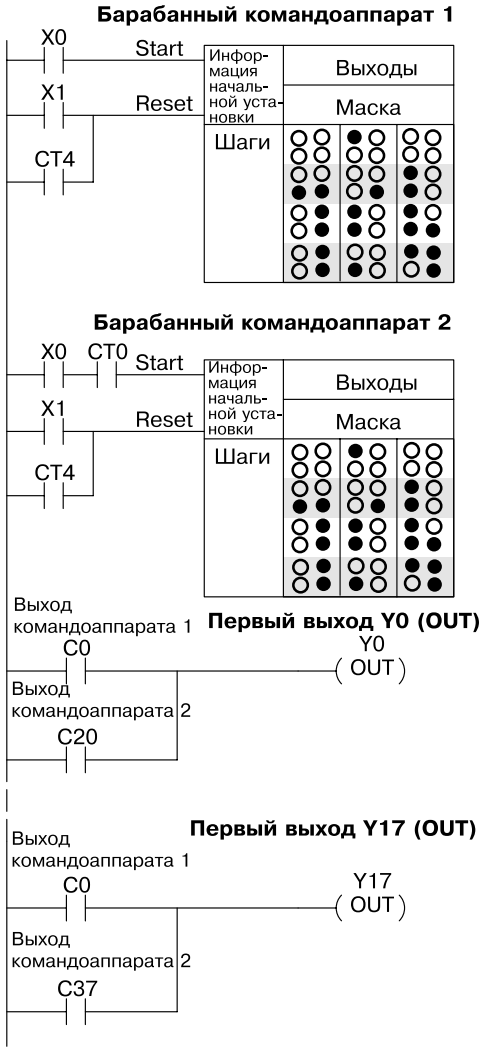
Иногда требуется барабанный командоаппарат, число шагов которого превышает 16. Решение состоит в том, чтобы использовать два или более командоаппарата, логически объединенных в каскад. Когда заканчивает работу первый командоаппарат, начинает работать второй, и т.д. Помните, что команда барабанного командоаппарата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



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

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

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

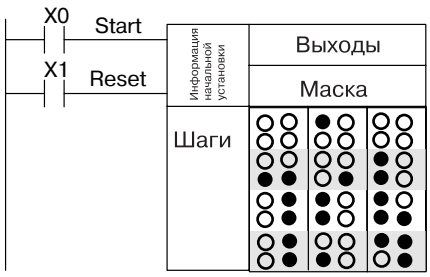


Мнемоника барабанного командоаппарата для ручного программатора

Для ввода или редактирования команд барабанного командоаппарата можно использовать пакет DirectSOFT или ручной программатор. В этой части описан ввод с ручного программатора.

Примечание: для редактирования команд барабанного командоаппарата нужен ручной программатор версии 5.5 и новее.

Сначала введите команды Store для цепей релейной логики управляюших входами барабанного командоаппарата. В примере справа, входами Start, Reset управляют X0 и X1. Ниже приведены нажатия клавиш для ввода этих команд.



Нажатия клавиш ручного программатора:

Store X0	STR	X(IN)	0	⟵
Store X1	STR	X(IN)	1	⟵

Этот Ввод предшествует вводу мнемоники DRUM. Заметьте, что цепи для входов Start, Reset не обязательно должны быть с одним контактом.

После ввода команд Store, введите команду барабанного командоаппарата. Например, введем команду основного временного командаппарата(мнемоника DRUM).Мы будем использовать счетчики- CT0, CT1,CT2 и CT3. Нажатия клавиш ручного программатора:

Нажатия клавиш ручного программатора:

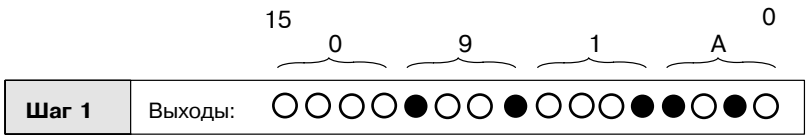
DRUM CNT0	SHFT	D	R	U	M	SHFT	CNT	0	⟵

После ввода мнемоники DRUM ручной программатор создает в памяти и на экране входную форму для всех параметров барабанного командоаппарата. Входная форма содержит 50 и более вводов по умолчанию с определенными назначениями (DEF). Мнемоника по умолчанию всегда готова к вводу. Используйте клавиши NXT и PREV для перемещения по форме. Требуется только редактирование значений по умолчанию. Вводы необходимые для основного барабанного командоаппарата приведены ниже.

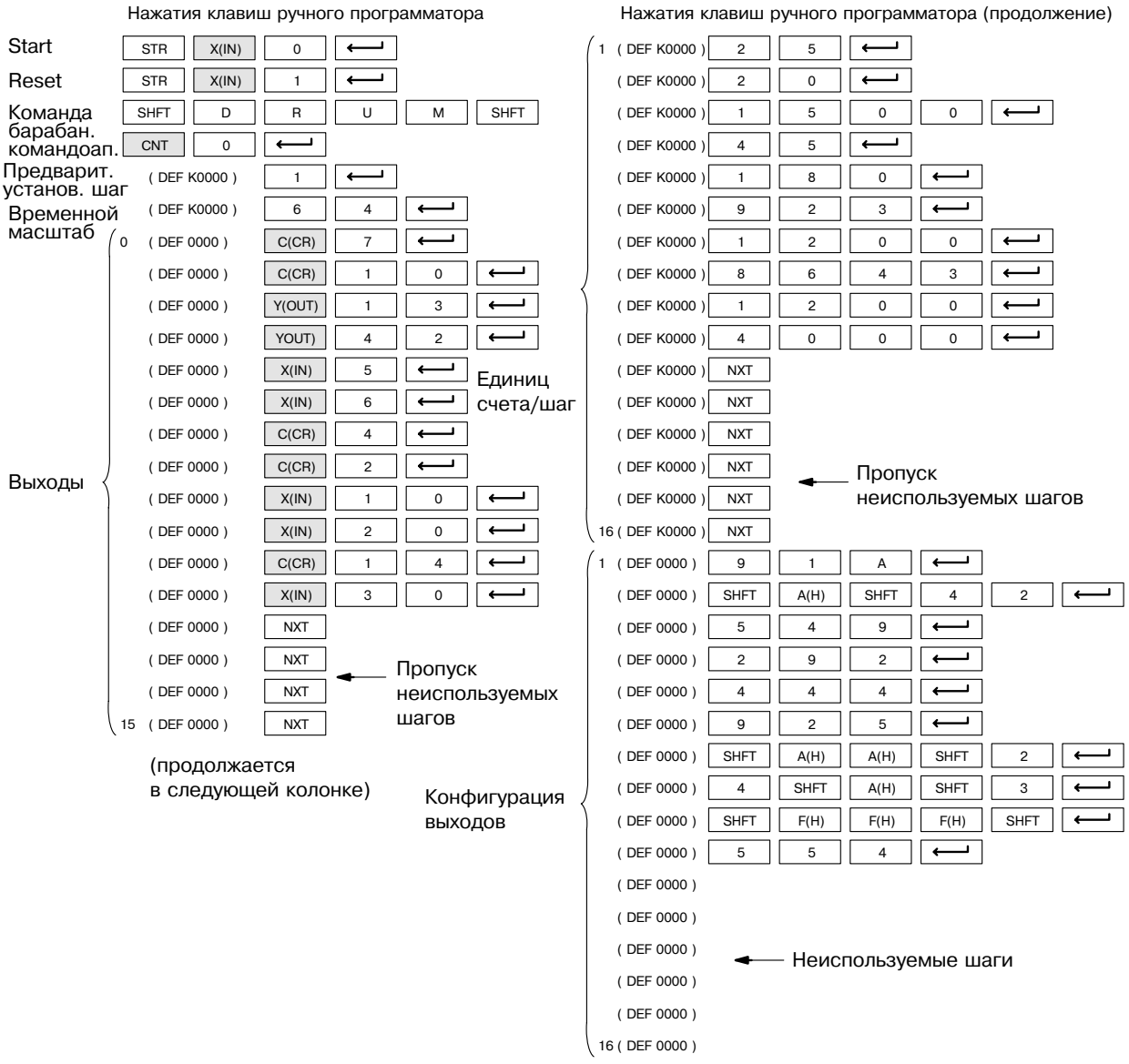
Параметры барабанного командоаппарата	Кол-во вводов	Кол-во вводов Мнемоника/Ввод	Мнемоника по умолчанию	Типы данных	Диапазон
Вход Start	-	STR (+вх.цепь)	-	-	-
Вход Reset	-	STR (+вх.цепь)		-	-
Мнемоника барабанного командоаппарата	-	DRUM CNT aaaa	-	K	0 - 777
Предварительно установленный шаг	1	bb	DEF K0000	K	1 - 16
Временной масштаб	1	cc	DEF K0000	K	0 - 9999
Дискретные выходы	16	Fffff	DEF 0000	X, Y, C *	
Единиц счета на шаг	16	dddd	DEF K0000	K	
Конфигурация выходов	16	gggg	DEF K0000	K	

Примечание: Ввод по умолчанию для дискретных выходов и событий - DEF 0000 означает, что они не связаны. Если Вы хотите изменить назначенный выход на неиспользуемый (не связанный), введите "K0000". Ввод покажет " DEF 0000"снова.

Используя входную форму команды DRUM показанную на предыдущей странице мы покажем способ ввода комады DRUM. Сначала мы преобразуем все конфигурации выходов на каждом шагу в шестнадцатиричный формат, как показано ниже.



На приведенной далее диаграмме показаны нажатия клавиш для ввода примера с командой DRUM. Вводы в форме по умолчанию заключены в скобках. После ввода команды барабанного командоаппарата остающиеся нажатия переписывают цифровую часть каждого определения - DEF.



Примечание. Для пропуска ввода последних неиспользуемых выходов или шагов можно использовать клавиши NXT и PREV.

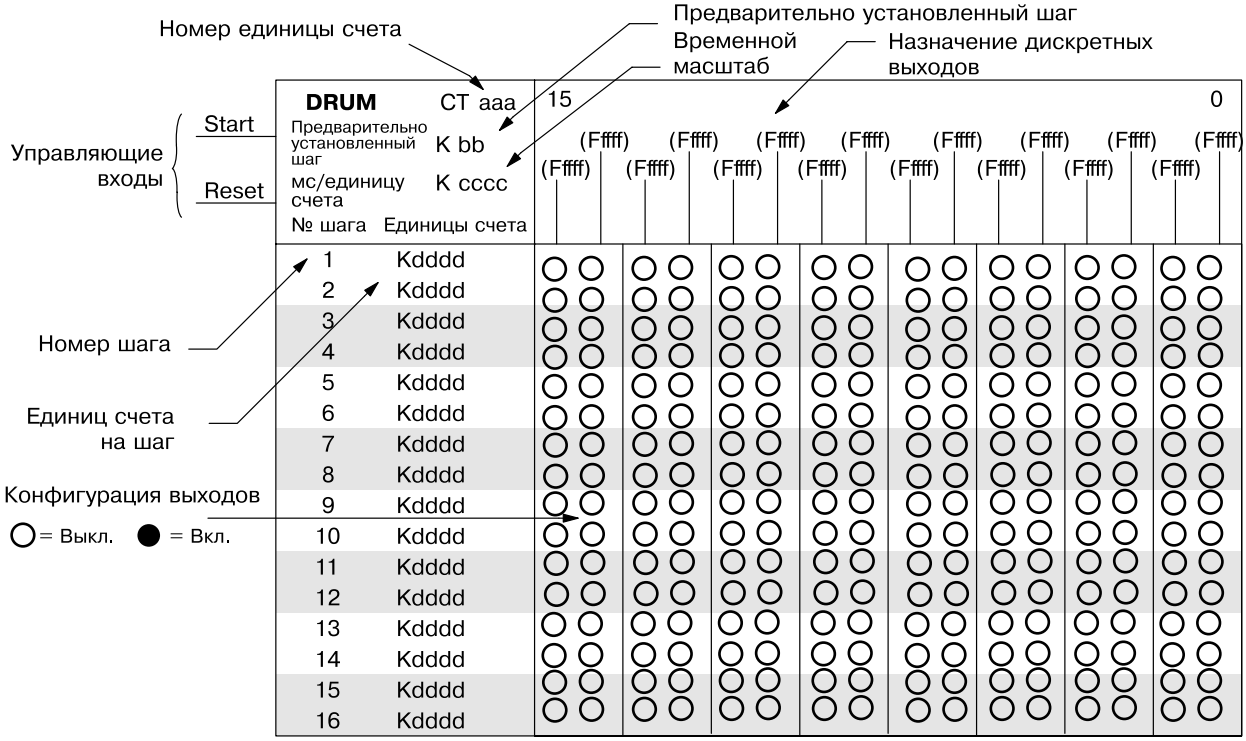
Команды барабанного командоаппарата

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

X	X	√
430	440	450

Команды барабанного командоаппарата DL450 могут быть запрограммированы с помощью **Direct**SOFT или, только для команды EDRUM, можно воспользоваться ручным программатором (с версией программного обеспечения 1.8 или более поздней). В данном разделе описано использование **Direct**SOFT для всех команд и мнемоника ручного программатора для команды EDRUM.

Барабанный командоаппарат с дискретными выходами и переходом по времени — это основная команда барабанного командоаппарата DL450. Он работает в соответствии с правилами, описанными на предыдущих страницах. Ниже команда показана в виде схемы, выводимой в **Direct**SOFT.



Барабанный командоаппарат с переходом по времени обеспечивает 16 шагов и 16 выходов. Переход с шага на шаг происходит только по времени, заданном числом единиц счета на шаг. Неиспользуемые шаги должны программироваться со значением «единиц счета на шаг» = 0 (значение по умолчанию). Дискретным точкам выхода может быть независимо назначен тип X, Y или C, или они могут быть оставлены неиспользуемыми. Конфигурацию выходов можно графически редактировать в **Direct**SOFT.

Как только будет инициализирован вход Start, станет доступным таймер барабанного командоаппарата. Он останавливается при завершении последнего шага или при инициализации входа Reset. Командоапапарат переходит к выбранному предварительно установленному шагу при переходе процессора в рабочий режим программы, а также при каждой инициализации входа Reset.

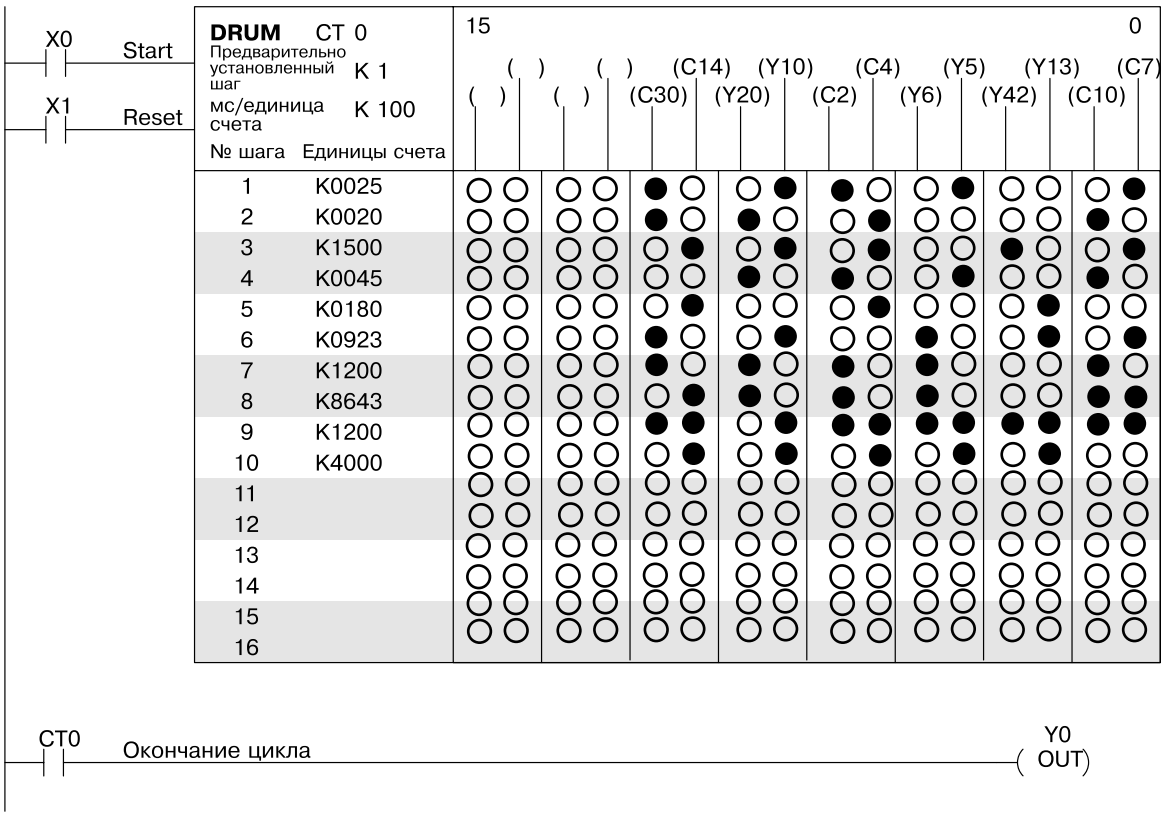
Параметры барабанного командоаппарата	Поле	Типы данных	Диапазон
Номер счетчика	aaa	—	0 - 177
Предварительно установленный шаг	bb	K	1 - 16
Временной масштаб	cccc	K	0 - 99.99 секунд
Единиц счета на шаг	dddd	K	0 - 9999
Дискретные выходы	Fffff	X, Y, C, GX, GY	см. стр 3-49

Команды барабанного командоаппарата используют четыре счетчика процессора. Программа может читать значения счетчиков, получая статус командоаппарата, а также в любой момент может записать в CT(n+2) новый номер предварительно установленного шага. Однако другие счетчики предназначены только для слежения.

Номер счетчика	Диапазон (n)	Функция	Функция бита счетчика
CT(n)	0 - 124	Единиц счета в шаге	CTn = Окончание цикла
CT(n+1)	1 - 125	Счетчик таймера	CT(n+ 1) = (не используется)
CT(n+2)	2 - 126	Предварительно установленный шаг	CT(n+ 2) = (не используется)
CT(n+3)	3 - 127	Текущий шаг	CT(n+ 1) = (не используется)

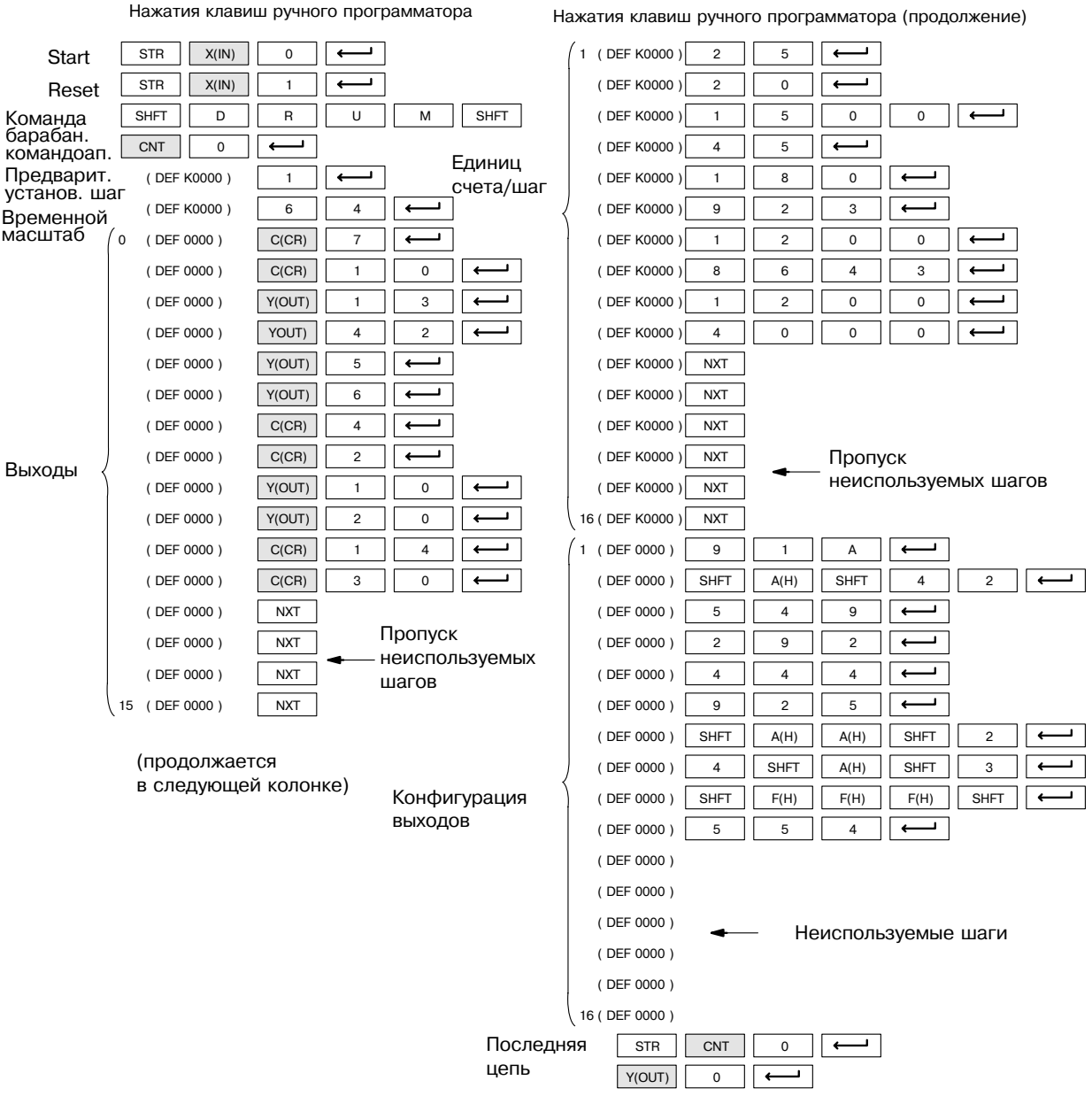
Следующая программа демонстрирует типичное применение команды DRUM в DirectSOFT. Используются шаги с 1 по 10 и двенадцать из шестнадцати выходных точек. Предварительно установленным шагом является шаг 1. Временной масштаб составляет 100 мс на единицу счета. Следовательно, длительность шага 1 равна (25 x 0.1) = 2.5 секунды. В последней цепи бит окончания вращения (CT0) включает выход Y0 по завершении последнего шага (шаг 10). Сброс барабанного командоаппарата также сбрасывает CT0.

Direct SOFT Display



Для ввода или редактирования команд барабанного командоаппарата можно также использовать ручной программатор. На приведенной ниже диаграмме перечислены нажатия клавиш для ввода примера команды, описанного на предыдущей странице.

Примечание. Для редактирования команд барабанного командоаппарата нужен ручной программатор с ПЗУ версии 5.5 или более поздней.



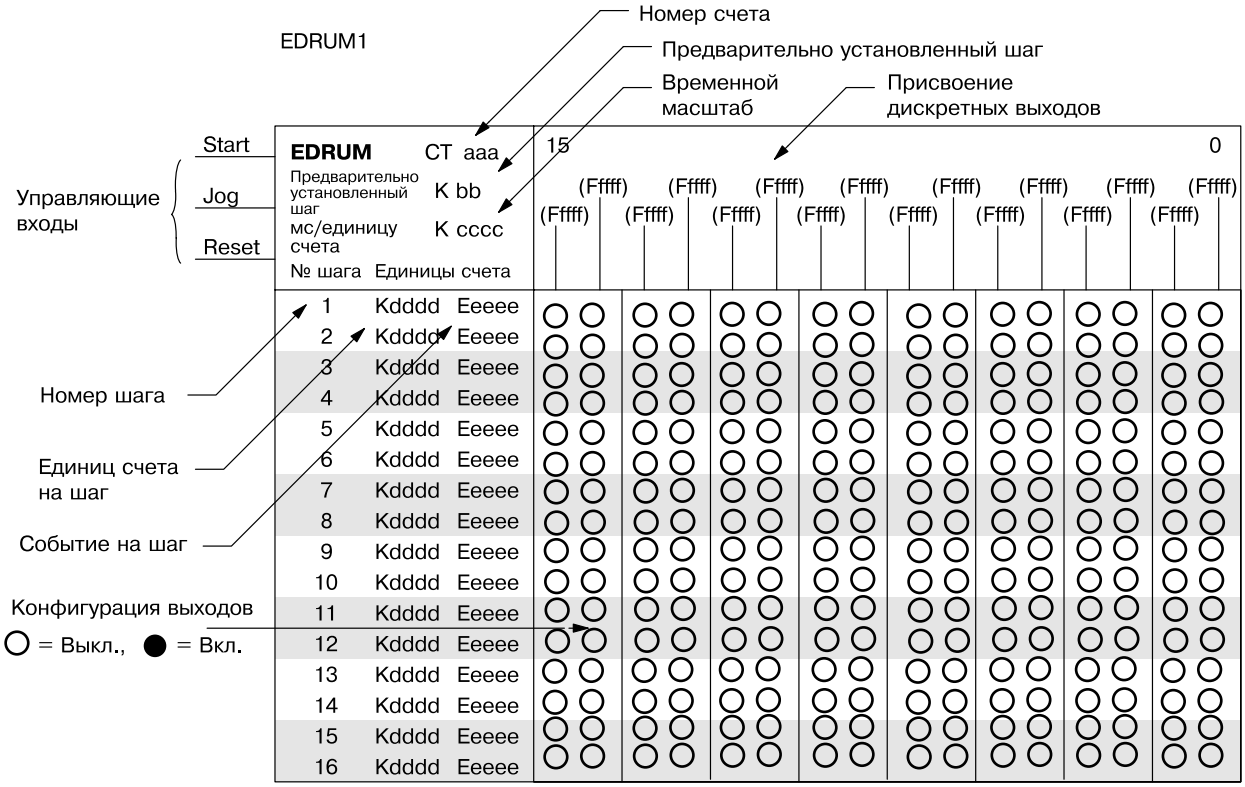
Примечание. Чтобы пропустить ввод последних неиспользуемых выходов или шагов, можно использовать клавиши NXT и PREV.

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

X	X	√
430	440	450

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

Барабанный командоаппарат с дискретными выходами и переходом по событию обеспечивает 16 шагов и 16 выходов. Переход шага происходит по времени и/или по событию. Вход Jog также вызывает переход шага при каждом включении. Время задается в единицах счета на шаг, а события задаются как дискретные контакты. Неиспользуемые шаги должны программироваться со значением «единиц счета на шаг» = 0 и событием = «0000». Тип дискретных точек выхода может быть назначен независимо. Конфигурацию выходов можно графически редактировать в **Direct**SOFT.



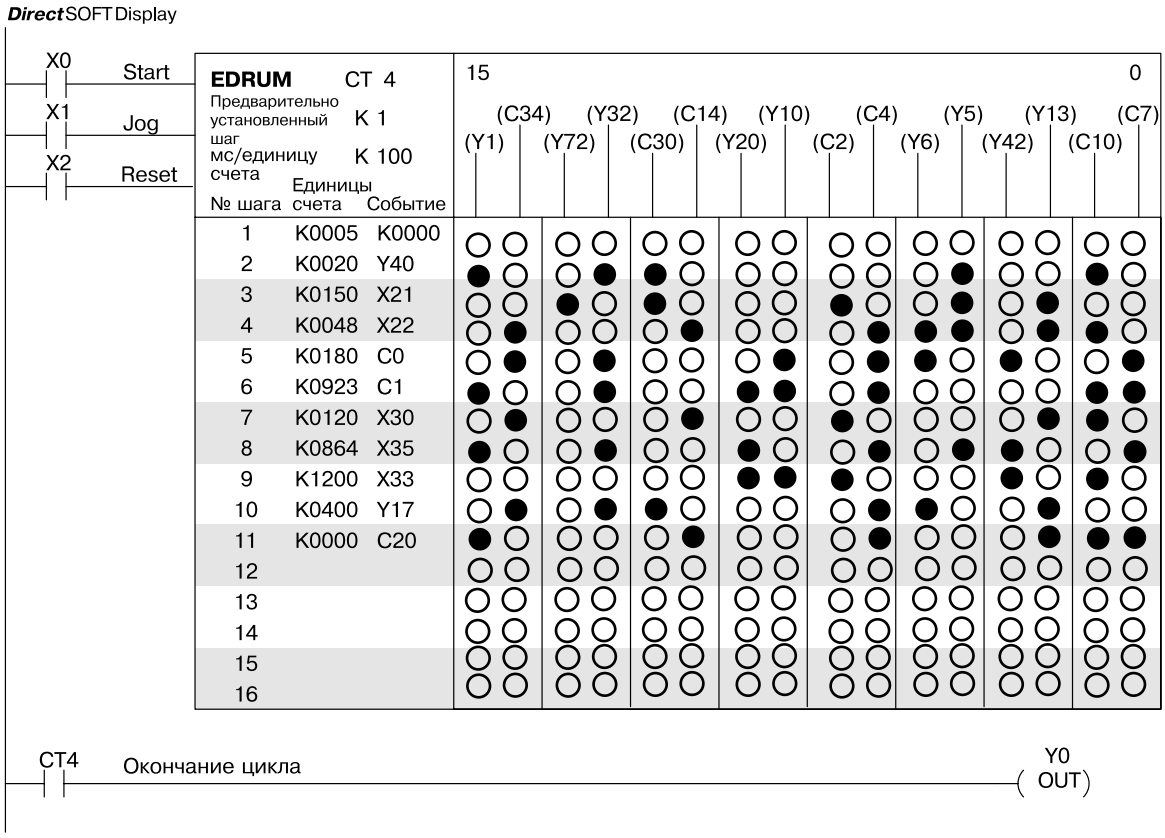
Как только будет инициализирован вход Start, станет доступным таймер барабанного командоаппарата. Таймер работает, пока имеет место событие для данного шага. Когда номер единицы шага становится равным значению единиц счета на шаг, командоаппарат переходит к следующему шагу. Этот процесс останавливается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

Параметры барабанного командоаппарата	Поле	Типы данных	Диапазон
Номер счетчика	aaa	—	0 - 177
Предварительно установленный шаг	bb	K	1 - 16
Временной масштаб	cccc	K	0 - 99.99 секунд
Единиц счета на шаг	dddd	K	0 - 9999
Событие	eeee	X, Y, C, GX, GY, S, T, ST	
Дискретные выходы	Fffff	X, Y, C , GX, GY	

Команды барабанного командоаппарата используют четыре счетчика процессора. Программа может читать значения счетчиков, получая статус командоаппарата, а также в любой момент может записать в CT(n+2) новый номер предварительно установленного шага. Однако другие счетчики предназначены только для слежения.

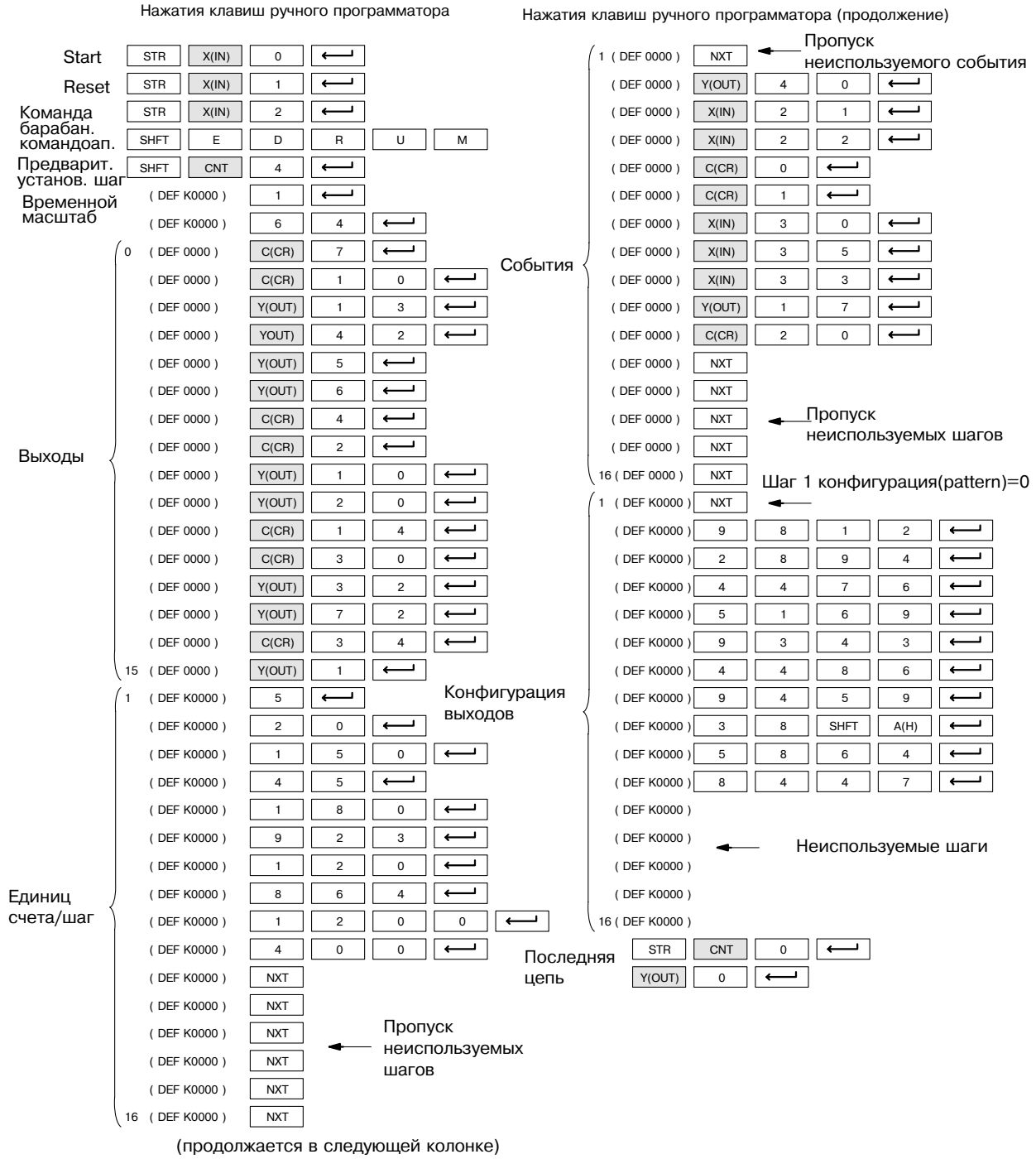
Номер счетчика	Диапазон (n)	Функция	Функция бита счетчика
CT(n)	0 - 124	Единиц счета в шаге	CTn = Окончание цикла
CT(n+1)	1 - 125	Счетчик таймера	CT(n+ 1) = (не используется)
CT(n+2)	2 - 126	Предварительно установленный шаг	CT(n+ 2) = (не используется)
CT(n+3)	3 - 127	Текущий шаг	CT(n+ 1) = (не используется)

Следующая программа демонстрирует типичное применение команды EDRUM в **Direct**SOFT. Используются шаги с 1 по 11 и все шестнадцать выходных точек. Предварительно установленным шагом является шаг 1. Временной масштаб составляет 100 мс на единицу счета. Следовательно, длительность шага 1 равна (5 x 0.1) = 0.5 секунды. Обратите внимание, что переход с шага 1 происходит только по времени (событие=«K0000»). Кроме того, в заданной конфигурации выходов для шага 1 все выходы отключены, что является типичным условием на включение питания. В последней цепи бит завершения работы командоаппарата (CT4) включает выход Y0 по завершении последнего шага (шаг 11). Сброс барабанного командоаппарата также сбрасывает CT4.



Для ввода или редактирования команд барабанного командоаппарата можно также использовать ручной программатор. На приведенной ниже диаграмме перечислены нажатия клавиш для ввода примера команды, описанного на предыдущей странице.

Примечание. Для редактирования команд барабанного командоаппарата нужен ручной программатор с ПЗУ версии 5.5 или более поздней.



(продолжается в следующей колонке)

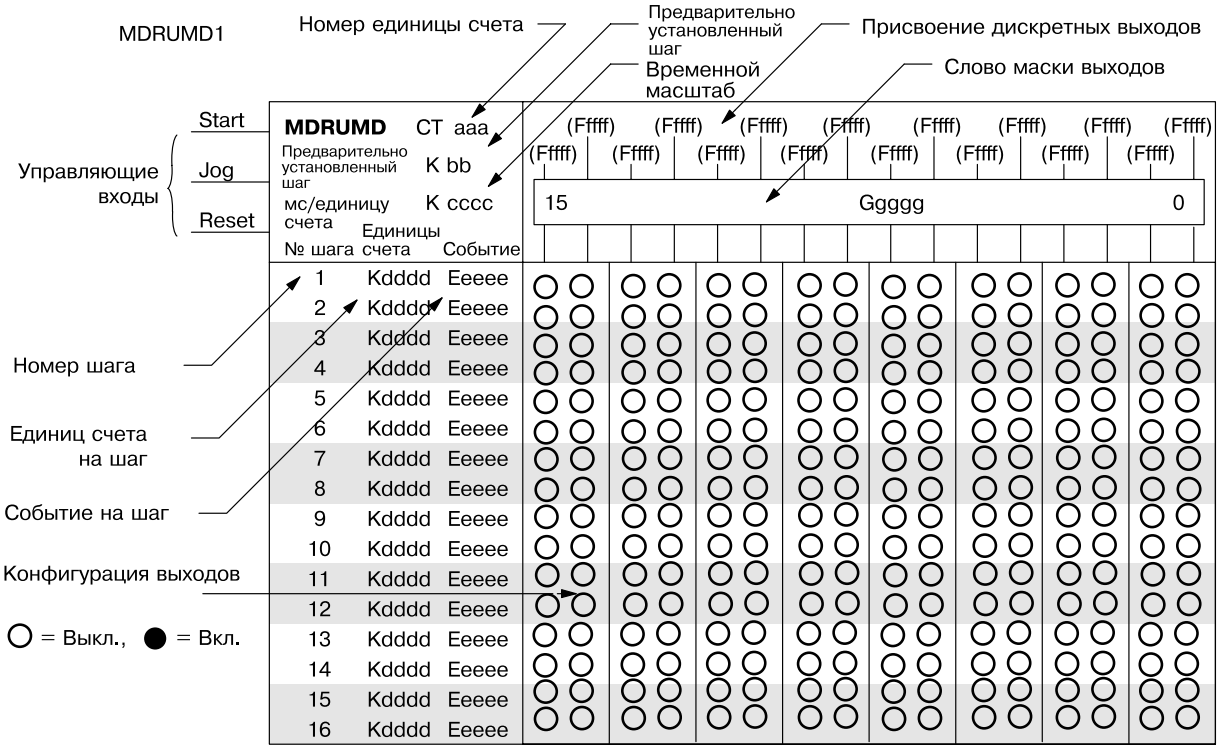
Примечание. Чтобы пропустить ввод последних неиспользуемых выходов или шагов, можно использовать клавиши NXT и PREV.

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

X	X	√
430	440	450

Маскированный барабанный командоаппарат с дискретными выходами и переходом по событию обладает всеми свойствами командоаппарата с переходом по событию, плюс заключительное управление выходами для каждого шага. Он работает в соответствии с общими правилами работы барабанных командоаппаратов, описанными в начале этого раздела. Ниже данная команда показана в виде схемы, выводимой в **Direct**SOFT.

Маскированный барабанный командоаппарат с дискретными выходами и переходом по событию обеспечивает 16 шагов и 16 выходов. На каждом шаге над выходами выполняется операция побитового И со словом маски выходов. Поле Ggggg задает начальную ячейку 16 слов маски. Переход шага происходит по времени и/или по событию. Вход Jog также вызывает переход шага при каждом включении. Время задается в единицах счета на шаг, а события задаются как дискретные контакты. Неиспользуемые шаги должны программироваться со значением «единиц счета на шаг» = 0 и событием = «0000».



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

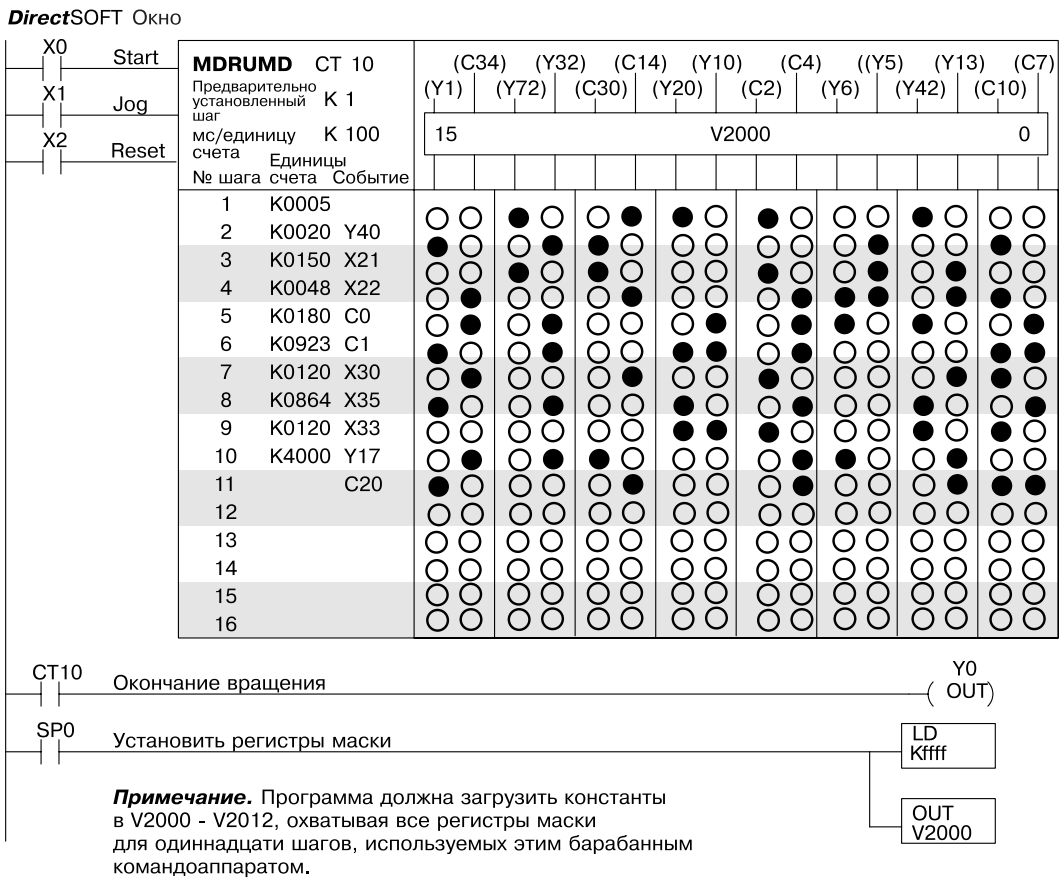
Параметры барабанного командоаппарата	Поле	Типы данных	Диапазон
Номер счетчика	aaa	—	0 - 177
Предварительно установленный шаг	bb	K	1 - 16
Временной масштаб	cccc	K	0 - 99.99 секунд
Единиц счета на шаг	dddd	K	0 - 9999
Событие	eeee	X, Y, C, GX, GY, S, T, ST	
Дискретные выходы	Fffff	X, Y, C, GX, GY	
Маска выходов	Ggggg	V	

Команды барабанного командоаппарата используют четыре счетчика процессора. Программа может читать значения счетчиков, получая статус командоаппарата, а также в любой момент может записать в CT(n+2) новый номер предварительно установленного шага. Однако другие счетчики предназначены только для слежения.

Номер счетчика	Диапазон (n)	Функция	Функция бита счетчика
CT(n)	0 - 124	Единиц счета в шаге	CTn = Окончание цикла
CT(n+1)	1 - 125	Счетчик таймера	CT(n+ 1) = (не используется)
CT(n+2)	2 - 126	Предварительно установленный шаг	CT(n+ 2) = (не используется)
CT(n+3)	3 - 127	Текущий шаг	CT(n+ 1) = (не используется)

Следующая программа демонстрирует типичное применение команды MDRUMD в **Direct**SOFT. Используются шаги с 1 по 11 и все шестнадцать выходных точек. Словом маски выходов является V2000. Окончательные выходы барабанного командоаппарата показаны над словом маски в виде отдельных битов. Над битами данных V2000 и конфигурацией выходов текущего шага командоаппарата выполняется операция логического И. Если нужно, чтобы при включении питания все выходы были отключены, введите для первого сканирования нули в V2000. Программа может в любой момент обновить маску выходов, включая или отключая выходы барабана. Предварительно установленным шагом является шаг 1. Временной масштаб составляет 100 мс на единицу счета. Следовательно, длительность шага 1 равна (5 x 0.1) = 0.5 секунды. Обратите внимание, что переход с шага 1 происходит только по времени (событие=«K0000»). Кроме того, в заданной конфигурации выходов для шага 1 все выходы отключены, что является типичным условием на включение питания. В последней цепи бит окончания вращения (CT10) включает выход Y0 по завершении последнего шага (шаг 11). При сбросе барабанного командоаппарата также сбрасывается CT10.

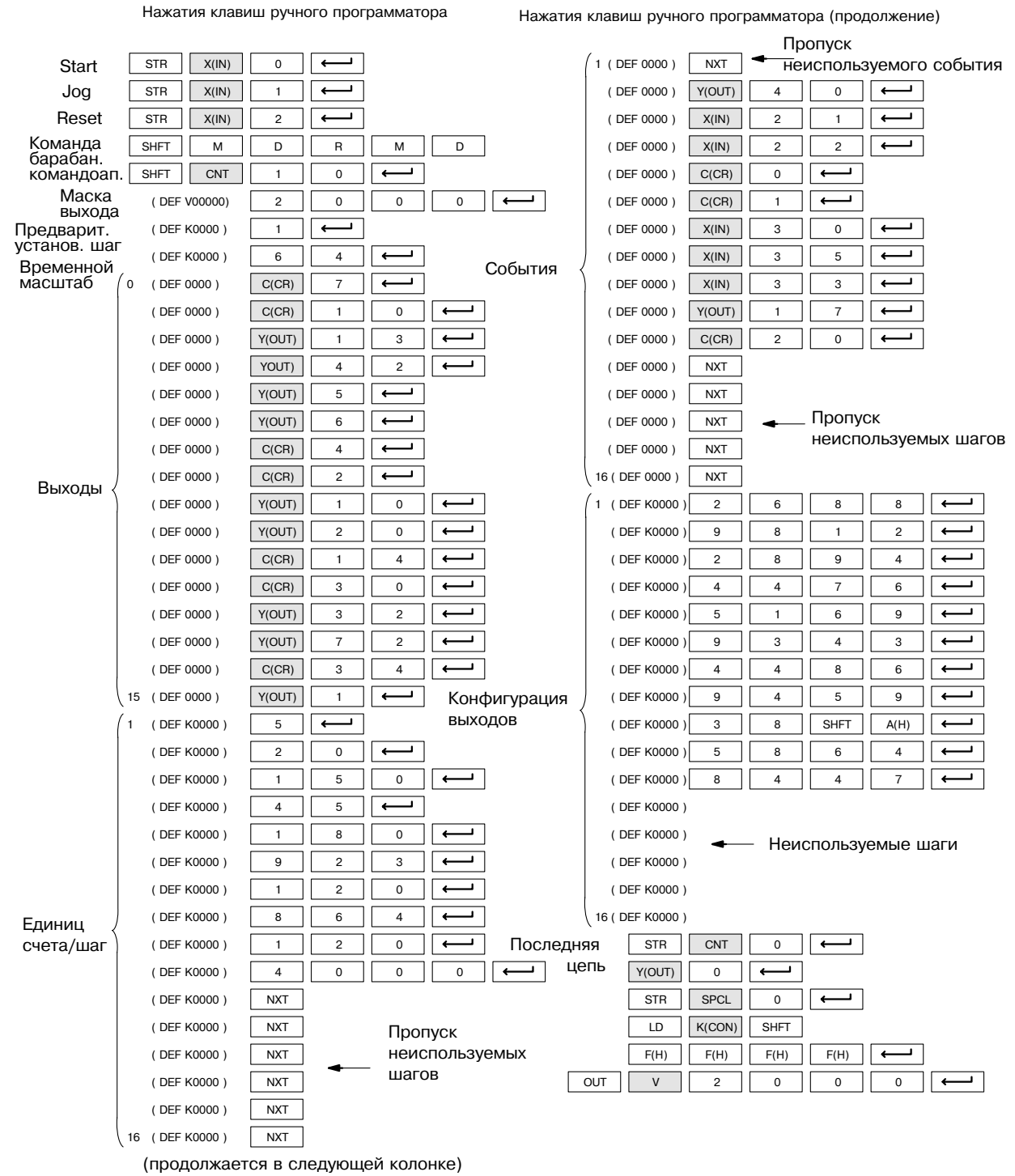
DirectSOFT Окно



Примечание. Программа должна загрузить константы в V2000 - V2012, охватывая все регистры маски для одиннадцати шагов, используемых этим барабанным командоаппаратом.

Для ввода или редактирования команд барабанного командоаппарата можно также использовать ручной программатор. На приведенной ниже диаграмме перечислены нажатия клавиш для ввода примера команды, описанного на предыдущей странице.

Примечание. Для редактирования команд барабанного командоаппарата нужен ручной программатор с ПЗУ версии 5.5 или более поздней.



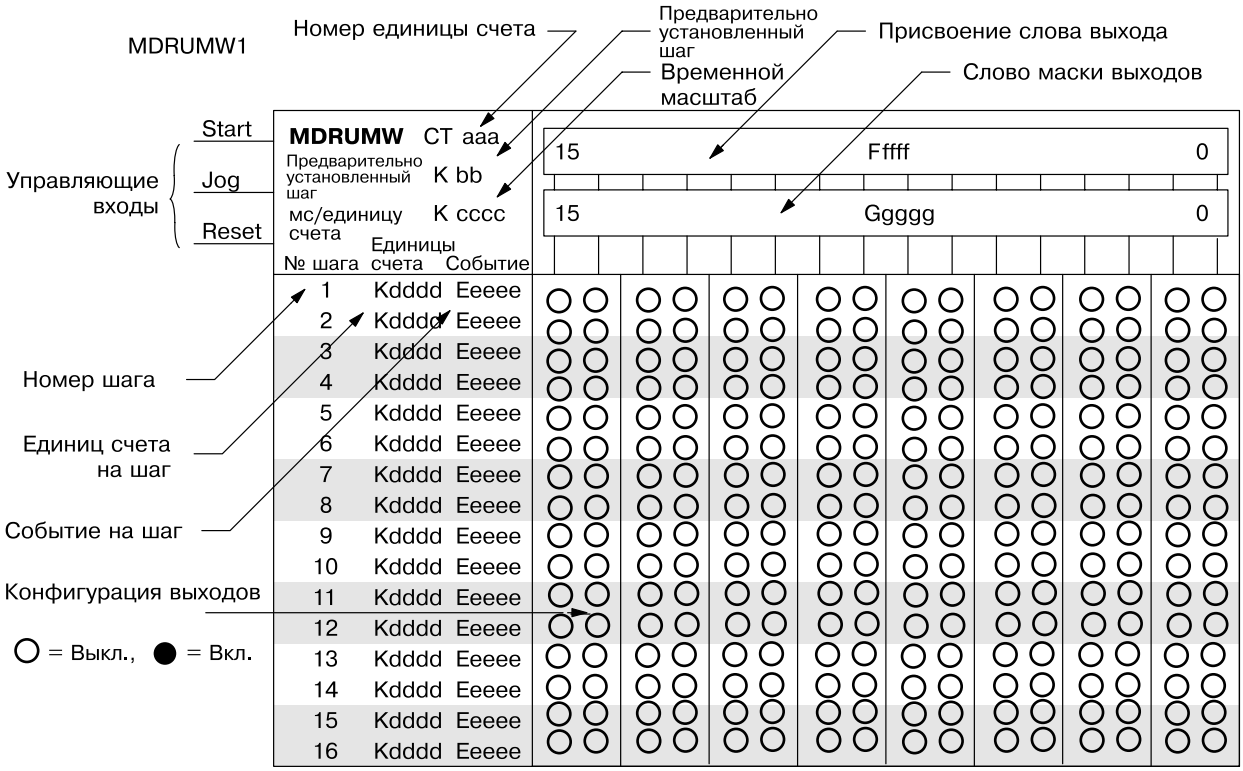
Примечание. Чтобы пропустить ввод последних неиспользуемых выходов или шагов, можно использовать клавиши NXT и PREV.

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

X	X	√
430	440	450

Выходы маскированного барабанного командоаппарата с выходом по словам и переходом по событию представляют собой биты одного слова, а не дискретные точки. Командоаппарат работает в соответствии с общими правилами работы барабанных командоаппаратов, описанными в начале этого раздела. Ниже команда показана в виде схемы, выводимой в **Direct**SOFT.

Маскированный барабанный командоаппарат с выходом по словам и переходом по событию обеспечивает 16 шагов и 16 выходов. На каждом шаге над выходами выполняется операция побитового И со словом маски выходов. Поле Ggggg задает начальную ячейку 16 слов маски, создавая окончательный выход (поле Fffff). Переход шага происходит по времени и/или по событию. Вход Jog также вызывает переход шага при каждом включении. Время задается в единицах счета на шаг, а события задаются как дискретные контакты. Неиспользуемые шаги должны программироваться со значением «единиц счета на шаг» = 0 и событием = «0000».



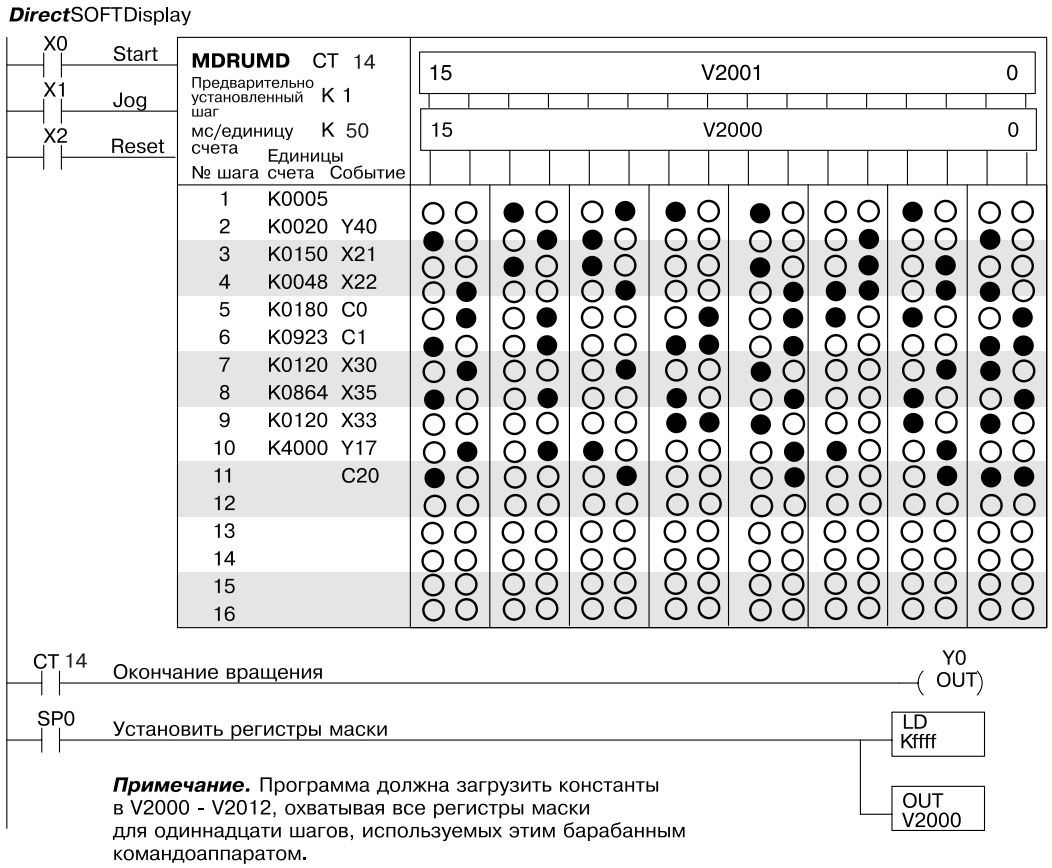
Как только будет инициализирован вход Start, станет доступным таймер барабанного командоаппарата. Таймер работает, пока имеет место событие для данного шага. Когда номер единицы счета шага становится равным значению единиц счета на шаг, командоаппарат переходит к следующему шагу. Этот процесс останавливается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

Параметры барабанного командоаппарата	Поле	Типы данных	Диапазон
Номер счетчика	aaa	—	0 - 177
Предварительно установленный шаг	bb	K	1 - 16
Временной масштаб	cccc	K	0 - 99.99 секунд
Единиц счета на шаг	dddd	K	0 - 9999
Событие	eeee	X, Y, C, GX, GY, S, T, ST	см. стр. 3-49
Слово-выход	Fffff	V	см. стр. 3-49
Маска выходов	Ggggg	V	см. стр. 3-49

Команды барабанного командоаппарата используют четыре счетчика процессора. Программа может читать значения счетчиков, получая статус командоаппарата, а также в любой момент может записать в CT(n+2) новый номер предварительно установленного шага. Однако другие счетчики предназначены только для слежения.

Номер счетчика	Диапазон (n)	Функция	Функция бита счетчика
CT(n)	0 - 124	Единиц счета в шаге	CTn = Окончание цикла
CT(n+1)	1 - 125	Счетчик таймера	CT(n+ 1) = (не используется)
CT(n+2)	2 - 126	Предварительно установленный шаг	CT(n+ 2) = (не используется)
CT(n+3)	3 - 127	Текущий шаг	CT(n+ 1) = (не используется)

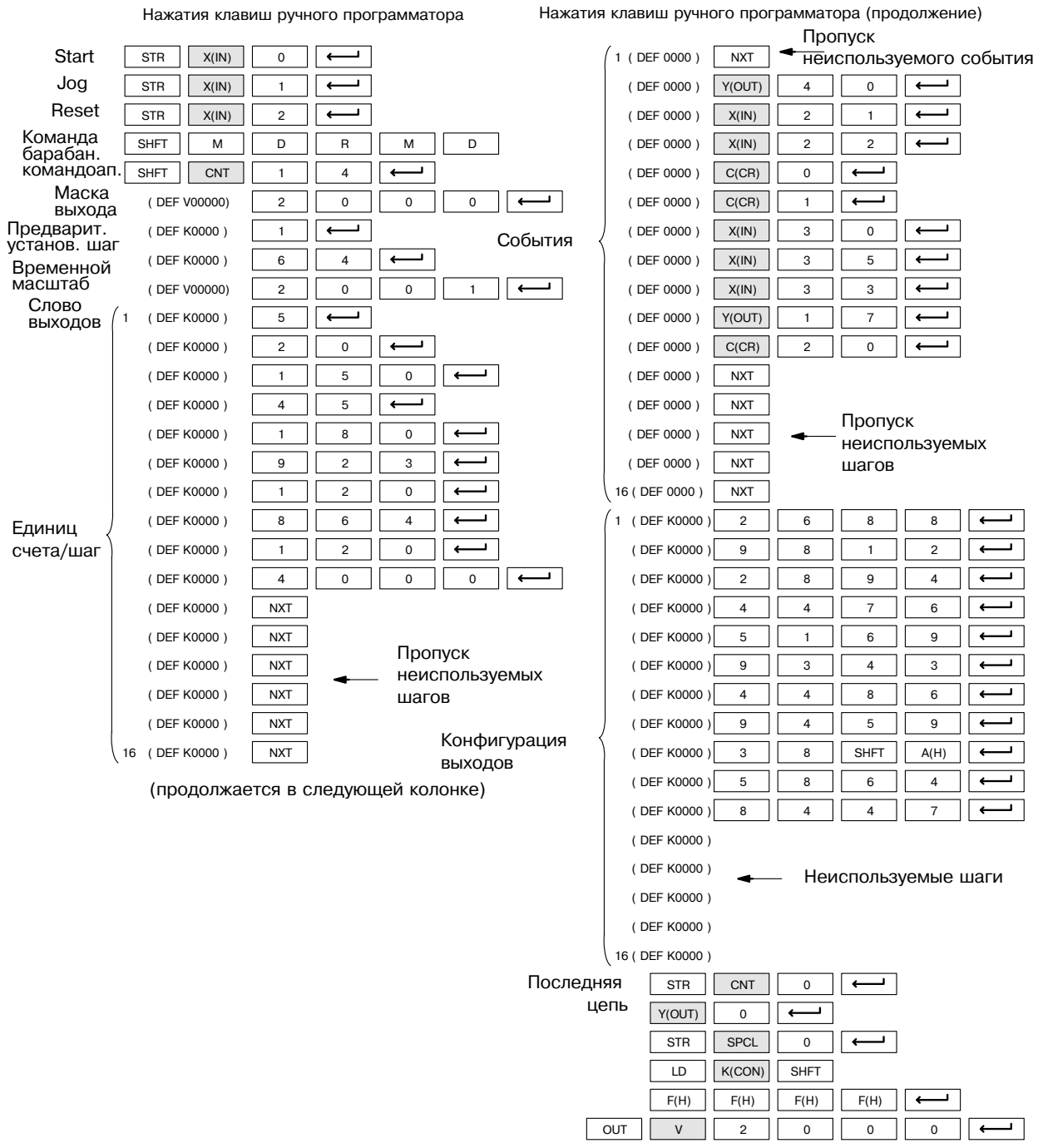
Следующая программа демонстрирует типичное применение команды MDRUMW в **Direct**SOFT. Используются шаги с 1 по 11 и все шестнадцать выходных точек. Словом маски выходов является V2000. Окончательные выходы барабанного командоаппарата показаны над словом маски как слово с адресом V2001. Над битами данных V2000 и конфигурацией выходов текущего шага командоаппарата выполняется операция логического И, создавая содержимое V2001. Если нужно, чтобы при включении питания все выходы были отключены, введите нули в V2000 для первого сканирования. Программа может в любой момент обновить маску выходов, включая или отключая выходы командоаппарата. Предварительно установленным шагом является шаг 1. Временной масштаб составляет 100 мс на единицу счета. Следовательно, длительность шага 1 равна (5 x 0.1) = 0.5 секунды. Обратите внимание, что переход с шага 1 происходит только по времени (событие=«K0000»). Кроме того, в заданной конфигурации выходов для шага 1 все выходы отключены, что является типичным условием на включение питания. В последней цепи бит окончания вращения (CT14) включает выход Y0 по завершении последнего шага (шаг 11). Сброс барабанного командоаппарата также сбрасывает CT14.



Примечание. Программа должна загрузить константы в V2000 - V2012, охватывая все регистры маски для одиннадцати шагов, используемых этим барабанным командоаппаратом.

Для ввода или редактирования команд барабанного командоаппарата можно также использовать ручной программатор. На приведенной ниже диаграмме перечислены нажатия клавиш для ввода примера команды, описанного на предыдущей странице.

Примечание. Для редактирования команд барабанного командоаппарата нужен ручной программатор с ПЗУ версии 5.5 или более поздней.



Примечание. Для пропуска ввода последних неиспользуемых выходов или шагов можно использовать клавиши NXT и PREV.

Стадийное программирование RLL PLUS

7

В этой главе……

— Введение в стадийное программирование

— Как составлять диаграммы перехода состояния

— Использование команды Stage Jump для переходов состояния

— Пример стадийной программы: контроллер включения/выключения лампочки

— Четыре этапа написания стадийной программы

— Пример стадийной программы: управление дверью гаража

— Анализ проекта стадийной программы

— Понятия параллельной обработки

— Управление большими программами

— Команды RLL PLUS

— Вопросы и ответы по стадийному программированию

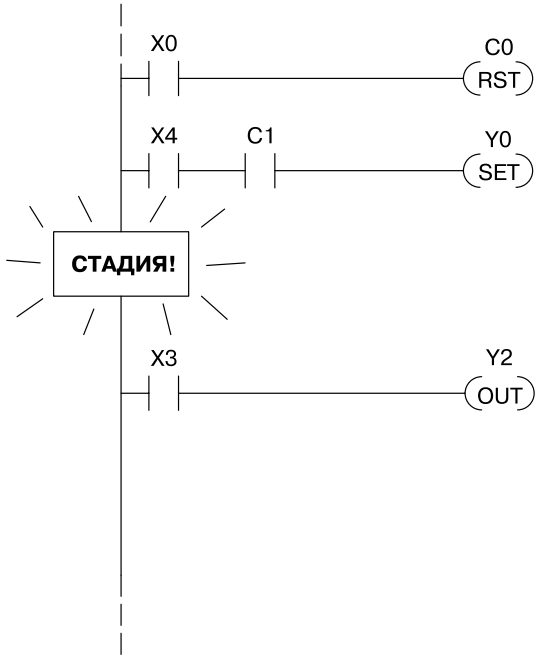
Введение в стадийное программирование

√	√	√
430	440	450

Стадийное программирование (доступно на всех процессорах DL450) обеспечивает относительно простой (по сравнению с решениями традиционной релейной логики, RLL) способ организации и программирования сложных приложений. Стадийное программирование не заменяет и не отвергает традиционное релейное программирование. Именно поэтому стадийное программирование называют также RLL[plus]. Вам не придется отказываться от уже имеющихся знаний и опыта. Стадийное программирование просто позволяет организовать RLL-программу в виде групп команд, называемых стадиями, что по сравнению с традиционной RLL позволяет разрабатывать программу быстрее и понятнее.

Преодоление "Боязни стадий"

Многие программисты ПЛК почувствовали себя комфортнее, используя RLL для всех своих программ ПЛК. Однако они нередко относятся скептически, если не со страхом, к изучению новых методов, таких как стадийное программирование. Хотя RLL отлично работает при определении взаимосвязей логической схемы, у нее все же имеются недостатки:



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

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

Итак, мы призываем изучить стадийное программирование, чтобы добавить его в свой «инструментарий» методов программирования. Чтобы достичь наилучших результатов:

- Начните с самого начала и не пропускайте ни одного раздела.
- Изучайте каждое понятие стадийного программирования, прорабатывая каждый пример. Примеры выстроены последовательно один за другим.
- Прочтите вопросы и ответы в конце главы для быстрого повторения всей темы

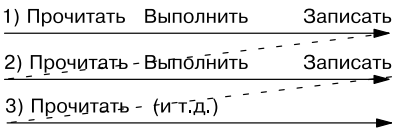
Как составлять диаграммы перехода состояния

Введение в состояния процесса

Те, кто знаком с выполнением релейной программы, знают, что процессор должен повторно, снова и снова, сканировать программу. Три основных шага сканирования:

1. Чтение входов
2. Выполнение программы
3. Запись выходов

Сканирование ПЛК



Преимущество такого подхода в том, что изменение входов может повлиять на выходы за считанные миллисекунды.

Большинство производственных процессов состоит из последовательности действий или их режимов, которые могут длиться несколько секунд, минут или даже часов. Мы можем назвать такие периоды «состояниями процесса», которые в каждый конкретный момент времени либо активны, либо неактивны. Проблема RLL-программ состоит в том, что временная протяженность конкретного входного события может быть очень короткой. Обычно, чтобы удержать входное событие с целью сохранения состояния процесса в течение требуемого периода времени, мы создаем в RLL блокирующие реле.

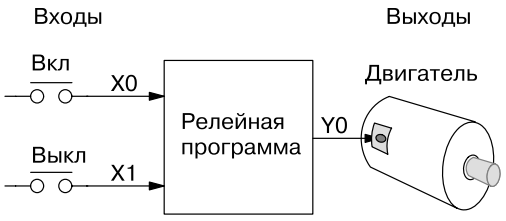
Мы можем разбить релейную программу на разделы, которые называются «стадиями» и представляют состояния процесса. Но прежде чем подробнее описать стадии, откроем **секрет понимания стадийного программирования**: диаграммы перехода состояния.

Необходимость диаграмм состояния

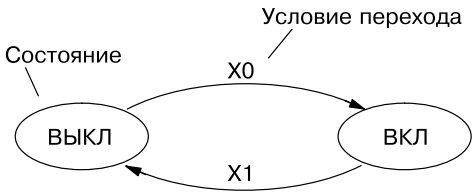
Иногда необходимо забыть о сканирующей природе ПЛК и сфокусироваться на состояниях процесса, которые нужно выделить. Наилучшими предпосылками к написанию эффективных, свободных от ошибок программ являются ясное понимание приложения и его краткий анализ. Диаграммы состояния призваны помочь изобразить общую картину нашего процесса! Вы убедитесь в том, что если мы сможем сделать это правильно, то и **наша программа также будет безошибочной**!

Процесс с 2-мя состояниями

Рассмотрим простой процесс, показанный справа, который управляет промышленным двигателем. Мы будем использовать зеленую нефиксируемую кнопку SPST для включения двигателя, а красную — для выключения. Два состояния нашего процесса — это ВКЛ и ВЫКЛ.



На следующем этапе требуется нарисовать диаграмму перехода состояния, как показано справа. Она изображает два состояния ВЫКЛ и ВКЛ с линиями двух переходов между ними. Когда событие X0 истинно, мы переходим от состояния ВЫКЛ к состоянию ВКЛ. Когда истинно X1, мы переходим от ВКЛ к ВЫКЛ.

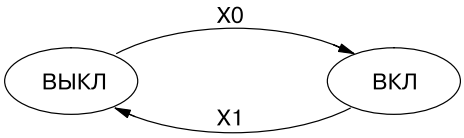


Уравнение выхода: Y0=ВКЛ

Двигаясь дальше, вы вплотную приблизитесь к пониманию концепции диаграмм перехода состояния и их эффективности при решении поставленных задач. Выход нашего контроллера равен значению Y0, которое будет истинным всякий раз, когда мы находимся в состоянии ВКЛ. Используя булевы выражения, можно записать Y0=состояние ВКЛ.

Далее мы реализуем диаграмму состояния сначала как RLL, а затем как стадийную программу. Это поможет понять взаимосвязь между двумя методами решения поставленной задачи.

Приведенная справа диаграмма перехода состояния отображает решение, которое необходимо реализовать. Достоинство ее в том, что она выражает проблему независимо от языка программирования, используемого для ее реализации. Иными словами, **нарисовав диаграмму, мы уже решили задачу управления**!

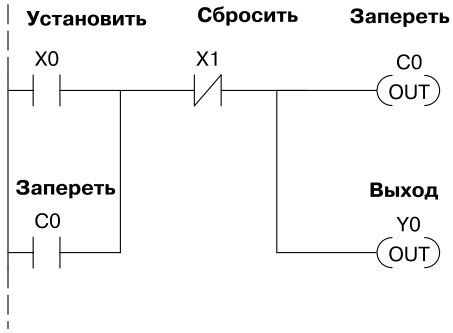


Уравнение выхода: Y0=ВКЛ

Вначале мы переведем диаграмму состояния в традиционную RLL. Затем покажем, насколько просто преобразовать диаграмму в решение стадийного программирования.

Эквивалент RLL

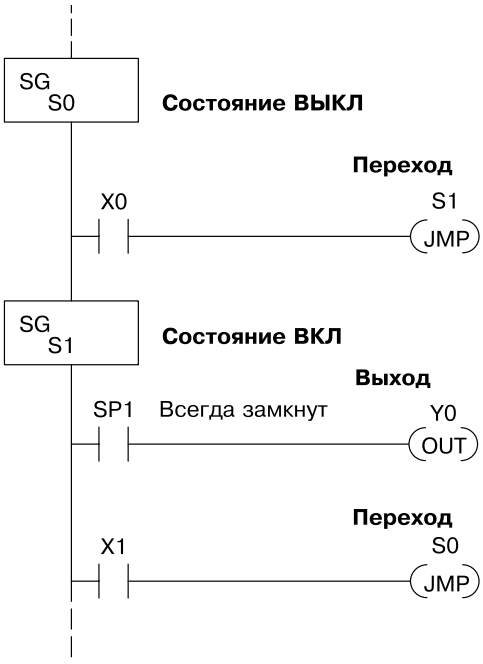
Решение RLL показано справа. Оно состоит из управляющего реле с самоблокировкой C0. При нажатии нефиксируемой кнопки X0 включается выходная обмотка C0, и запирается во включенном состоянии контакт C0 во второй цепи. Таким образом, X0 запирает C0, оставляя его во включенном состоянии и после размыкания контакта X0. В результате питание также подается на выход двигателя Y0, так что двигатель теперь включен.



Когда нажата выключающая кнопка (X1), размыкается обычно замкнутый контакт X1, освобождая запертый контакт. Выход двигателя Y0 выключается, когда запирающая обмотка C0 переходит в выключенное состояние.

Стадийный эквивалент

Решение стадийной программы показано на рисунке справа. Два линейно расположенных прямоугольника стадий S0 и S1 соответствуют двум состояниям ВЫКЛ и ВКЛ. Цепи программы под каждым прямоугольником связаны с соответствующей стадией. Это означает, что ПЛК должен просматривать только цепи, входящие в стадии, которые находятся в активном состоянии!



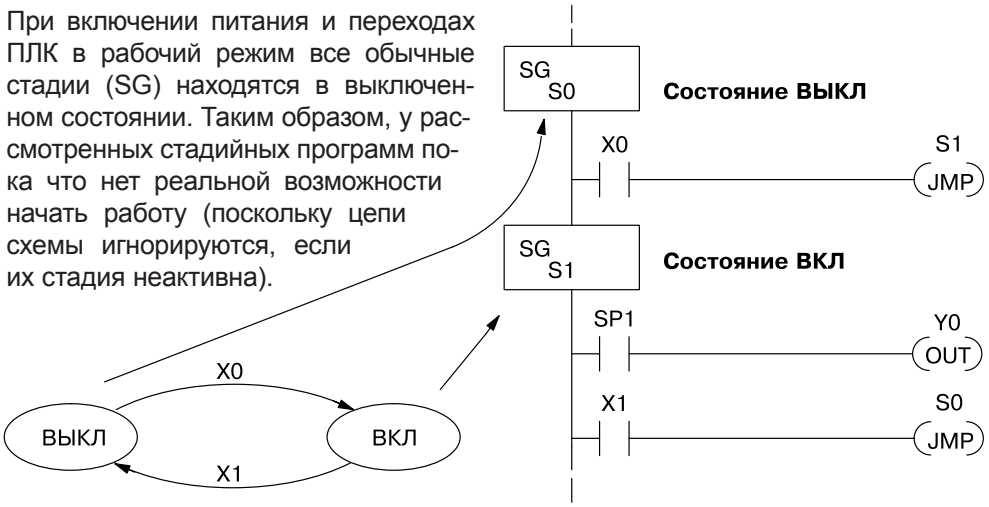
Допустим, что мы начинаем в состоянии ВЫКЛ, так что активна стадия S0. При нажатии клавиши ВКЛ (X0) происходит переход между стадиями. Выполняется команда JMP S1, которая просто сбрасывает бит стадии S0 и устанавливает бит стадии S1. Так что при следующем проходе процессор ПЛК проигнорирует стадию S0 и выполняет стадию S1! В состоянии ВКЛ (стадия S1) мы хотим чтобы двигатель всегда был включен. Определено, что контакт специального реле SP1 постоянно замкнут, так что Y0 включает двигатель.

Когда нажимается клавиша ВЫКЛ (X1), происходит обратный переход к состоянию ВЫКЛ. Выполняется команда JMP S0, которая сбрасывает бит стадии S1 и устанавливает бит стадии S0. При следующем сканировании процессор ПЛК не будет выполнять стадию S1, в результате выход двигателя Y0 будет отключен. Состояние ВЫКЛ (стадия 0) будет готово к следующему циклу.

Давайте сравним

Сейчас вы, возможно, не видите особых преимуществ в стадийном программировании, и стадийная программа кажется длиннее простой RLL. Ну что ж, самое время довериться нам. С ростом сложности задач управления стадийное программирование оставит RLL позади по простоте, размеру программы и т.д. Например, рассмотрим диаграмму, изображенную ниже. Обратите внимание, насколько прозрачно состояния ВЫКЛ и ВКЛ на диаграмме перехода состояния соответствуют стадийной программе, изображенной справа. Ну а теперь попробуйте-ка с той же легкостью идентифицировать аналогичные состояния в RLL-программе, изображенной на предыдущей странице!

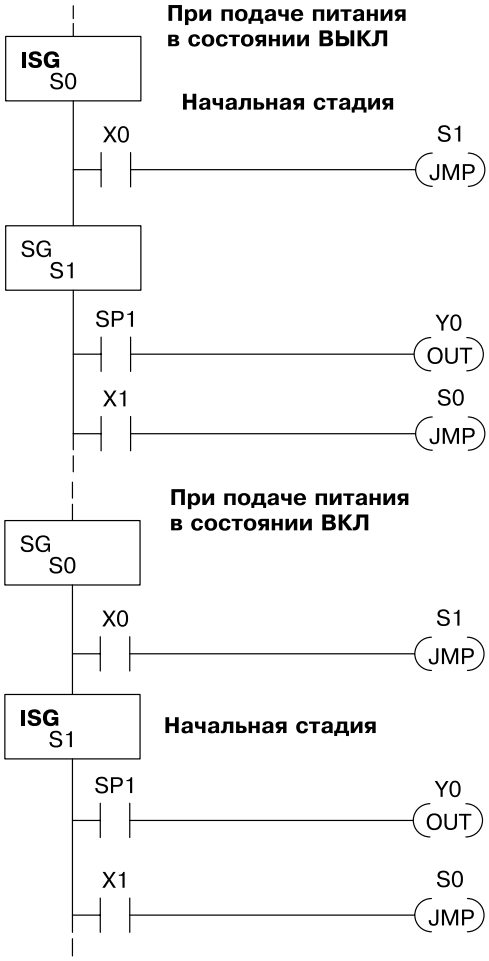
При включении питания и переходах ПЛК в рабочий режим все обычные стадии (SG) находятся в выключенном состоянии. Таким образом, у рассмотренных стадийных программ пока что нет реальной возможности начать работу (поскольку цепи схемы игнорируются, если их стадия неактивна).



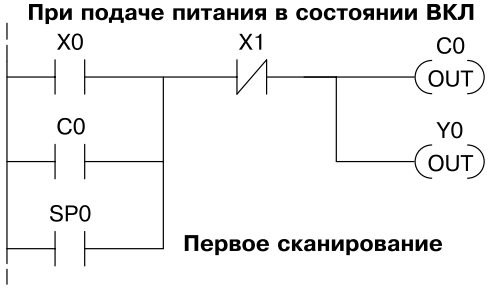
Начальные стадии

Допустим, что мы хотим всегда начинать в состоянии ВЫКЛ (двигатель выключен), именно так работает RLL-программа. Начальной стадией (ISG) по определению является стадия, активизируемая при включении питания. В модифицированной программе, изображенной справа, мы изменили тип стадии S0 на ISG. Это значит, что после включения питания ПЛК обязательно проверит контакт X0, поскольку стадия S0 будет активна. После включения питания начальная стадия (ISG) работает также как и любая другая стадия!

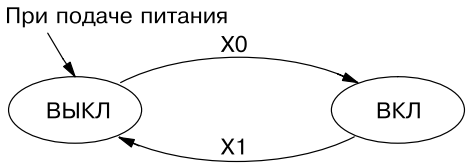
Мы можем изменить обе программы так, чтобы при включении питания (пуске контроллера) включался двигатель. В приведенной ниже программе RLL мы должны добавить реле первого сканирования SP0, запирающее C0 во включенном состоянии. В приведенном примере справа стадийной программы мы просто сделаем стадию S1 начальной стадией (ISG) вместо S0.



При подаче питания в состоянии ВКЛ



Мы можем пометить, как показано справа, состояние, активизирующееся при включении питания, что поможет не забыть использовать соответствующие начальные стадии при создании стадийной программы. Допускается иметь столько начальных стадий, сколько требуется процессу.



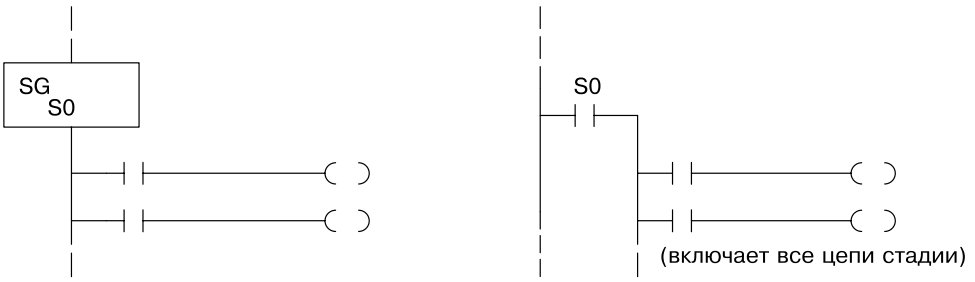
Функция битов стадии

Напоминаем, что стадия — это раздел программы, который либо активен, либо неактивен в данный момент времени. Все биты стадий (S0 — Sxxx) располагаются в регистре отображения PLC как отдельные биты статуса. Каждый бит стадии принимает в любой момент времени **булевы** значения 0 или 1.

При выполнении программы цепи программы всегда считываются сверху вниз и слева направо. На следующем чертеже показано влияние статуса бита стадии. Цепи схемы, расположенные ниже команды стадии и продолжающиеся вплоть до следующей команды стадии или конца программы, принадлежат стадии 0. Справа показана эквивалентная по функциональности схема. Когда S0 принимает значение истина, через две цепи проходит ток питания.

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

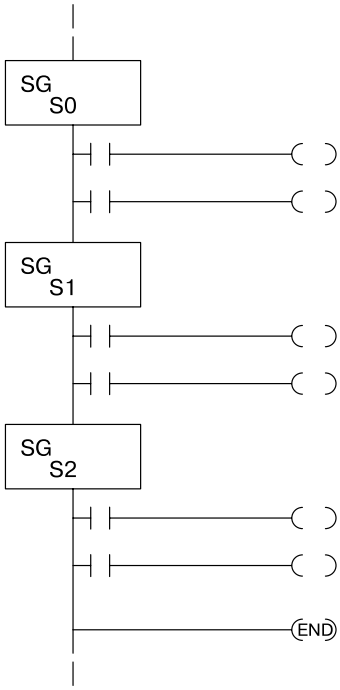
Вид реальной программы **Функционально эквивалентная программа**



Характеристики команды Stage (Стадия)

Прямоугольники стадий, расположенные слева в линию на шине питания, делят цепи программы на стадии. Ниже приведены правила, применимые к стадиям:

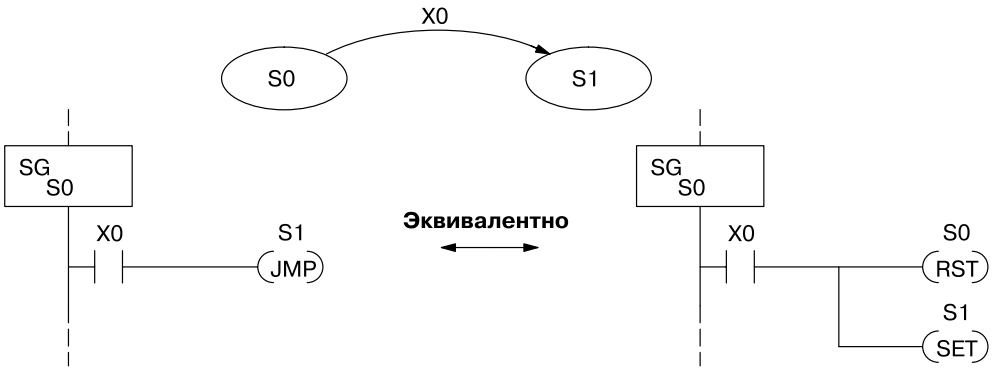
- **Выполнение** — при любом проходе выполняется логика только активных стадий.
- **Переходы** — действие команд перехода стадии проявляется только при следующем проходе включенных стадий.
- **Восьмеричная нумерация** — стадии нумеруются в восьмеричной системе, подобно точкам ввода/вывода и т.д. Так что запись «S8» не верна.
- **Количество стадий** — максимально допустимое количество стадий зависит от процессора.
- Дублирование запрещено — каждый номер стадии уникален и может быть использован лишь один раз.
- **Произвольный порядок** — можно пропускать номера стадий и выстраивать их в любом порядке.
- **Последняя стадия** — последняя стадия программы включает все цепи от прямоугольника стадии вплоть до последней обмотки.



Использование команды Stage Jump для переходов состояния

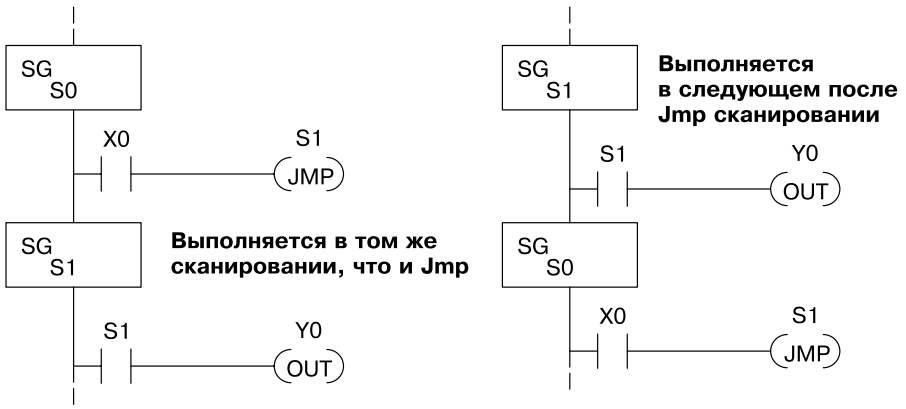
Команды перехода, установки и сброса стадий

Использованная команда Stage JMP (переход стадии) дезактивирует стадию, в которой она вызывается, и одновременно делает активной ту стадию, на которую она ссылается. Обратите внимание на переход состояния, изображенный ниже. При замыкании контакта X0 возникает переход состояния от S0 к S1. Показанные ниже два стадийных примера эквивалентны. Итак, команда Stage JMP состоит из двух команд: команды сброса (Stage Reset) текущей стадии и команды установки (Set Stage) той стадии, к которой нужно перейти.



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

- Команда перехода сбрасывает бит вызывающей ее стадии. Тем не менее, все цепи стадии будут выполняться во время текущего сканирования, *даже если они расположены ниже команды перехода!*
- Сброс проявится только при следующем сканировании, следовательно, стадия, в которой перед этим была выполнена команда перехода, станет неактивной и будет пропущена.
- Бит стадии, заданной в команде перехода, будет установлен сразу же, так что эта стадия будет выполнена при своем ближайшем появлении. В программе, показанной ниже слева, стадия S1 выполняется в том же сканировании, в котором в стадии S0 выполняется команда JMP S1. В примере справа стадия S1 будет выполнена только при следующем сканировании после выполнения команды JMP S1, поскольку стадия S1 располагается выше стадии S0.

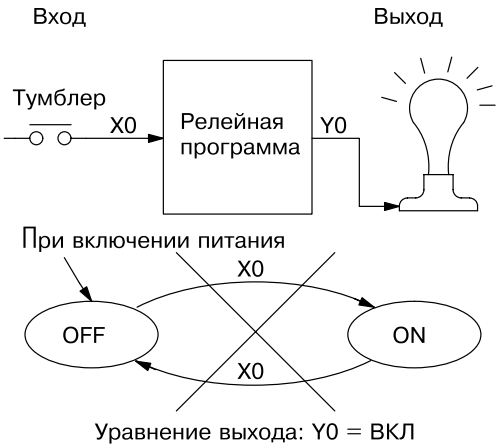


Замечание: в обоих примерах предполагается, что в начале стадия 0 активна, а стадия 1 неактивна.

Пример стадийной программы: контроллер включения/выключения лампочки

Процесс с 4-мя состояниями

В процессе, показанном справа, для управления электрической лампочкой мы используем обычную нефиксируемую кнопку. Релейная программа будет запирать вход переключателя, так что для включения лампочки нужно будет нажать и отпустить кнопку, а для выключения проделать это повторно (иногда это называют функцией тумблера). Конечно, можно приобрести механический переключатель со встроенными функциями вкл/выкл… Однако этот учебный пример достаточно забавен!

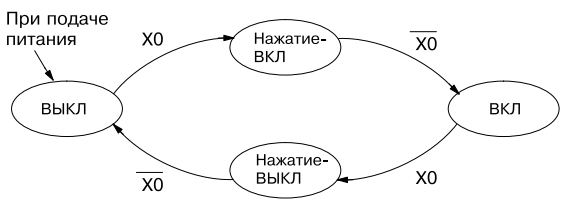


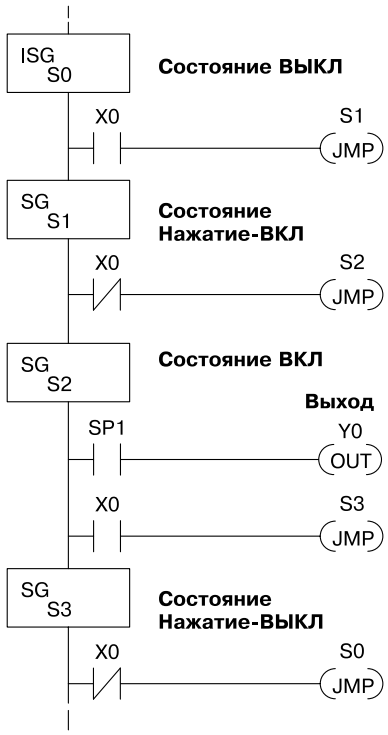
Теперь нарисуем диаграмму перехода состояния. Обычно первое желание — использовать X0 для обоих переходов (подобно примеру, показанному справа). Однако *это некорректно* (см. дальше).

Заметим, что этот пример отличается от примера с двигателем, поскольку у нас имеется только одна кнопка. При нажатии кнопки оба условия перехода пересекутся. Мы будем циклически проходить диаграмму состояния с предельно большой скоростью. При реализации с помощью стадий такое решение вызвало бы мигание света при каждом проходе (что, очевидно, нежелательно)! Правильное решение заключается в том, чтобы сделать нажатие и освобождение кнопки отдельными событиями. Взгляните на приведенную ниже новую диаграмму перехода состояния. При подаче питания мы входим в состояние ВЫКЛ. Когда нажимается переключатель X0, мы входим в состояние нажатие-ВКЛ. Когда он освобождается, мы входим в состояние ВКЛ. Заметьте, что X0 с чертой сверху обозначает НЕ X0. В состоянии ВКЛ последующий цикл нажатия и освобождения кнопки просто возвращает нас в состояние ВЫКЛ. Теперь у нас есть два уникальных состояния (ВЫКЛ и ВКЛ), используемых после освобождения кнопки, что и требовалось для решения задачи управления.

Эквивалентная стадийная программа показана справа. Поскольку желаемым состоянием при подаче питания является ВЫКЛ, мы делаем S0 начальной стадией (ISG). В состоянии ВКЛ добавим специальный контакт реле SP1, который всегда замкнут.

Заметим, что даже при условии роста сложности наших программ, диаграмма перехода состояния по-прежнему просто и понятно соответствует стадийной программе!





Четыре этапа написания стадийной программы

Вы уже вероятно заметили, что при решении каждого примера мы придерживаемся одной и той же последовательности действий. Если вы проработаете все примеры этой главы то, возможно, автоматически запомните эту последовательность. Тем не менее, при решении задачи полезно в качестве руководства иметь контрольный пошаговый перечень действий. Суммируя все вышесказанное, приведем основные этапы процедуры разработки стадийной программы.

1. Дайте словесное описание приложения.

Опишите все функции процесса своими словами. Начните с перечисления того, что происходит в первую очередь, во вторую и т.д. Если обнаружите, что одновременно происходит слишком много вещей, попытайтесь разбить задачу на несколько процессов. Помните, что процессы могут взаимодействовать друг с другом для координации их совместной деятельности.

2. Нарисуйте блок-схему.

Входы предоставляют процессу всю информацию, необходимую для принятия решений, а выходы подключаются ко всем устройствам, которые управляются процессом.

- Создайте списки входов и выходов процесса.
- Присвойте номера точек ввода/вывода (X и Y) физическим входам и выходам.

3. Нарисуйте диаграмму перехода состояния.

Диаграмма перехода состояния описывает центральную функцию блок-схемы, чтение входов и генерацию выходов.

- Идентифицируйте и назовите состояния процесса.
- Идентифицируйте событие(я), требуемое(ые) для каждого перехода между состояниями.
- Обеспечьте, чтобы у процесса был способ возобновления работы, либо чтобы он был циклическим.
- Выберите для вашего процесса состояние, активизируемое при подаче питания.
- Запишите уравнения выхода.

4. Напишите стадийную программу.

Преобразуйте диаграмму перехода состояния в стадийную программу.

- Сделайте каждое состояние стадией. В процессорах DL430 и DL440 доступно вплоть до 384 стадий одновременно. В процессорах DL450 их количество может достигать 1024 (1777 восьмеричных).
- Поместите логику перехода внутрь стадии, с которой начинается каждый переход (все стрелки стадии указывают наружу).
- Используйте начальную стадию (ISG) для всех состояний, которые должны быть активны при подаче питания.
- Разместите выходы или действия в соответствующих стадиях.

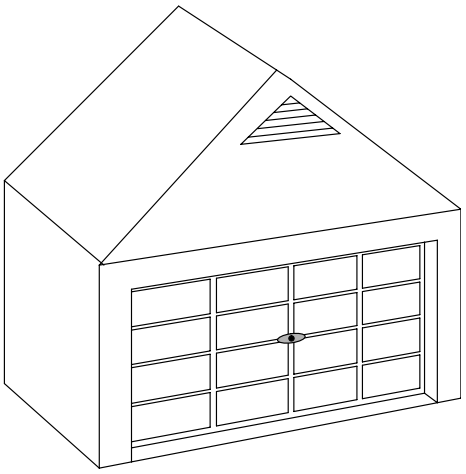
Вы заметите, что этапы с 1 по 3 подготавливают нас к написанию стадийной программы на этапе 4. Однако, благодаря предварительной подготовке, программа, в сущности, пишет сама себя. Скоро вы будете способны, начав со словесного описания приложения, закончить создание стадийной программы за один прием.

Пример стадийной программы: управление дверью гаража

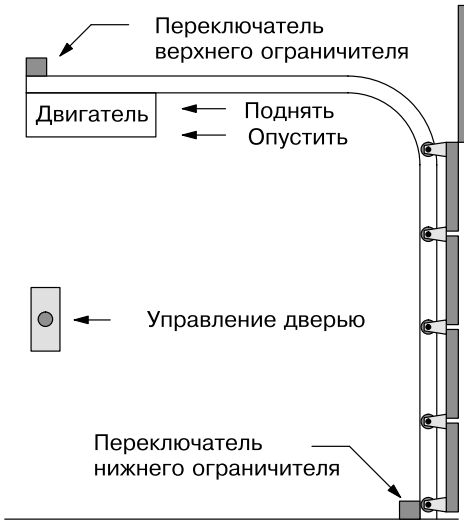
Пример управления дверью гаража

В этом примере стадийного программирования мы создадим контроллер управления дверью гаража. В надежде, что большинство читателей знакомы с подобным приложением, мы кроме прочего еще и весело проведем время!

Первым делом мы должны описать, как работает устройство открывания двери. Начнем с выполнения основных действий, добавляя впоследствии дополнительные возможности (стадийные программы очень легко модифицировать).



У нашего контроллера двери гаража есть двигатель, который по команде поднимает или опускает дверь. Чтобы поднять дверь, владелец гаража один раз нажимает и отпускает кнопку. После того как дверь поднята, другой цикл нажать-отпустить опустит дверь.

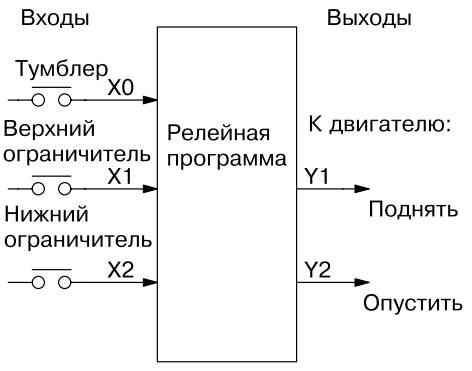


Для того чтобы определить входы и выходы системы, иногда бывает полезно сделать набросок ее основных компонентов, подобный показанному справа виду со стороны двери. У двери есть верхний и нижний ограничители. Переключатель каждого ограничителя срабатывает только тогда, когда дверь дойдет до конца в соответствующем направлении. В промежуточном положении двери переключатели ограничителей не замкнуты.

Двигатель имеет два командных входа: поднять и опустить. Когда оба входа неактивны, двигатель останавливается. Команда двери выдается с помощью простой кнопки. Независимо от того, встроена ли кнопка в стену, или находится на пульте дистанционного управления, все команды управления дверью логически исключают друг друга подобно паре контактов переключателя.

Рисуем блок-схему

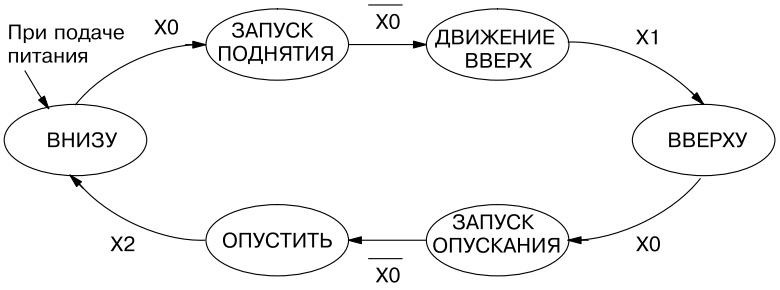
Блок-схема контроллера показана на рисунке справа. Здесь вход X0 подается от кнопки управления дверью. Вход X1 включается, когда дверь полностью достигает верхнего положения. Вход X2 включается, когда дверь полностью достигает нижнего положения. Когда дверь находится в промежуточном положении, оба переключателя ограничителей разомкнуты. Контроллер имеет два выхода для управления двигателем. Y1 — это команда вверх (поднять дверь), а Y2 — команда вниз (опустить дверь).



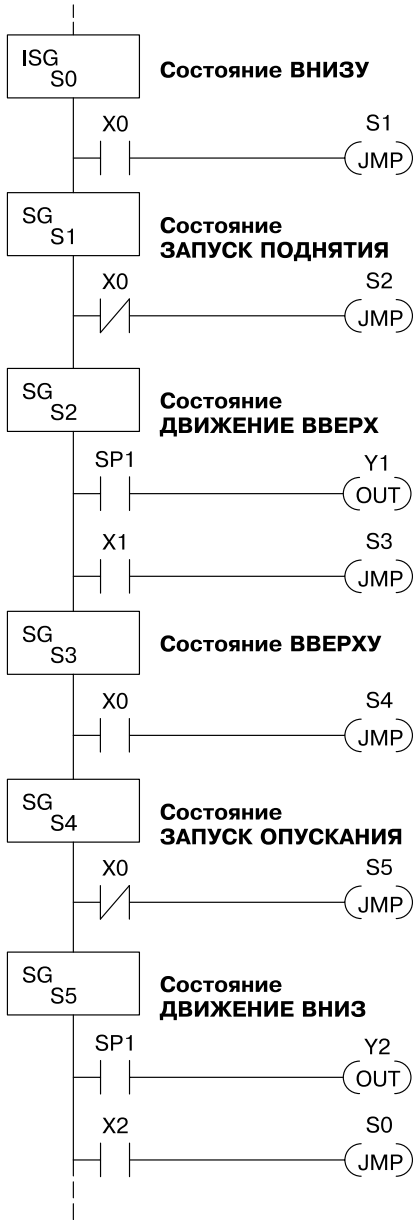
Рисуем диаграмму состояния

Теперь мы готовы нарисовать диаграмму перехода состояния. Аналогично предыдущему примеру с контроллером лампочки, это приложение также использует только один переключатель для ввода команды. Взгляните на следующий рисунок.

- Когда дверь находится внизу (состояние ВНИЗУ), ничего не происходит, пока не будет включен X0. Нажатие и освобождение кнопки переводит процесс в состояние ЗАПУСК ПОДНЯТИЯ, в котором выход Y1 включен и заставляет двигатель поднимать дверь.
- Мы переходим в состояние ВВЕРХУ, когда срабатывает переключатель верхнего ограничителя (X1) и выключает двигатель.
- Затем вновь ничего не происходит вплоть до следующего цикла нажать-отпустить X0. Он переводит процесс в состояние ЗАПУСК ОПУСКАНИЯ, включая выход Y2 и заставляя двигатель опустить дверь. Мы возвращаемся в состояние ВНИЗУ, когда срабатывает переключатель нижнего ограничителя (X2).



Уравнения выхода: Y2 = ОПУСТИТЬ Y1 = ПОДНЯТЬ



Эквивалентная стадийная программа показана справа. С этого момента мы будем полагать, что при подаче питания дверь находится внизу, так что желаемым состоянием при подаче питания является состояние ВНИЗУ. Сделаем S0 начальной стадией (ISG). Стадия S0 остается активной до тех пор, пока не активизируется кнопка управления дверью. Затем мы переходим (JMP) к стадии ОТЖАТЬ, S1.

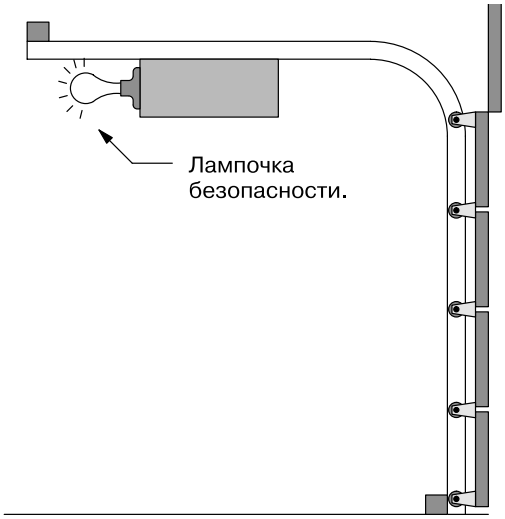
Цикл нажать-отпустить кнопку проводит нас через стадию S1 к стадии ДВИЖЕНИЕ ВВЕРХ, S2. Мы используем всегда замкнутый контакт SP1, чтобы инициировать команду двигателя ДВИЖЕНИЕ ВВЕРХ, Y1. Когда дверь достигает верхнего положения, активизируется переключатель верхнего ограничителя X1. Это переводит нас в стадию ВВЕРХУ S3, где мы ждем следующей команды управления дверью.

В стадии ВВЕРХУ (S3) цикл нажать-отпустить кнопку переведет нас в стадию ДВИЖЕНИЕ ВНИЗ (S5), где мы активизируем Y2, командуя двигателю опустить дверь. Это продолжается до тех пор, пока дверь не достигнет переключателя нижнего ограничителя, X2. Когда X2 закрывается, мы переходим от стадии S5 к стадии ВНИЗУ (S0), с которого мы начинали.

Примечание. Единственным отличием начальной стадии (ISG) является то, что она автоматически активизируется при подаче питания. В дальнейшем она ведет себя подобно любой другой стадии.

Добавляем лампочку безопасности

Далее мы добавим новую деталь в систему управления дверью — лампочку безопасности. Это лучший способ изменения программы: вначале добиться, чтобы работала основная функция, что уже сделано, а затем добавлять второстепенные детали.

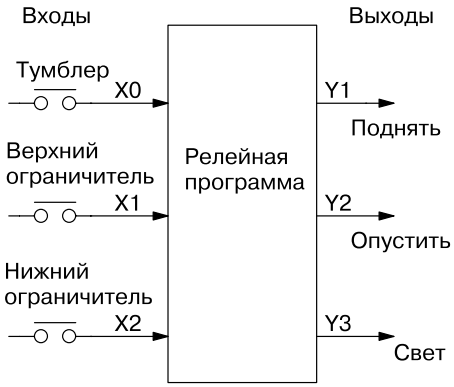


Лампочка безопасности является стандартом во многих коммерческих системах управления гаражной дверью. На рисунке справа она установлена на корпусе двигателя. Лампочка загорается при любом движении двери, оставаясь включенной еще приблизительно 3 минуты.

Данная часть упражнения продемонстрирует использование параллельных состояний в нашей диаграмме. Вместо использования команды JMP мы воспользуемся командами установки и сброса.

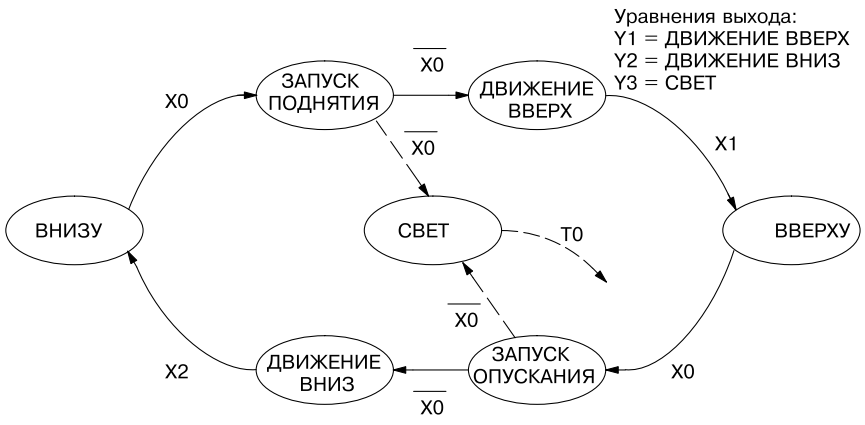
Модифицируем блок-схему и диаграмму состояния

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



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

Можно рассматривать состояние СВЕТ как процесс, параллельный состояниям ДВИЖЕНИЕ ВВЕРХ и ДВИЖЕНИЕ ВНИЗ. Пути, ведущие к состоянию СВЕТ, являются не переходом (Stage JMP), а командой установки состояния (State Set). В стадии СВЕТ мы поместим трехминутный таймер. Когда он истечет, установится бит таймера T0, возвращая стадию СВЕТ в исходное состояние. Путь из стадии СВЕТ никуда не ведет, показывая, что стадия СВЕТ становится неактивной, и лампочка просто гаснет.



Использование таймера внутри стадии

Законченная модифицированная программа показана на рисунке справа. Закрашенные области обозначают дополнения к программе. В стадии ЗАПУСК ПОДНЯТИЯ S1 мы добавляем команду «Установки бита» стадии S6. Когда открывается контакт X0, мы переходим из S1 в два новых активных состояния: S2 и S6. В состоянии ЗАПУСК ОПУСКАНИЯ S4 мы делаем аналогичные дополнения. Таким образом, всякий раз, когда кто-нибудь нажимает кнопку управления дверью, загорается свет.

Большинство новичков в стадийном программировании забеспокоились бы о том, как разместить в схеме и пронумеровать стадию СВЕТ. Это не имеет значения!

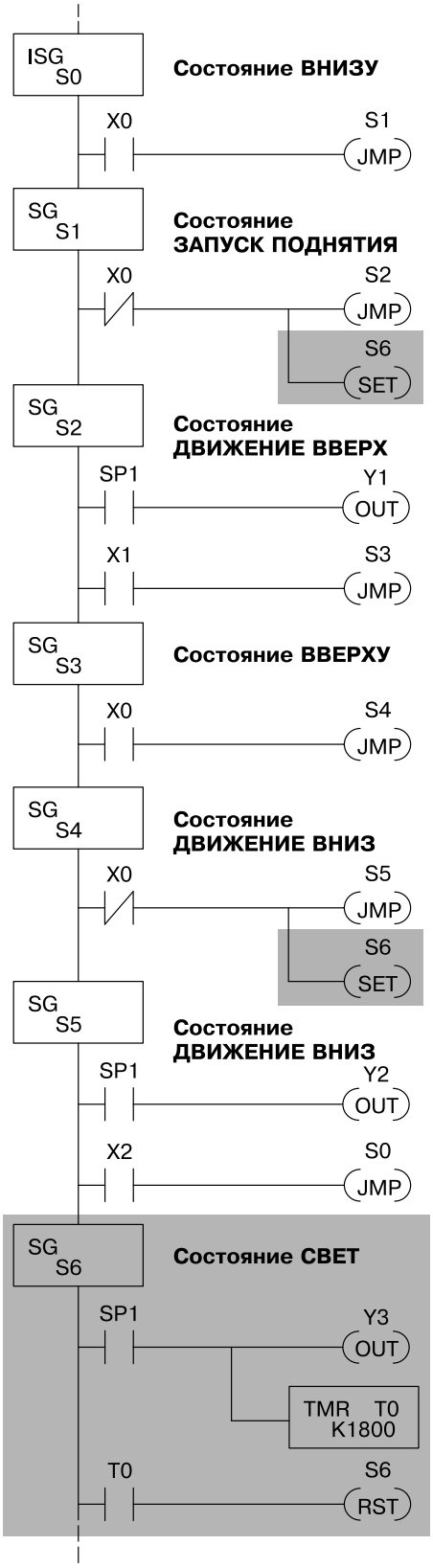
- Выберите любой неиспользуемый номер и используйте его для новой стадии и всех соответствующих ссылок из других стадий.
- Местоположение в программе не существенно, так что мы помещаем стадию в конце программы.

Вы, возможно, подумали, что каждая стадия должна находиться прямо под той стадией, которая на нее указывает. Хотя подобная практика неплоха, она не является обязательной (что хорошо, поскольку два положения команды установки S6 делают это невозможным). Номера стадий вместе с тем, как они используются, определяют пути перехода.

В стадии S6 мы включаем лампочку безопасности, возбуждая Y3. Контакт специального реле SP1 всегда замкнут. Таймер T0 работает с периодом 0.1 секунды. Для достижения 3-минутного интервала проведем следующие вычисления:

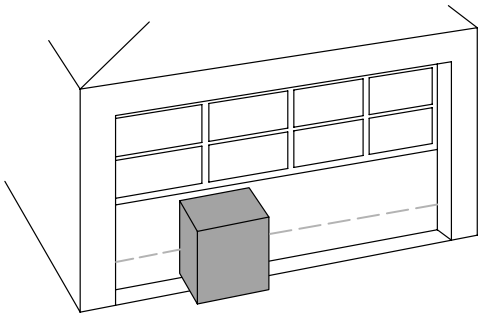
K=(3 мин. x 60 с/мин)/(0.1 с/отсчет)
K= 1800 отсчетов

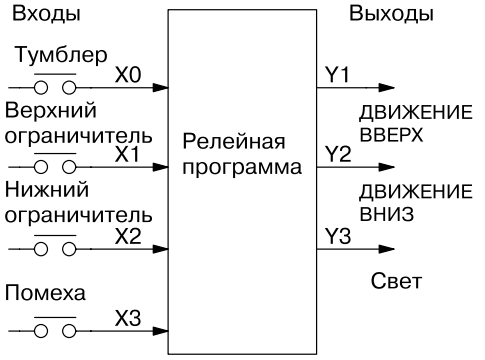
Таймер работает всегда, когда активна стадия S6. Соответствующий бит таймера T0 устанавливается, когда таймер истекает. Так что спустя 3 минуты выполнится T0=1, и команда Сбросить S6 переведет стадию в неактивное состояние. Пока стадия S6 активна и горит свет, переходы стадий по основному пути совершаются как обычно, независимо от стадии 6. Т.е., дверь может подниматься, опускаться или выполнять любые другие действия, но свет будет гореть точно в течение 3 минут.



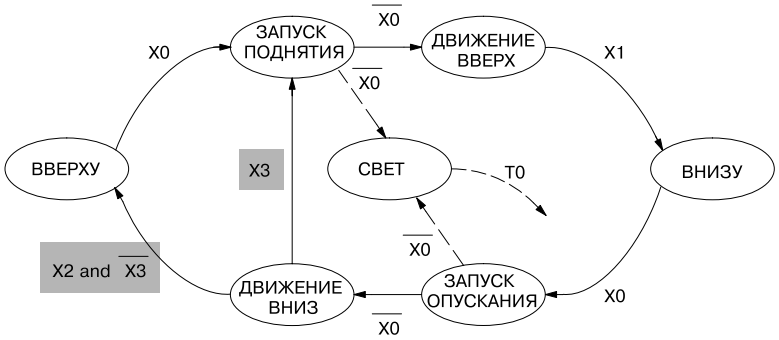
Добавляем аварийную остановку

Сегодня некоторые устройства управления дверью гаража умеют обнаруживать какой-либо объект под дверью. Это приводит к остановке двери, когда она опускается. Обычно снабженная фотоэлементом («электронным глазом») опускающаяся дверь остановится и начнет подниматься. Мы определим наш элемент безопасности с аналогичными функциями, добавив вход от фотоэлемента в блок-схему, как изображено справа. X3 включится, если на пути двери окажется какой-либо объект.





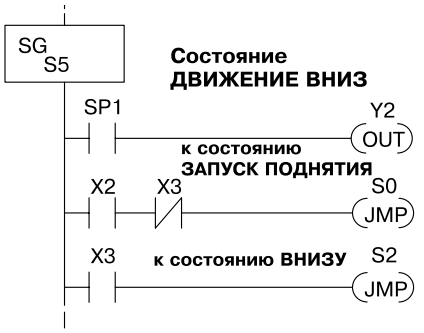
Далее мы сделаем простое дополнение к диаграмме перехода состояния, показанное в затененных областях на следующем рисунке. Обратите внимание на новую траекторию перехода над состоянием ДВИЖЕНИЕ ВНИЗ. Если мы, опуская дверь, обнаружим помеху (X3), то совершим переход к состоянию ЗАПУСК ПОДНЯТИЯ. Мы делаем этот переход вместо перехода непосредственно в состояние ДВИЖЕНИЕ ВВЕРХ, чтобы предоставить выходу Y2 (ДВИЖЕНИЕ ВНИЗ) одно сканирование до отключения прежде, чем будет возбужден выход Y1 (ДВИЖЕНИЕ ВВЕРХ).



Исключающие переходы

Теоретически входы нижнего ограничителя (X2) и помехи (X3) могли бы возбудиться одновременно. В этом случае, мы бы «перепрыгнули» одновременно к состояниям ЗАПУСК ПОДНЯТИЯ и ВНИЗУ, что не имеет смысла.

Вместо этого мы отдаем приоритет помехе, изменяя условие перехода к состоянию ВНИЗУ на [X2 AND NOT X3]. Это гарантирует приоритет события помехи. Изменения, которые необходимо сделать в схеме стадии ДВИЖЕНИЕ ВНИЗ (S5), показаны справа. Первая цепь остается неизменной. Вторая и третья цепи реализуют необходимые переходы. Обратите внимание на использование для X3 противоположного контакта реле, в результате чего стадия будет выполнять только одну команду JMP.

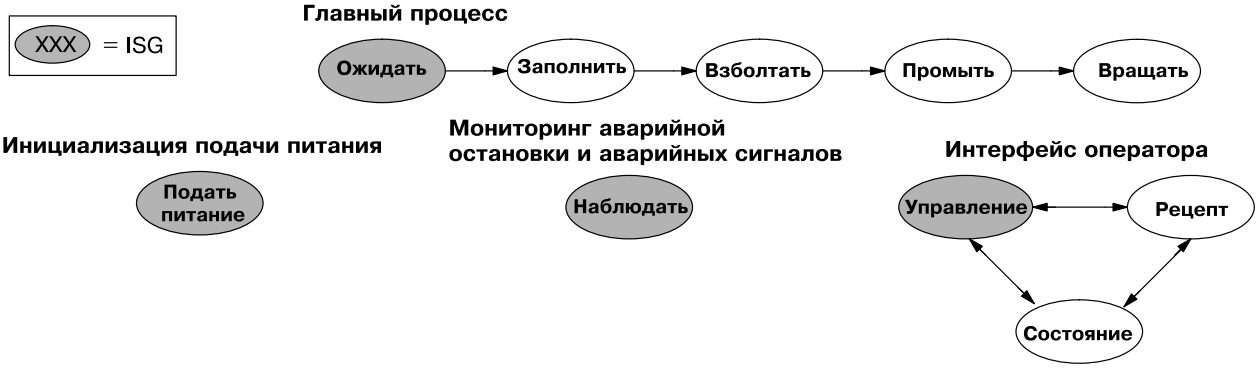


Анализ проекта стадийной программы

Организация стадийной программы

До сих пор в примерах данной главы для представления основного процесса использовалась одна замкнутая диаграмма состояния. Однако мы могли реализовать в стадиях несколько процессов, все в одной и той же программе. Новички в стадийном программировании иногда пытаются активизировать и дезактивировать стадию в каждом сканировании, основываясь на неверном предположении, что в любой момент времени может быть активна только одна стадия. Поместите цепи программы, которые нужно выполнять в каждом проходе, в постоянно активной стадии.

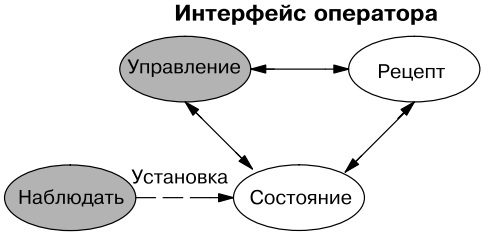
На следующем рисунке показано обычное приложение. Во время его работы запущены все процессы: главный процесс основного производства, инициализация при включении питания, мониторинг аварийной остановки и аварийных сигналов и интерфейс оператора. При подаче питания, как показано, начинают работать четыре начальных стадии.



В типичном приложении отдельные последовательности стадий, показанных выше, работают следующим образом:

- **Инициализация** при включении питания. Эта стадия содержит цепи программы, задачи которых выполняются один раз при подаче питания. Ее последняя цепь сбрасывает стадию, так что данная стадия активна только для одного сканирования (либо нужного количества сканирований).
- **Главный процесс.** Эта последовательность стадий управляет сердцем процесса или механизма. Одно сканирование последовательности представляет один цикл машины или одну загрузку процесса.
- **Мониторинг аварийной остановки и аварийных сигналов.** Эта стадия всегда активна, поскольку она отслеживает ошибки, которые могли бы указывать на возникновение опасной ситуации или потребовать аварийной остановки. Общим для этой стадии является сброс стадий в главном процессе либо где-то еще для их инициализации после ошибочной ситуации.
- **Интерфейс оператора.** Это другая задача, которая всегда должна быть активной и готовой ответить оператору. Интерфейс оператора может изменять режимы работы и т.п. независимо от текущего шага главного процесса.

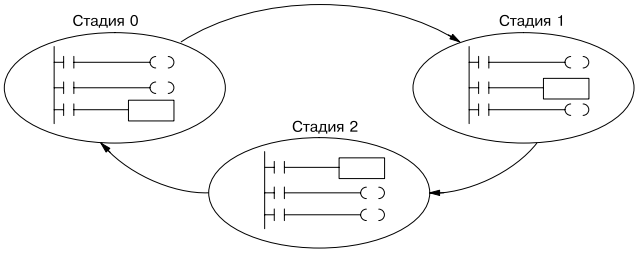
Хотя это и отдельные процессы, между ними может существовать согласованность. Например, в случае возникновения ошибочной ситуации, стадия Состояние может пожелать автоматически переключить интерфейс оператора в режим состояния, чтобы вывести информацию об ошибке, как изображено справа. Стадия наблюдения может установить бит стадии для Состояния и сбросить стадии Управление и Рецепт.



Аварийная остановка и мониторинг

Как команды работают внутри стадий

Можно рассматривать состояния или стадии просто как отдельные части нашей релейной программы, как изображено на следующем рисунке. Каждая стадия содержит только те цепи программы, которые необходимы для соответствующего состояния процесса. Логика выхода из стадии содержится внутри этой стадии. Выбрать цепи программы, активные при подаче питания, очень легко, просто воспользуйтесь типом стадии «начальная стадия» (ISG).



Большинство команд работают так же как и в стандартном RLL. Стадию можно рассматривать как миниатюрную RLL-программу, которая либо активна, либо нет.

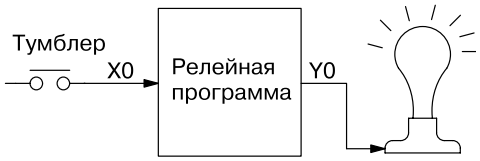
Выходные обмотки - как и следовало ожидать, выходные обмотки в активных стадиях будут включать или выключать выходы в зависимости от тока питания через обмотку. Однако заметим следующее:

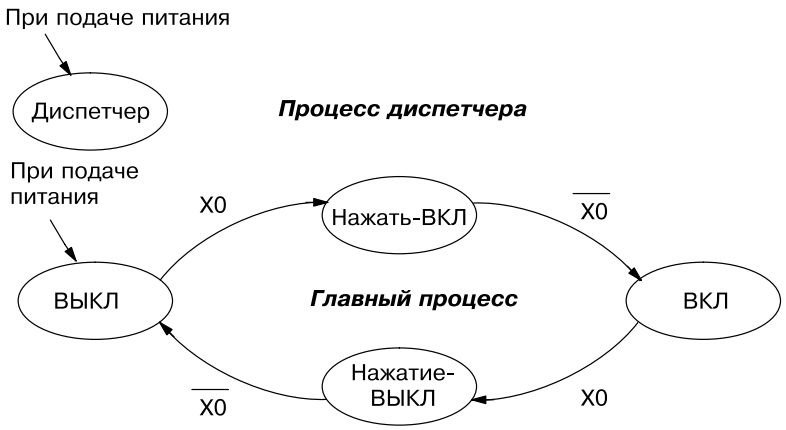
- Выходы работают как обычно, ссылка на каждый выход (такая как «Y3») используется только в одной стадии.
- Выходные обмотки автоматически выключаются на выходе из стадии. Однако команды установки и сброса не останутся «незавершенными» на выходе из стадии.
- На выходы можно ссылаться из нескольких стадий, если в каждый момент времени активна только одна из этих стадий.
- Если выходная обмотка одновременно управляется несколькими стадиями, то окончательное состояние выхода в каждом сканировании определяется активной стадией, ближайшей к концу программы. Так что, если хотите, чтобы несколько стадий управляли выходом с помощью логическим ИЛИ, используйте команду OROUT.

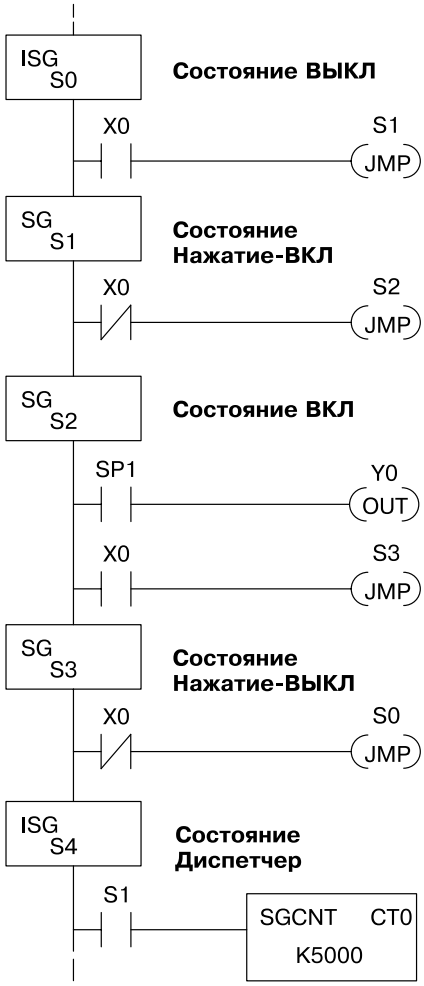
Одноразовые или PD-обмотки. Если нужно использовать в стадии обмотку Положительного дифференциала (PD), делайте это осторожно. Помните, что вход в такую обмотку должен выполнять переход 0-1. Если обмотка уже возбуждена при первом сканировании, то при активизации стадии PD-обмотка не будет работать. Именно из-за того, что не было перехода 0-1.
Альтернатива для PD-обмотки. Если есть задача, которую вы хотите выполнить всего лишь один раз (на 1-м сканировании), ее можно поместить в стадию, которая в том же сканировании переходит к следующей стадии .
Счетчик. При использовании счетчика внутри стадии необходимо, чтобы стадия активизировалась за одно сканирование до того, как вход счетчика выполнит переход 0-1. В противном случае, реального перехода не происходит, и счетчик не запустится. У обычной команды Счетчика (Counter) действительно есть ограничение: его невозможно перезапустить из других стадий, используя команду RST для бита счетчика. Однако готовое решение предоставляет специальный Стадийный счетчик (Stage Counter) (смотрите следующий абзац).
Счетчик стадий. Преимущество счетчика стадий состоит в том, что его значение может быть сброшено глобально из других стадий с помощью команды RST. У него есть вход значения, но нет входа сброса. Это единственное отличие от стандартной команды счетчика.
Барабанный командоаппарат. Реализация барабанного командоаппарата - это отдельный процесс, а также метод программирования, отличный от стадийного. Если нужно использовать барабанный командоаппарат и стадии, не забудьте поместить команду барабанного командоаппарата в стадию ISG, которая постоянно остается активной.

Использование стадии в качестве процесса-диспетчера

Вспомним пример контроллера электрической лампочки, разобранный ранее в данной главе. Для иллюстрации предположим, что мы хотим проследить за «производительностью» процесса лампочки, подсчитывая количество происходящих циклов ВКЛ-ВЫКЛ. Такое приложение потребует добавления простого счетчика, но ключевой проблемой здесь будет выбор места для счетчика.







Начинающие изучать стадийное программирование, как правило, пытаются поместить счетчик внутрь одной из стадий наблюдаемого процесса. Проблема при таком подходе в том, что стадия активна только часть времени. Чтобы счетчик мог считать, вход значения должен совершить переход от ВЫКЛ к ВКЛ, по меньшей мере, спустя одно сканирование после активизации его стадии. Для этого потребуется дополнительная логика, которая может оказаться чересчур сложной.

В данном случае, как показано выше, нам нужна всего лишь другая стадия-диспетчер, следящая за главным процессом. Счетчик внутри управляющей стадии использует бит стадии S1 главного процесса в качестве своего входа отсчетов. *Биты стадии, используемые в качестве связи, позволяют нам наблюдать за процессом.*

Заметим, что как стадия Диспетчер, так и стадия ВЫКЛ являются начальными стадиями. Стадия Диспетчер остается активной бесконечно долго.

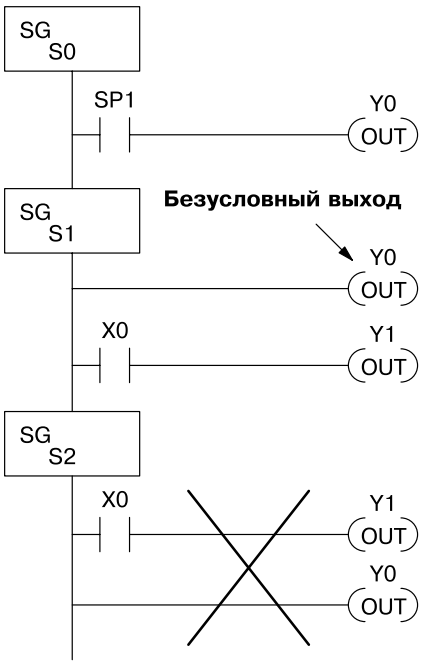
Счетчик стадий

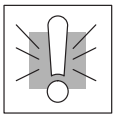
Счетчик в приведенном выше примере является специальным стадийным счетчиком. Заметим, что у него нет входа сброса. Счетчик обнуляется выполнением команды сброс (Reset) с именем бита счетчика (CT0 в данном случае). Преимущество счетчика стадий заключается в том, что его значение может быть сброшено глобально из других стадий. Стандартная команда Счетчик не обладает такой способностью глобального сброса. Ее все же можно использовать внутри стадии… однако, в таком случае единственным способом обнулить счетчик является использование его входа сброса.

Безусловные выходы

Как в большинстве примеров данной главы и в стадии 0 на рисунке справа вашему приложению может потребоваться, чтобы конкретный выход находился в состоянии ВКЛ независимо от того, когда активна конкретная стадия. До сих пор примеры всегда использовали последовательно с выходными обмотками специальный контакт реле SP1 (постоянно замкнутый).

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

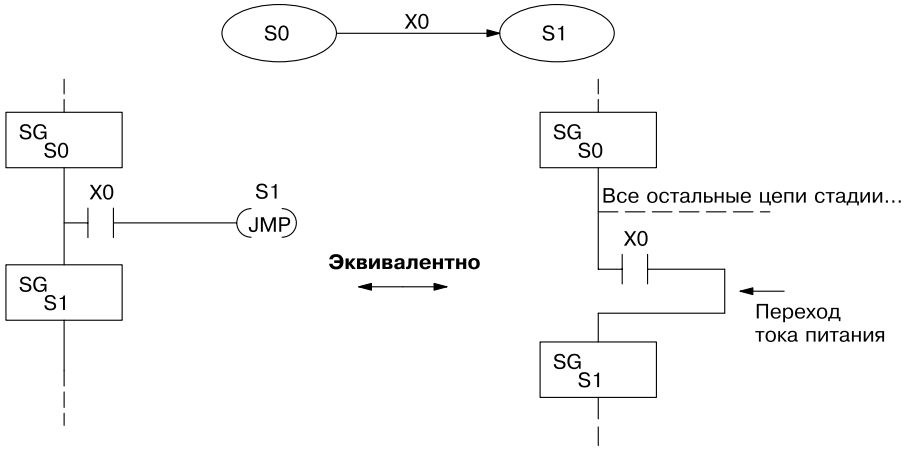




Предупреждение. Другие безусловные выходы стадии необязательно останутся подключенными во время активности стадии. Справа Y0 в стадии 2 изображен как безусловный выход, однако он получает ток питания от вышерасположенной цепи. Таким образом, статус Y0 совпадет со статусом Y1 (что неправильно).

Метод перехода тока питания

При обсуждении переходов состояния было показано, как команда Stage JMP делает текущую стадию неактивной, а следующую стадию (заданную в JMP) — активной. В качестве альтернативного способа ввода такого перехода в **Direct**SOFT можно использовать для стадийных переходов метод тока питания. Основным требованием является, чтобы текущая стадия была расположена в программе прямо над следующей (к которой совершается переход) стадией. Такое расположение иллюстрируют на следующей диаграмме стадии S0 и S1, соответственно.



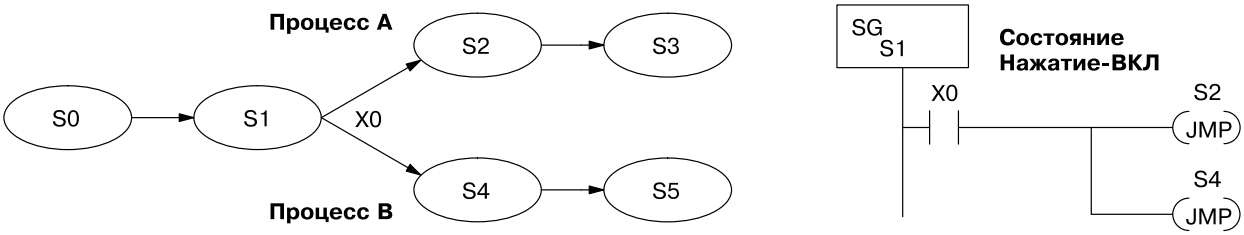
Напоминаем, что команда Stage JMP может появляться в любом месте текущей стадии, результат от этого не изменится. Однако переходы тока питания (показанные выше) обязаны быть последней цепью стадии. Все остальные цепи стадии должны им предшествовать. Метод перехода тока питания реализуем и на ручном программаторе, для следующей стадии надо просто следовать условию перехода и команде Stage.

Метод перехода тока питания исключает одну команду Stage JMP, и это его единственное преимущество. Однако теперь без использования команды Stage JMP программу становится труднее изменить. Поэтому мы советуем для большинства программ использовать переходы Stage JMP.

Понятия параллельной обработки

Параллельные процессы

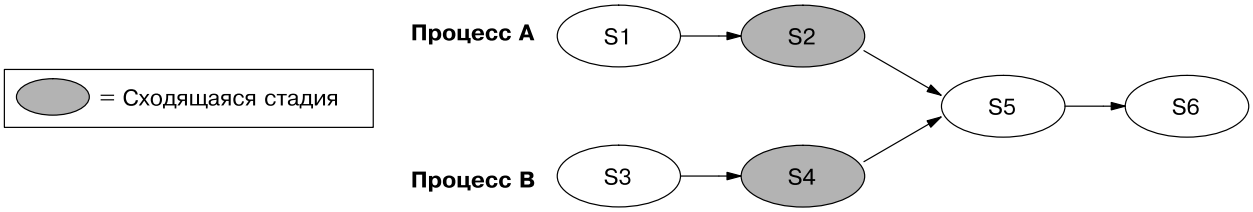
Ранее в этой главе мы обсудили, как состояние может переходить в одно из остальных состояний, это называется исключающим переходом. В других случаях может понадобиться разветвить процесс одновременно на два или более параллельных процесса, как показано ниже. Можно использовать, как показано, только команды JMP, либо мы могли бы использовать одну команду JMP и команду(ы) Установки бита стадии, Set Stage bit, (чтобы выйти из S1, по меньшей мере одна команда должна быть командой JMP). Запомните, выполняются все команды стадии, даже при выполнении перехода (JMP это не GOTO).



Заметим, что если нужно, чтобы стадии S2 и S4 активизировались точно в одном и том же сканировании, обе стадии должны быть одновременно размещены в программе ниже или выше стадии S1 (смотрите объяснение в нижней части страницы 7-7). В целом, параллельное ветвление это просто.

Схождение процессов

Теперь рассмотрим противоположный параллельному ветвлению случай, который называется *схождением процессов*. Под этим термином понимается, что мы останавливаем выполнение нескольких задач и продолжаем выполнять только одну задачу. На следующем рисунке в некоторый момент времени, когда стадии S2 и S4 совершают переход к S5, сходятся процессы A и B. Таким образом, S2 и S4 это сходящиеся стадии.

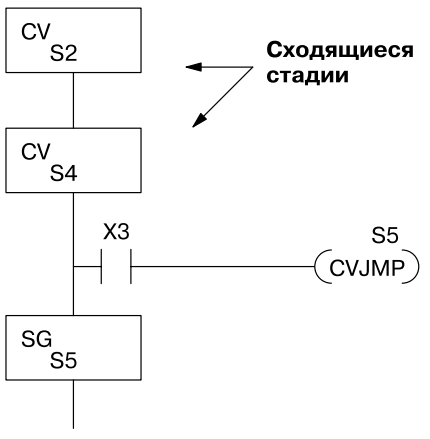


Сходящиеся стадии (CV)

X	√	√
430	440	450

Хотя принцип схождения достаточно прост, он вносит новое усложнение. По мере выполнения параллельной обработки, различные процессы почти никогда не заканчиваются одновременно. Другими словами, откуда нам знать, какая из стадий S2 или S4 финиширует последней? Это важный момент, поскольку мы должны решить, как перейти к стадии S5.

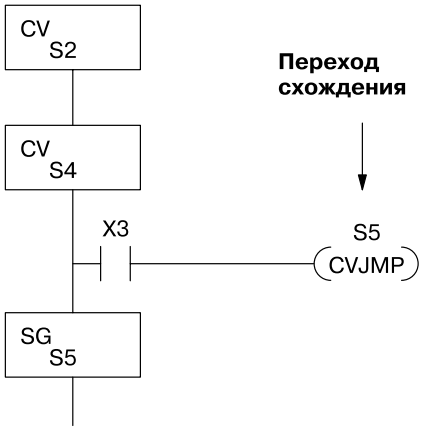
Решение заключается в том, чтобы координировать условие перехода вне сходящихся стадий. Мы выполним это с помощью типа стадии, разработанного специально для этой цели: сходящаяся стадия (тип CV). В приведенном справа примере необходимо, чтобы сходящиеся стадии S2 и S4 были сгруппированы так, как показано. **Между стадиями CV недопустимы никакие цепи программы!** Условие перехода (X3 в данном случае) должно быть помещено в последнюю сходящуюся стадию. Когда становятся активными все сходящиеся стадии в группе, условие перехода просто получает ток питания.



Переход схождения (CVJMP)

X	√	√
430	440	450

Вспомним, что когда все CV-стадии в группе становятся активны, на последнюю сходящуюся стадию просто подается ток питания. Для завершения сходящейся стадии нам потребуется новая команда перехода. Показанный справа переход схождения (CVJMP) совершит переход к стадии S5, когда активизируется X3 (как и следовало ожидать), но, кроме того, эта команда также автоматически сбросит все сходящиеся стадии в группе. Это делает переход CVJMP очень мощной командой. Заметим, что эта команда может быть использована только со сходящимися стадиями.



Основные принципы сходящихся стадий

Ниже сведены требования к использованию сходящихся стадий, включая некоторые советы по их более эффективному применению:

- Сходящаяся стадия должна использоваться в качестве последней стадии процесса, выполняющегося параллельно другому процессу (процессам). Переход к сходящейся стадии означает, что конкретный процесс заканчивается, и представляет собой точку ожидания окончания работы всех остальных процессов.
- Максимальное число сходящихся стадий, образующих группу, равно 17. Другими словами, в одну стадию может сойтись не более 17 стадий.
- Сходящиеся стадии одной группы должны размещаться в программе вместе, соединяясь одной шиной питания без какой-либо другой логики между ними.
- В пределах группы сходящиеся стадии могут располагаться сверху вниз в любом порядке. Неважно, какая стадия группы является последней, поскольку прежде, чем ток питания будет подан на последнюю стадию, активными должны стать все сходящиеся стадии.
- Последняя в группе сходящаяся стадия может содержать релейную логику в пределах стадии. Однако эта логика не будет выполнена, пока не активизируются все сходящиеся стадии группы.
- Под переходом схождения (CVJMP) понимается специальный метод, используемый для перехода от группы сходящихся стадий к следующей стадии. CVJMP сбрасывает все сходящиеся стадии в группе и активизирует стадию, заданную в команде перехода.
- Команда CVJMP должна использоваться исключительно в сходящейся стадии, и неприменима в обычных или начальных стадиях.
- Сходящиеся стадии или команды CVJMP нельзя использовать в подпрограммах или обработчиках прерываний.

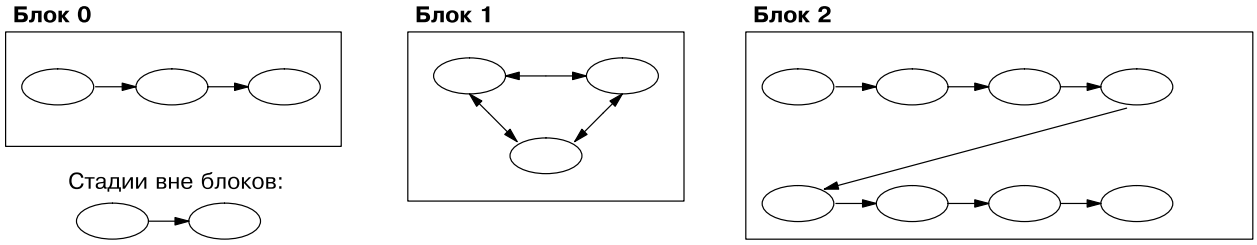
Управление большими программами

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

Блоки стадий (BLK, BEND)

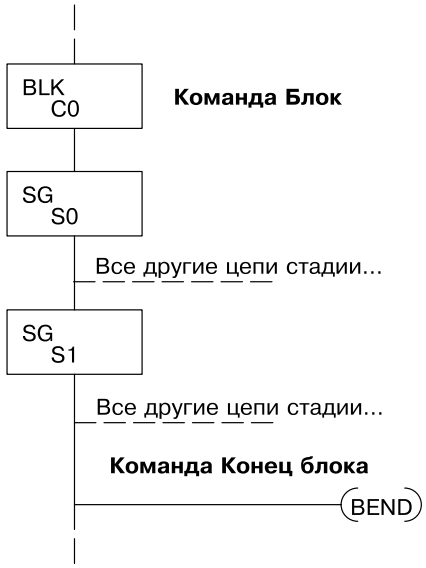
X	√	√
430	440	450

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



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

Блок стадий отделяется от остальной программы специальными начальной и завершающей командами. На рисунке справа команда BLK вверху обозначает начало блока стадий. Внизу команда BEND обозначает конец блока. Стадии между этими граничными метками (S0 и S1 в данном случае) и все связанные с ними цепи образуют блок.



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

Вызов блока (BCALL)

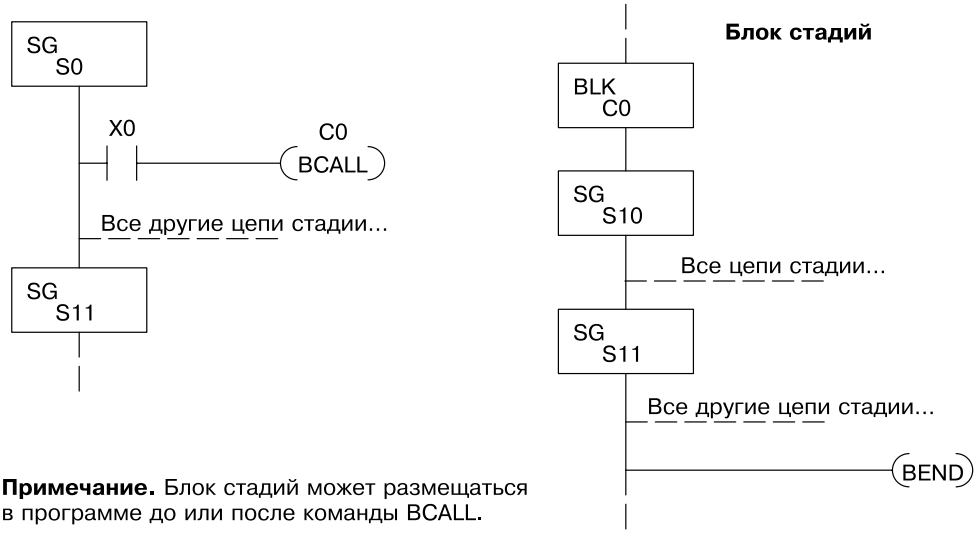
X	√	√
430	440	450

Цель команды Вызов блока — активизировать блок стадий. При включении питания или после переходов в рабочий режим все блоки стадий и их стадии неактивны. Как показано на следующем рисунке, команда Вызов блока имеет тип выходной обмотки. Когда контакт X0 замкнут, BCALL заставит стать активным блок стадий, на который ссылается команда (C0). Когда BCALL отключается, соответствующий блок стадий и стадии внутри него становятся неактивными.

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

Когда блок стадий становится активным, автоматически на том же сканировании становится активной первая стадия блока. «Первая» стадия блока — это стадия, расположенная в программе непосредственно под командой блока (BLK). Так что роль этой стадии аналогична роли стадии начального типа, рассмотренной ранее.

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



Примечание. Блок стадий может размещаться в программе до или после команды BCALL.

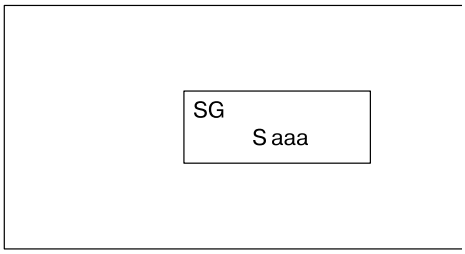
Команда BCALL может быть использована многими способами и в различных контекстах, так что найти для нее наилучшее применение может оказаться непросто. Запомните, что цель блоков стадий состоит в том, чтобы помочь организовать задачу приложения, группируя взаимосвязанные стадии. Запомните, что начальные стадии должны располагаться вне блоков стадий.

Команды RLL PLUS

Стадия (SG)

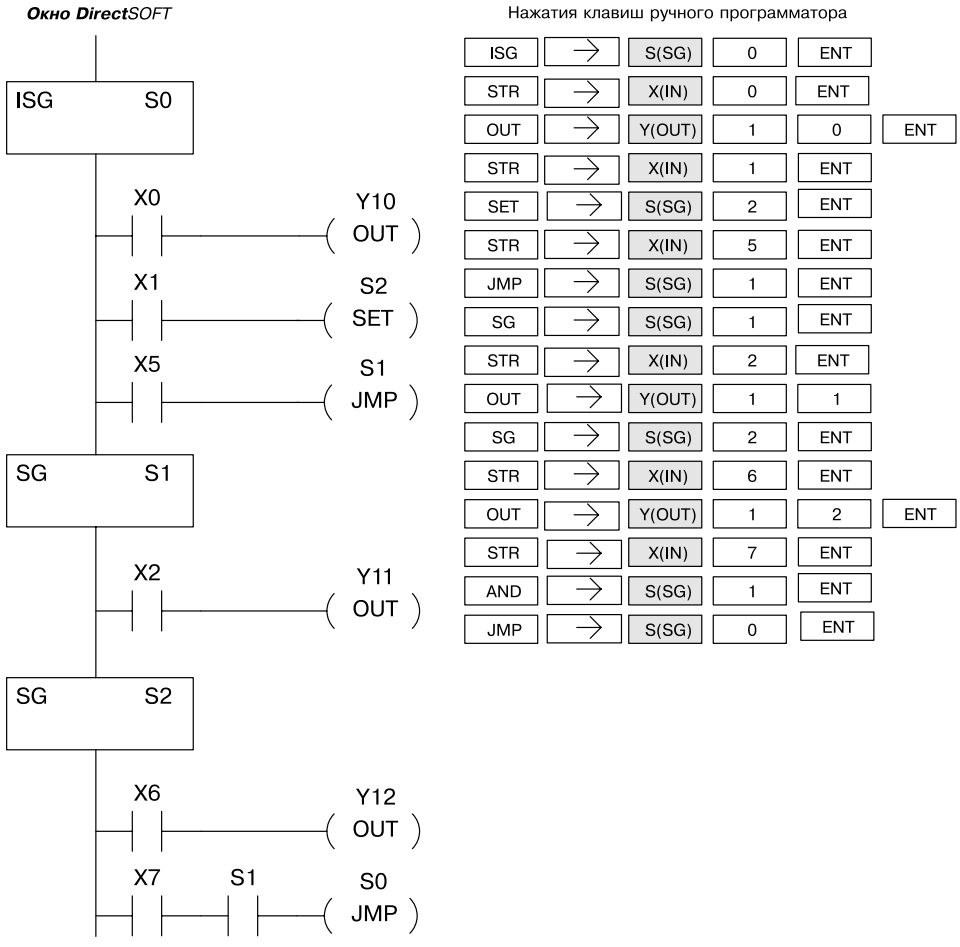
√	√	√
430	440	450

Команда Стадия используется для создания структурированных программ RLL PLUS. Стадии — это сегменты программы, которые могут быть активизированы командами переходной логики, перехода или установки бита стадии, выполняемыми из активной стадии. Стадии дезактивируются спустя одно сканирование после выполнения команды переходной логики, перехода или сброса стадии.



Тип данных операнда	Диапазон DL430	Диапазон DL440	Диапазон DL450
	aaa	aaa	aaa
Стадия S	0-377	0-777	0-1777

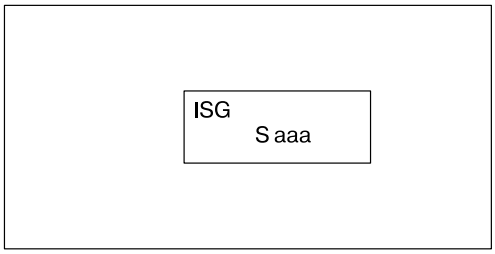
Следующий пример — это простая программа RLL PLUS. Для структурирования эта программа использует команды Начальная стадия, Стадия и Переход.



Начальная стадия (ISG)

√	√	√
430	440	450

Команда Начальная стадия обычно используется в качестве первого сегмента программы RLL[PLUS]. Начальные стадии активизируются при переходе процессора в рабочий режим, обеспечивая тем самым стартовую точку программы. Начальные стадии также активизируются командами переходной логики, перехода или установки бита стадии, которые выполняются из активной стадии. Начальные стадии дезактивируются спустя одно сканирование после выполнения команды переходной логики, перехода или сброса стадии. Начальные стадии в программе можно использовать многократно.

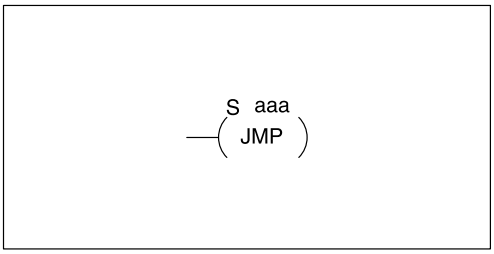


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
		aaa	aaa	aaa
Стадия	S	0-377	0-777	0-1777

Переход (JMP)

√	√	√
430	440	450

Команда Переход позволяет программе перейти из активной стадии, в которой содержится команда, к другой стадии, на которую эта команда ссылается. Переход произойдет, когда логический узел входа принимает значение истина. Активная стадия, содержащая команду Переход, будет дезактивирована спустя одно сканирование после выполнения команды Переход.

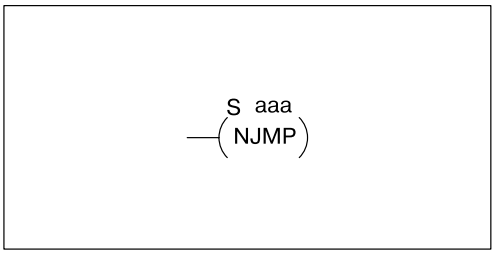


Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
		aaa	aaa	aaa
Стадия	S	0-377	0-777	0-1777

Не-переход (NJMP)

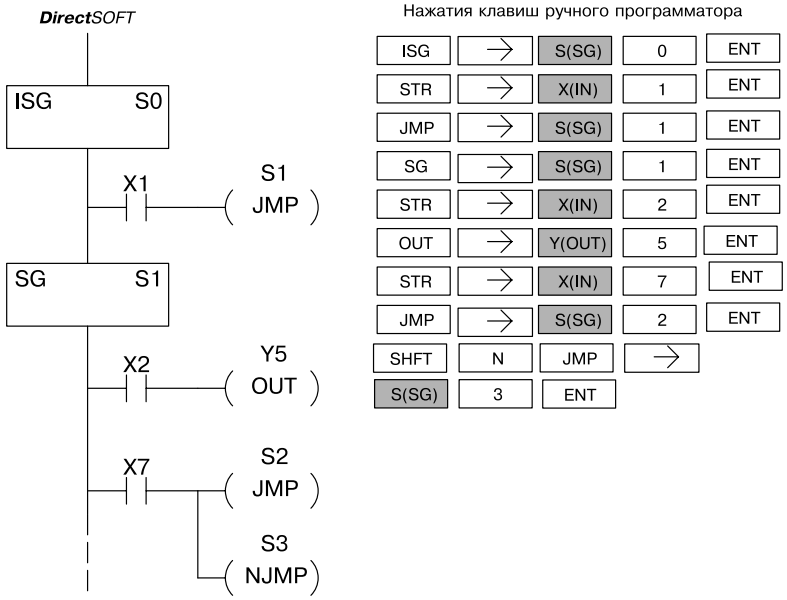
√	√	√
430	440	450

Команда Непереход позволяет программе переходить из активной стадии, в которой она содержится, к другой стадии, на которую эта команда ссылается. Не-переход произойдет при логическом выключении входа. Активная стадия, содержащая команду Не-переход, будет деактивирована спустя одно сканирование после выполнения команды Непереход.



Тип данных операнда		Диапазон DL430	Диапазон DL440	Диапазон DL450
		aaa	aaa	aaa
Стадия	S	0-377	0-777	0-1777

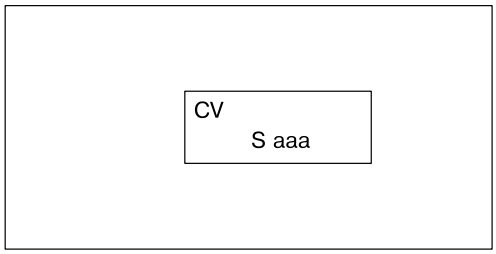
В следующем примере, когда процессор начнет выполнение программы, в активном состоянии будет только стадия ISG 0 . При включении X1 программа переходит из начальной стадии 0 в стадию 1. В стадии 1, если X2 подключен, будет включен выход Y5. Если включен X7, программа переходит из стадии 1 в стадию 2. Если же X7 выключен, программа переходит из стадии 1 в стадию 3.



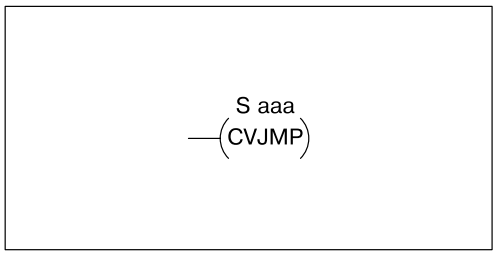
Сходящаяся стадия (CV) и переход схождения (CVJMP)

X	√	√
430	440	450

Команда Сходящаяся стадия используется для группирования определенных стадий, объявляя их сходящимися стадиями.



Когда все сходящиеся стадии в группе станут активными, будет выполнена команда CVJMP (а также любая дополнительная логика в последней CV-стадии). Все предшествующие CV-стадии должны стать активными, прежде чем будет выполнена логика последней CV-стадии. Все сходящиеся стадии дезактивируются спустя одно сканирование после выполнения команды CVJMP.

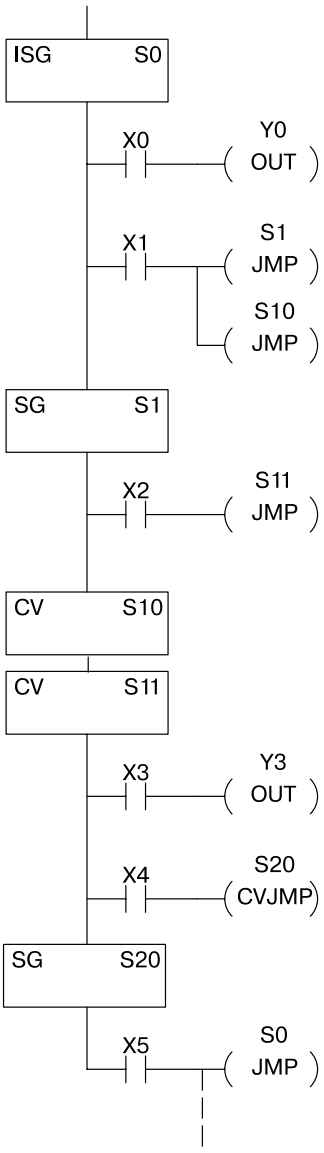


Команды дополнительной логики разрешены только после последней команды Сходящаяся стадия и до команды CVJMP. Разрешены также многократные команды CVJMP.

Сходящиеся стадии должны быть запрограммированы в основном теле прикладной программы. Это означает, что их нельзя помещать в подпрограммах или обработчиках прерываний.

Тип данных операнда		Диапазон DL440
		aaa
Стадия	S	0-777

В следующем примере при активизации обеих сходящихся стадий S10 и S11 будет выполнена команда CVJMP, если включен X4. Команда CVJMP дезактивирует S10 и S11 и активизирует S20. Затем, если включен X5, программа переходит назад к начальной стадии S0.



Нажатия клавиш ручного программатора

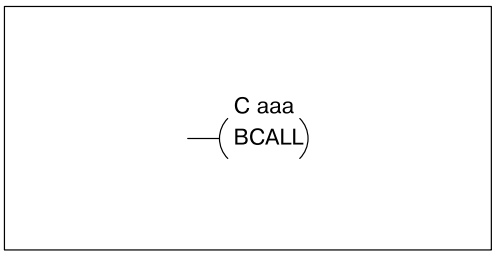
ISG	→	S(SG)	0	ENT				
STR	→	X(IN)	0	ENT				
OUT	→	Y(OUT)	0	ENT				
STR	→	X(IN)	1	ENT				
JMP	→	S(SG)	1	ENT				
JMP	→	S(SG)	1	0	ENT			
SG	→	S(SG)	1	ENT				
STR	→	X(IN)	2	ENT				
JMP	→	S(SG)	1	1	ENT			
SHFT	C	V	→	S(SG)	1	0	ENT	
SHFT	C	V	→	S(SG)	1	1	ENT	
STR	→	X(IN)	3	ENT				
OUT	→	Y(OUT)	3	ENT				
STR	→	X(IN)	4	ENT				
SHFT	C	V	SHFT	JMP	S(SG)	2	0	ENT
SG	→	S(SG)	2	0	ENT			
STR	→	X(IN)	5	ENT				
JMP	→	S(SG)	0	ENT				

Команды блока стадий используются для активизации блока стадий. Команды Вызов блока, Блок и Конец блока должны использоваться совместно.

Вызов блока (BCALL)

X	√	√
430	440	450

Команда BCALL используется для активизации блока стадий. Существует несколько вещей, которые необходимо знать о команде BCALL.



Использует номера CR. Команда BCALL представляется в виде выходной обмотки, но фактически не ссылается на номер стадии, как можно было бы подумать. Вместо этого блок отождествляется с управляющим реле (Caaa). Такое управляющее реле не может быть использовано в качестве выхода в другом месте программы.

Должна оставаться активной — команда BCALL фактически управляет всеми стадиями между командами BLK и BEND, даже после того, как началось выполнение стадий внутри блока. Команда BCALL должна оставаться *активной*, в противном случае все стадии в блоке будут автоматически дезактивированы. *Если либо команда BCALL, либо стадия, содержащая команду BCALL, будут дезактивированы, то автоматически будут дезактивированы все стадии в соответствующем блоке.*

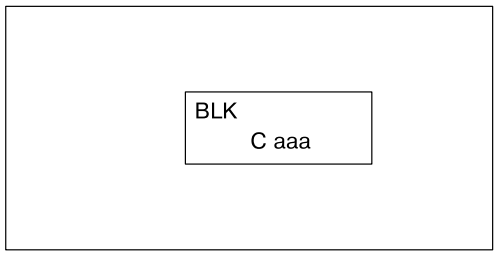
Активизирует первую стадию блока. Выполнение команды BCALL автоматически активизирует первую стадию, следующую за командой BLK.

Тип данных операнда	Диапазон DL440
	aaa
Управляющее реле C	0-1777

Блок (BLK)

X	√	√
430	440	450

Команда Блок — это метка, помечающая начало блока стадий, которые могут быть активизированы как группа. Непосредственно за начальной командой Блока должна следовать команда Стадия. В блоке запрещены команды Начальная стадия. Управляющее реле (Caaa), указанное в команде Блок, не должно использоваться в качестве выхода в другом месте программы.

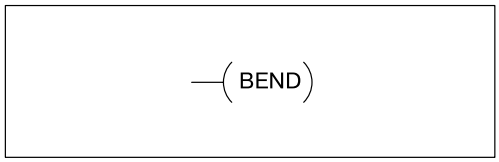


Тип данных операнда	Диапазон DL440
	aaa
Управляющее реле C	0-1777

Конец блока (BEND)

X	√	√
430	440	450

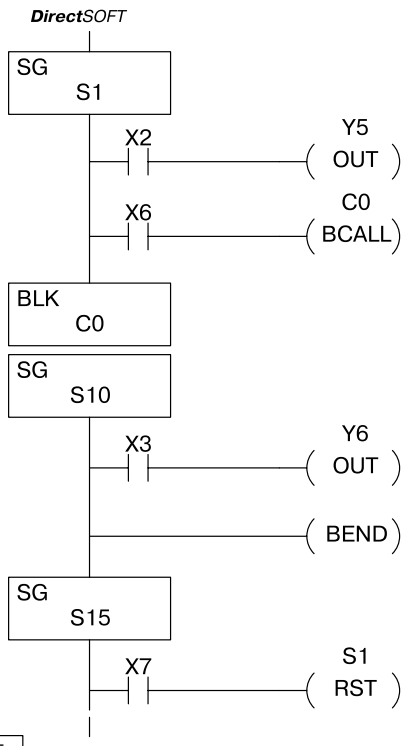
Команда Конец блока — это метка, используемая с командой Блок. Она помечает конец блока стадий. У нее нет операнда. Для команды Вызов блока разрешена только одна команда BEND .



В данном примере Вызов блока выполняется, когда стадия 1 активна, и X6 включен. Вызов блока в данном случае автоматически активизирует стадию S10, которая следует непосредственно за командой Блок.

Это позволяет стадиям между S10 и командой Конец блока выполняться, как было запрограммировано. Если команда BCALL выключается, либо выключается содержащая ее стадия, то автоматически выключаются все стадии между командами BLK и BEND.

При внимательном рассмотрении S15 заметно, что X7 мог бы сбросить стадию S1, которая в этом случае отключила бы команду BCALL, сбрасывая тем самым все стадии внутри блока.

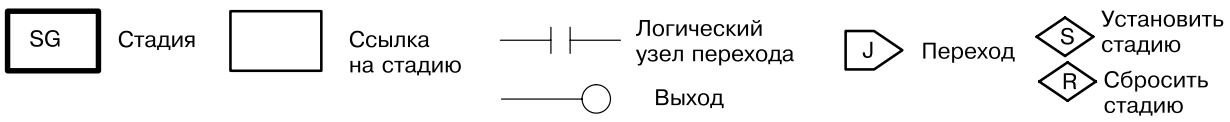


Нажатия клавиш ручного программатора

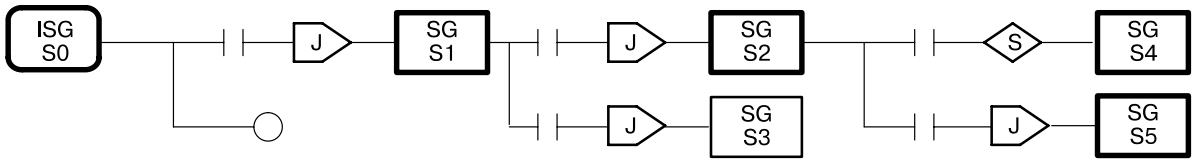
SG	→	S(SG)	1	ENT					
STR	→	X(IN)	2	ENT					
OUT	→	Y(OUT)	5	ENT					
STR	→	X(IN)	6	ENT					
SHFT	B	C	A	L	L	→	C(CR)	0	ENT
SHFT	B	L	K	→	C(CR)	0	ENT		
SG	→	S(SG)	1	0	ENT				
STR	→	X(IN)	3	ENT					
OUT	→	Y(OUT)	6	ENT					
SHFT	B	E	N	D	ENT				
SG	→	S(SG)	1	5	ENT				
STR	→	X(IN)	7	ENT					
RST	→	S(SG)	1	ENT					

Просмотр стадий в DirectSOFT

Опция Просмотр стадий в **Direct**SOFT позволяет просмотреть программу в виде блок-схемы. На следующем рисунке показаны соглашения о символах, используемые в диаграммах. Возможно, Просмотр стадий окажется полезным в качестве средства проверки того, что ваша стадийная программа правильно воспроизвела реализуемую вами логику диаграммы переходов состояния.



На следующем рисунке показано, как будет выглядеть типичная релейная программа, содержащая стадии. Обратите внимание на направление рисунка: слева - направо.



Вопросы и ответы по стадийному программированию

В качестве помощи начинающим изучение стадийного программирования мы включаем следующие вопросы, задаваемые чаще всего. Более подробно предметы всех вопросов рассмотрены выше в данной главе.

В. Что именно может сделать стадийное программирование, чего я не могу добиться с помощью обычных RLL-программ?

О. Стадии позволяют определить все состояния вашего процесса прежде, чем начнется программирование. Такой подход является более организованным, поскольку программа разбивается на разделы. В качестве стадий эти разделы программы активны только когда они действительно нужны процессу. Большинство процессов могут быть организованы в виде последовательности стадий, соединенных переходами на основе событий.

В. Разве стадия на самом деле не аналогична подпрограмме?

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

В. Что такое Биты стадии?

О. Бит стадии — это единственный бит в регистре отображения процессора, представляющий в реальном времени активный/неактивный статус стадии. Например, пусть «S0» — это ссылка на бит стадии 0. Если S0=0, то при каждом сканировании процессора цепи стадии 0 пропускаются (не выполняются). Если S0=1, то цепи стадии 0 выполняются при каждом сканировании процессора. Биты стадии, используемые в качестве связей, позволяют одной части программы наблюдать за другой, определяя активный/неактивный статус стадии.

В. Как стадия становится активной?

О. Существует три способа:

- Если стадия является начальной (ISG), она автоматически активизируется при включении питания.
- Другая стадия может выполнить команду JMP со ссылкой на данную стадию, которая сделает данную стадию активной при следующем сканировании программы.
- Цепь программы может выполнить команду Установка бита стадии (например, SET S0).

В. Как стадия становится неактивной?

О. Существует три способа:

- Стандартные стадии (SG) автоматически неактивны при включении питания.
- Стадия может выполнить команду JMP, сбросив при этом свой бит стадии в 0.
- Любая цепь программы может выполнить команду Сброс бита стадии (например, RST S0).

В. А что такое метод тока питания для стадийных переходов?

О. Метод тока питания для соединения соседних стадий (расположенных непосредственно выше или ниже в программе) в действительности совпадает с командой Стадийный переход, выполненной в стадии, расположенной выше со ссылкой на нижерасположенную стадию. Редактирование в **Direct**SOFT переходов тока питания более сложно, мы рассматриваем их отдельно от двух предыдущих вопросов.

В. Могу ли я создать стадию, активную только в течение одного сканирования?

О. Да, но это не лучший способ использования стадии. Вместо этого воспользуйтесь цепью программы, активной в одном проходе, вставив в конце цепи команду Stage JMP. Тогда программа выполнит цепь при последнем сканировании до того, как ее стадия совершит переход к новой стадии.

В. Разве Stage JMP не аналогична обычной команде GOTO, используемой в программном обеспечении?

О. Нет, имеются существенные различия. Команда GOTO переводит выполнение программы непосредственно к месторасположению кода, на которое ссылается GOTO. Stage JMP просто сбрасывает бит стадии для текущей стадии, одновременно устанавливая бит стадии, на которую ссылается команда JMP. Биты стадий, равные 0 или 1, определяют неактивный/активный статус соответствующих стадий. **Stage** JMP приводит к следующим результатам:

- Когда выполняется JMP, остальные цепи текущей стадии все равно выполняются, даже если они располагаются после (ниже) команды JMP. При следующем проходе эта стадия не выполняется, поскольку она уже не активна.
- Стадия, указанная в команде Stage JMP, будет выполняться при ее следующем появлении. Если она расположена после (ниже) текущей стадии, она будет выполнена в том же сканировании. Если она расположена до (выше) текущей стадии, она будет выполнена при следующем сканировании.

В. Как определить, где использовать Stage JMP, а где Установку бита стадии или Сброс бита стадии?

О. Эти команды используются согласно топологии разработанной диаграммы состояния:

- Используйте команду **Stage** JMP для смены состояний… двигаясь от одного состояния к другому.
- Используйте команду Установка бита стадии, когда текущее состояние порождает новое параллельное состояние или последовательность стадий, либо когда состояние-диспетчер своей командой запускает последовательность состояний.
- Используйте команду Сброс бита стадии, когда текущее состояние является последним в последовательности состояний, и его задача выполнена, либо когда состояние-диспетчер своей командой завершает последовательность состояний.

В. Что такое начальная стадия и когда ее использовать?

О. Начальная стадия автоматически активизируется при подаче питания. Впоследствии она работает подобно любой другой стадии. Если потребуется, можно использовать несколько начальных стадий. Используйте начальную стадию либо для цепи программы, которая должна всегда быть активной, либо в качестве стартовой точки.

В. Могу ли я разместить цепи программы вне стадий, так чтобы они всегда были активны?

О. Можете, но это плохая практика разработки программного обеспечения. Поместите релейную логику, которая всегда должна быть активна, в начальную стадию, не допускайте сброса этой стадии и не используйте внутри нее команду **Stage** JMP. Эта стадия может в нужный момент запускать другие последовательности стадий, устанавливая соответствующий(е) бит(ы) стадии.

В. Могу ли я использовать одновременно несколько активных стадий?

О. Да, и это обычная практика для многих программ. Однако важно организовать приложение в виде отдельных процессов, каждый из которых составлен из стадий. При этом хорошо сконструированный процесс будет в основном последовательным, с одной одновременно активной стадией. Однако все процессы программы могут быть активны одновременно.

Работа контуров ПИД-регулирования

8

(только DL450)

В этой главе……

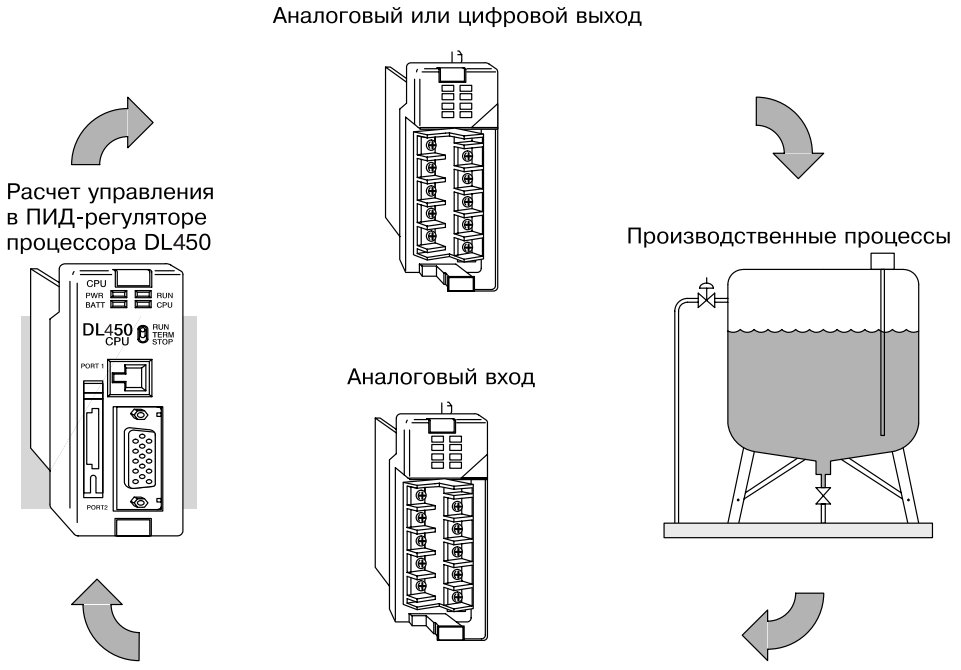
Характеристики контуров ПИД-регулирования в DL450

Основные возможности

Контур управления процессом в DL450 предоставляет большой набор возможностей, удовлетворяя потребности множества приложений. Среди основных возможностей:

- До четырех контуров с независимым программированием частоты опроса
- Возможность ручного/автоматического/каскадного управления контуром
- Наличие двух типов безударных переходов
- Полнофункциональные аварийные сигналы
- Генератор профилей программного изменения задания

Помимо выполнения релейной программы процессор DL450 поддерживает работу контуров управления процессом. Можно выбрать и сконфигурировать до четырех контуров. Все датчики и привода, как показано ниже, подключаются к стандартным модулям ввода/вывода DL405. Все переменные процесса, значения коэффициентов усиления, уровни аварийных сигналов и т.п., связанные с каждым контуром, хранятся в таблице переменных контура процессора. При каждом сканировании процессор DL450 считывает входы переменной процесса (process variable, PV). Затем при каждом сканировании процессор выполняет часть расчета реакции контура в течение выделенного временного интервала, обновляя значение управляющего выхода. Контуры управления используют для расчета управляющего сигнала ПИД-алгоритм регулирования (пропорционально-дифференциально-интегральный). В данной главе описывается работа контуров и действия по их конфигурированию и настройке.



Лучшим средством конфигурирования контуров DL450 является пакет **Direct**SOFT, версия 2.1 или более поздняя. **Direct**SOFT использует диалоговые окна для создания редактора форм, позволяющих настраивать контуры. После завершения конфигурирования можно использовать в **Direct**SOFT для настройки каждого контура окно Просмотр тренда ПИД-регулятора (PID Trend View). Выбранные параметры конфигурирования и настройки хранятся во флэш-памяти DL450, не стираясь при выключении питания. Параметры настройки контура могут также сохраняться для повторного использования на диске.

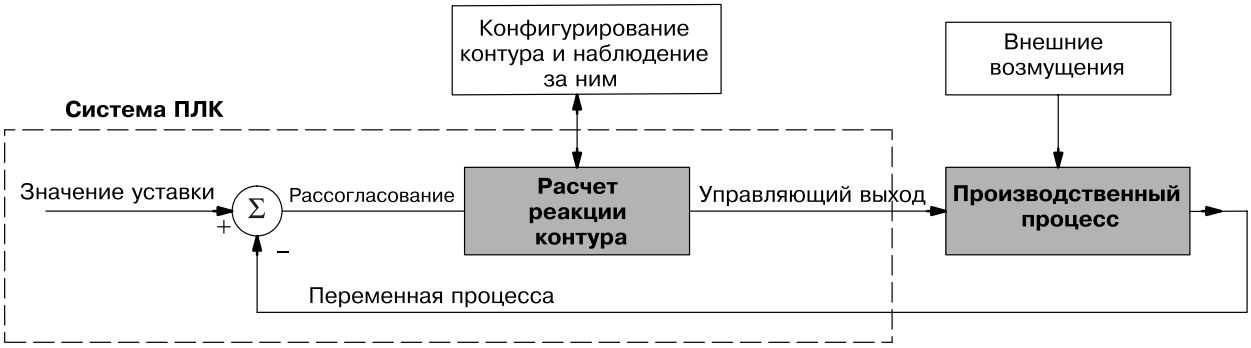
Свойства контура с ПИД-регулятором	Спецификации
Количество контуров	Настраивается, максимум 16
Требуемая V-память процессора	32 слова (V ячейки) на выбранный контур, 64 слова при использовании пилообразного сигнала
Алгоритм ПИД-регулятора	Уравнения ПИД-регулирования в позиционной форме или в форме скорости
Направление управляющего выхода	Прямое или обратное действие
Зависимость рассогласования	Выбирается линейная зависимость, квадратный корень из ошибки, ошибка в квадрате
Скорость обновления контура (время между расчетами управляющего воздействия)	0.05 - 99.99 секунды, программируется пользователем
Минимальная скорость обновления контура	0.05 с для 1-4 контуров, 0.1 с для 5-8 контуров, 0.2 с для 9-16 контуров
Режимы контуров	Автоматический, Ручной (управление оператором) или Каскадный
Профиль программного задатчика	До 8 шагов наклон/выдержка (16 участков) на контур с индикацией номера шага
Обработка переменной процесса (PV)	Выбирается, стандартная линейная или с извлечением квадратного корня (для входа измерения расхода)
Границы уставок (задания)	Минимальное и максимальное значения уставок
Границы переменной процесса	Минимальное и максимальное значения переменной процесса
Коэффициент пропорционального усиления	Коэффициент усиления от 0.00 до 99.99
Интегратор (Reset)	Время сброса от 0.0 до 999.8 в секундах или минутах
Производная (Rate)	Время дифференцирования от 0.00 до 99.99 секунды
Границы производной	Коэффициент дифференциального усиления в границах от 1 до 20
Безударный переход управления I	Смещение и уставка, автоматически инициализируются при переключении управления с ручного на автоматическое
Безударный переход управления II	Автоматически устанавливает смещение равным управляющему выходу при переключении управления с ручного на автоматическое
Пошаговое смещение	Дополнительная настройка смещения рабочей точки при больших изменениях уставок
Анти-выбег	Для позиционной формы ПИД-уравнения, подавляет работу интегратора, когда управляющий выход достигает 0% или 100 % (ускоряет возвращение контура к исходному режиму, когда выход выходит из насыщения)
Зона нечуствительности рассогласования	Задайте допуск (плюс/минус) на составляющую отклонения (SP - PV), при котором отсутствует изменение значения управляющего выхода.

Характеристика аварийного сигнала	Спецификации
Зона нечувствительности	От 0.1% до 5% для всех аварийных сигналов
Точки аварийных сигналов PV	Уровни уставок аварийных сигналов PV: Самый низкий, Низкий, Высокий и Самый высокий
Рассогласование PV	Выход рассогласования PV за значения двух предельных значений
Скорость изменения	Позволяет обнаружить превышение заданного предела скорости изменения PV

Знакомство с контурами, реализующими ПИД-регулирование

Чтобы разобраться с ключевыми элементами контура управления, взгляните на приведенную ниже блок-схему. Замкнутый путь, обходящий диаграмму, — это «контур», используемый в «управлении в замкнутом контуре».

Производственный процесс — это набор действий, преобразующих сырье в готовую продукцию. Процесс может включать физическую и/или химическую обработку материала. Изменения делают материал более полезным для конкретного применения, в конечном счете, делая его итоговым продуктом.



Переменная процесса — измерение некоторого физического свойства сырья. Измерения выполняются с помощью датчиков. Например, если в производственном процессе используется печь, нас может очень интересовать управление температурой. Следовательно, температура является переменной процесса.

Значение уставки — теоретически идеальное значение переменной процесса или заданное значение, обеспечивающее наилучший продукт. Оператор машины знает это значение и либо задает его вручную, либо программирует в контроллере для последующего автоматического задания.

Внешние возмущения — непредсказуемые источники ошибок, последствия которых система управления стремится устранить, компенсируя их. Например, если подача топлива является постоянной, печь будет горячее в теплую погоду, чем в холодную. Система управления печью должна противодействовать этому эффекту, чтобы поддержать постоянную температуру в печи в течение любого сезона. Таким образом, погода (которая не слишком предсказуема) является одним из источников возмущений для данного процесса.

Рассогласование — алгебраическая разность между переменной процесса и уставкой. Это ошибка контура управления, равная нулю, когда переменная процесса равна значению уставки (желаемому значению). Правильно работающий контур управления умеет обеспечивать малое значение рассогласования.

Расчет реакции контура — применение в реальном времени к сигналу рассогласования математического алгоритма, генерирующего команду управляющего выхода для минимизации величины рассогласования. Возможны различные алгоритмы управления, DL450 использует алгоритм ПИД-регуляторования (пропорционально-интегрально-дифференциальный), подробно описанный ниже.

Управляющий выход — результат расчета контура, который становится командой управления для процесса (например уровень нагревателя в печи).

Конфигурирование контура — введение оператором уставок и настроек, которые задают и оптимизируют работу контура управления. Функция расчета контура в реальном времени использует конфигурационные параметры для настройки коэффициента усиления, компенсаций, и т.п.

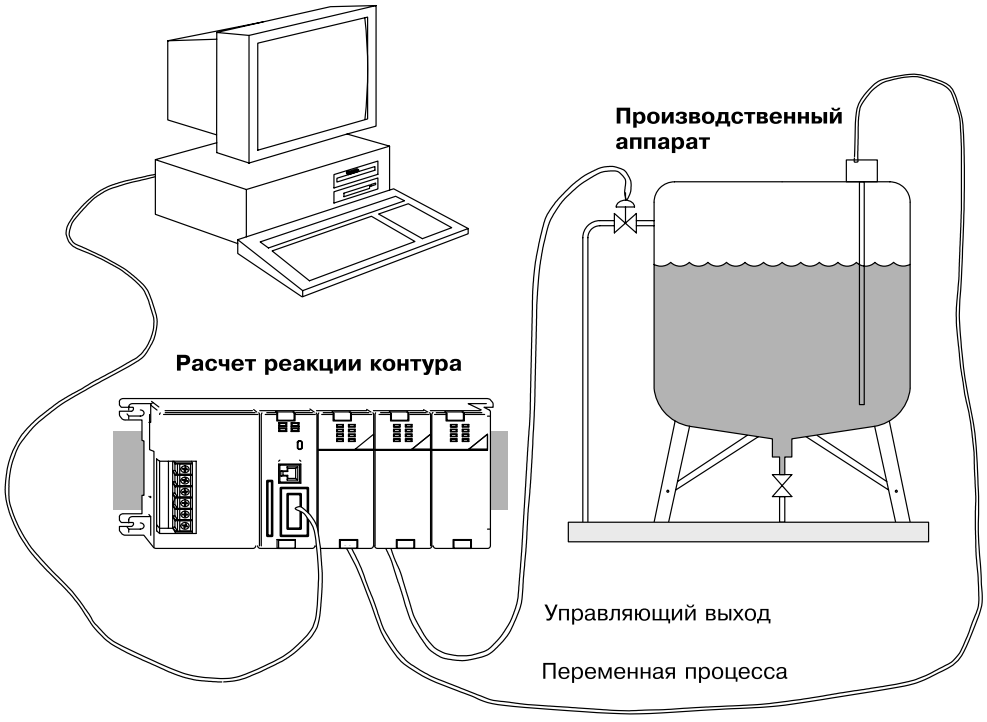
Наблюдение за контуром — функция, позволяющая оператору наблюдать состояние и работу контура управления. Она используется вместе с конфигурированием контура для оптимизации контура (минимизации составляющей отклонения).

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

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

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

Конфигурирование контура и наблюдение за ним



На показанном персональном компьютере работает **Direct**SOFT, пакет для программирования контроллеров **Direct**LOGIC. **Direct**SOFT версии 2.0 или более поздней может программировать процессор DL450. Программное обеспечение включает редактор для конфигурирования параметров контура. Самое главное, оно включает экран тренда контура с ПИД-регулятором, который незаменим при настройке контура. Подробно использование этого программного обеспечения описано в Справочном руководстве по **Direct**SOFT.

Параметры настройки контура

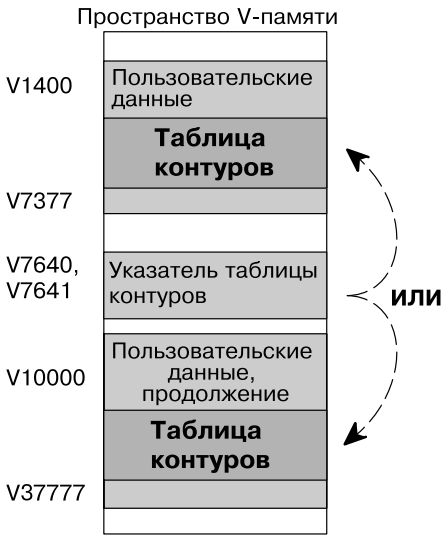
Таблица контуров и количество контуров

Процессор DL450 получает команды для контура ПИД-регулирования только из таблиц в V-памяти. Это означает, что в RLL отсутствует тип команды « ПИД-регулирование». Вместо этого процессор читает установленные параметры из зарезервированных ячеек V-памяти. Как показано в приведенной таблице, необходимо задать значение в V7640, указывая начальный адрес основной таблицы контура. Затем может понадобиться указать в V7641 количество контуров, с которым будет работать процессор. V7642 содержит флаги ошибок, устанавливаемые при неправильном задании V7640 или V7641.

Адрес	Параметр настройки	Тип данных	Пределы	Чтение/запись
V7640	Указатель таблицы параметров контура	Восьмеричные	V1400 – V7340, V10000 – V37740	Запись
V7641	Количество контуров	BCD	0 –16	Запись
V7642	Флаги отклонений контура	Двоичные	0 или 1	Чтение

Если количество контуров равно «0», то во время выполнения релейной программы задача расчета контуров отключена. Контроллер контуров позволяет использовать контуры в порядке возрастания, начиная с 1. Например, нельзя использовать контуры 1 и 4, пропуская 2 и 3. Контроллер контура пытается управлять всеми контурами, заданными в V7641.

Таблица параметров контуров может занимать блок памяти в нижней области данных пользователя (V1400 — V7340) или в верхней области данных пользователя (V10000 — V37777), как показано справа. Убедитесь, что выбранного вами адресного пространства достаточно для приложения. Значение в V7641 сообщает процессору размер таблицы, так как для каждого контура используется 32 ячейки.

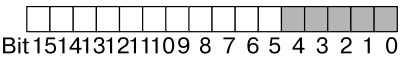


Диалоговое окно **Direct**SOFT Настройка ПИД-регулятора (PID Setup) — это способ задать данные параметры. Можно также использовать команды программы, например, LDA или LD, и команды OUT. Однако эти ячейки памяти являются частью сохраняющихся системных параметров, поэтому запись их из программы не обязательна.

Флаги ошибок ПИД-регулятора

Процессор сообщает обо всех ошибках программирования параметров установки в V7640 и V7641, устанавливая соответствующие биты в V7642 при переходе в режим выполнения программы.

Флаги ошибок ПИД-регулятора, V7642



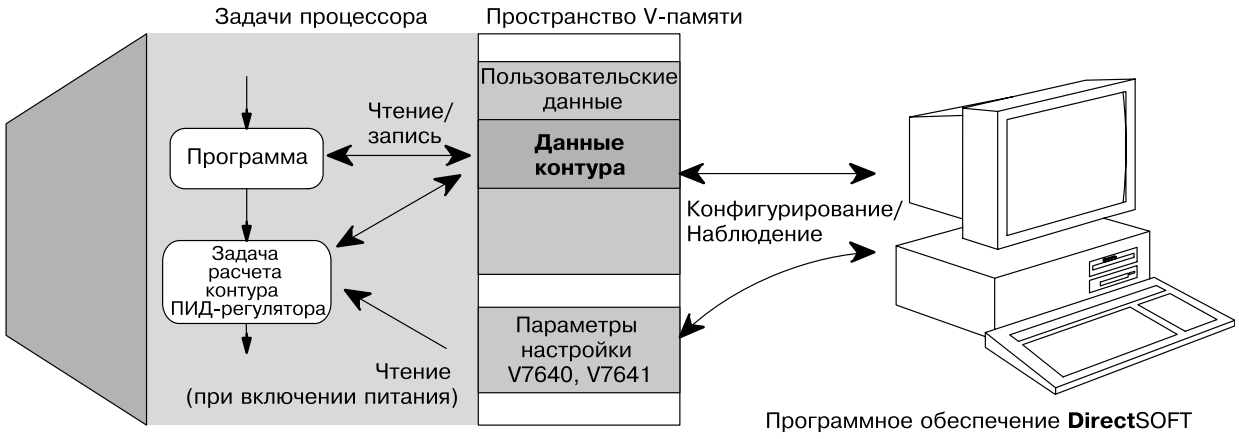
При использовании диалогового окна настройки контура **Direct**SOFT, встроенная автоматическая проверка диапазонов препятствует возможным ошибкам настройки. Однако параметры настройки могут быть записаны другими способами, например, из RLL, в подобных случаях регистр флагов ошибки может оказаться полезен.
В следующей таблице перечислены ошибки, о которых сообщается в V7642.

Бит	Описание ошибки (0 = нет ошибки, 1 = ошибка)
0	Начальный адрес (в V7640) вне пределов нижней области V-памяти.
1	Начальный адрес (в V7640) вне пределов верхней области V-памяти.
2	Выбранное количество контуров (в V7641) больше 16
3	Таблица контуров вышла за пределы V7377. Используйте адрес поближе к V1400
4	Таблица контуров вышла за пределы V37777. Используйте адрес поближе к V10000

Для быстрой проверки достаточно, если процессор находится в рабочем режиме и V7642=0000, это значит ошибки программирования отсутствуют.

Задание размера и местонахождения таблицы контуров

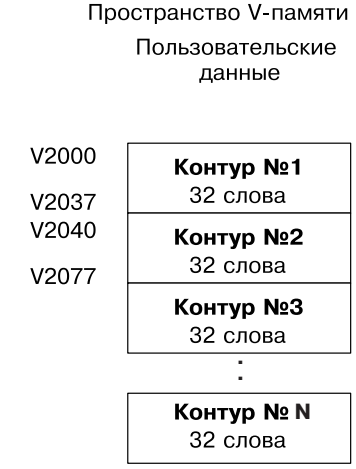
При переходе в режим выполнения процессор считывает параметры настройки контура, как показано на приведенном ниже рисунке. В этот момент процессор узнает местонахождение таблицы контуров и количество контуров. Затем, при выполнении релейной программы, задача расчета контура ПИД-регулятора использует данные контура для выполнения расчета, генерации аварийных



сигналов и т.п. Некоторые параметры таблицы контуров процессор будет считывать или записывать при каждом расчете контура.

Таблица параметров контуров содержит данные только для того количества контуров, которое было задано в V7641. Каждый сконфигурированный контур занимает 32 слова (с 0 по 37 восьмеричное) в таблице контуров.

Например, пусть наше приложение работает с 4 контурами. Выберем произвольно V2000 в качестве начальной ячейки. Параметры контуров займут ячейки V2000 — V2037 для контура 1, V2040 — V2077 для контура 2 и т.д. Контуру 4 соответствуют ячейки V2140 — V2177.



Описание слов таблицы контуров

В следующей таблице перечислены параметры, связанные с каждым контуром. Чтобы помочь при определении положения конкретных параметров, приведено восьмеричное смещение адреса. Например, если таблица начинается с V2000, то положение параметр перезапуска (общего) равно Addr+11, или V2011. Не пользуйтесь номером слова для вычисления адреса.

№ слова	Адрес + смещение	Описание	Формат
1	Addr + 0	Настройки режима 1 контура ПИД-регулятора	биты
2	Addr + 1	Настройки режима 2 контура ПИД-регулятора	биты
3	Addr + 2	Значение уставки — задание (SP)	слово/двоичный
4	Addr + 3	Переменная процесса (PV)	слово/двоичный
5	Addr + 4	Значение смещения (интегратора)	слово/двоичный
6	Addr + 5	Значение управляющего выхода	слово/двоичный
7	Addr + 6	Режим работы контура и статус аварийного сигнала	биты
8	Addr + 7	Частота опроса	слово/BCD
9	Addr + 10	Коэффициент усиления (пропорциональный) — Gain	слово/BCD
10	Addr + 11	Время перестановки (интегрирования) — Reset	слово/BCD
11	Addr + 12	Время производной (дифференцирования) — Rate	слово/BCD
12	Addr + 13	PV, аварийный сигнал Самый низкий	слово/двоичный
13	Addr + 14	PV, аварийный сигнал Низкий	слово/двоичный
14	Addr + 15	PV, аварийный сигнал Высокий	слово/двоичный
15	Addr + 16	PV, аварийный сигнал Самый высокий	слово/двоичный
16	Addr + 17	PV, аварийный сигнал рассогласования (YELLOW)	слово/двоичный
17	Addr + 20	PV, аварийный сигнал рассогласования (RED)	слово/двоичный
18	Addr + 21	PV, аварийный сигнал скорости изменений	слово/двоичный
19	Addr + 22	PV, установка гистерезиса аварийного сигнала	слово/двоичный
20	Addr + 23	PV, зона нечувствительности рассогласования	слово/двоичный
21	Addr + 24	Резерв	
22	Addr + 25	Установка признака ограничения коэффициента усиления дифференцирования	слово/BCD
23	Addr + 26	SP, установка предела нижнего значения задания	слово/двоичный
24	Addr + 27	SP, установка предела верхнего значения задания	слово/двоичный
25	Addr + 30	Установка предела нижнего значения управляющего выхода	слово/двоичный
26	Addr + 31	Установка предела верхнего значения управляющего выхода	слово/двоичный
27	Addr + 32	Указатель адреса V-памяти значения удаленного задания — SP	слово/ шестнадцатеричный
28	Addr + 33	Флаг программного задатчика	бит
29	Addr + 34	Начальный адрес таблицы программного задатчика	слово/ шестнадцатеричный
30	Addr + 35	Флаги ошибок таблицы программного задатчика	биты
31		Резерв	
32		Резерв	

Параметры режима 1 ПИД-егулятора. Описание битов (Addr+0)

В следующей таблице перечислены определения отдельных битов слова Параметры режима 1 ПИД-регулятора (Addr+0). Более подробно каждый из них описан в соответствующем разделе данной главы.

Бит	Описание установки режима ПИД-регулятора 1	Чтение/ Запись	Бит=0	Бит=1
0	Запрос работы контура в Ручном режиме	запись	–	0⇒1 запрос
1	Запрос работы контура в Автоматическом режиме	запись	–	0⇒1 запрос
2	Запрос работы 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 позиционный/ скоростной	запись	Позиционный	Скоростной
6	Преобразование PV: линейное/квадратный корень	запись	Линейное	Кв. корень
7	Выбор линейной/квадратичной составляющей отклонения	запись	Линейное	Квадратичная
8	Введение зона нечувствительности ошибки	запись	Нет	Да
9	Выбор предела коэффициента усиления дифференцирования	запись	Выкл	Вкл
10	Выбор замораживания смещения (интегратора)	запись	Выкл	Вкл
11	Режим работы программного задатчика	запись	Выкл	Вкл
12	Аварийного сигнала PV	запись	Выкл	Вкл
13	Аварийного сигнала отклонения PV	запись	Выкл	Вкл
14	Аварийный сигнала скорости изменения PV	запись	Выкл	Вкл
15	Резерв			

Параметры режима 2 ПИД-егулятора. Описание битов (Addr+1)

В следующей таблице перечислены определения отдельных битов слова Параметры режима 2 ПИД-регулятора (Addr+1). Более подробно каждый из них описан в соответствующем разделе данной главы.

Бит	Описание установки режима ПИД-регулятора 2	Чтение/ Запись	Бит= 0	Бит= 1
0	Тип входного сигнала переменной(PV)	запись	Униполярный	Биполярный
1	Формат данных контура	запись	12 bit	15 bit
2	Зарезервирован для будущего использования	–	–	–
3	Ограничение входа задания(SP)	запись	Невозможно	Возможно
4	Выбор единиц интегрального коэффициента усиления (Reset)	запись	секунды	минуты
5-7	Зарезервирован для будущего использования	–	–	–
8	Время сканирования ПИД-регулятора (для внутреннего использования)	чтение	–	–
9–15	Зарезервирован для будущего использования	–	–	–

Слово контроля режима/аварийного канала (Addr+6)

В следующей таблице перечислены определения отдельных битов слова контроля режима/аварийного канала (Addr+6). Подробности можно найти в разделах «Режим контура с ПИД-регулятором» и «Аварийные сигналы».

Бит	Описание битов режима/аварии	Чтение/ Запись	Бит= 0	Бит= 1
0	Индикация Ручного режима	чтение	–	Manual
1	Индикация Автоматического режима	чтение	–	Auto
2	Индикация Каскадного режима	чтение	–	Cascade
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 – 15	Зарезервированы для будущего использования	–	–	–

Флаги таблицы программного задатчика (Addr+33)

В следующей таблице перечислены определения отдельных битов флагов таблицы программного задатчика-Ramp/Soak Profile(Addr+33). Дальнейшие подробности можно найти в разделе «Работа программного задатчика».

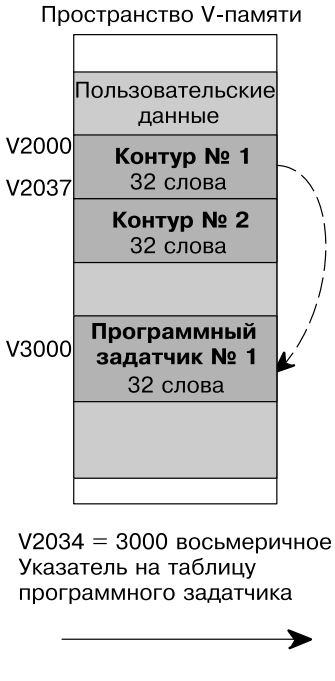
Бит	Описание битов флагов программного задатчика	Чтение/ Запись	Бит=0	Бит=1
0	Запуск программного задатчика	запись	–	0 ⇒1 Запустить
1	Остановка программного задатчика	запись	–	0⇒1 Удержать
2	Возобновление работы программного задатчика	запись	–	0 ⇒1 Продолжить
3	Толчок программного задатчика на шаг	запись	–	0 ⇒1 Толчок
4	Завершение профиля программного задатчика	чтение	–	Завершение
5	Отклонение входа PV от сигнала программного задатчика	чтение	Выкл	Вкл
6	Программный задатчик приостановлен	чтение	Выкл	Вкл
7	Зарезервирован	чтение	–	–
8–15	Текущий шаг профиля программного задатчика	чтение	Декодируется как байт (шестнадцатеричный)	

Биты 8-15 должны считываться как байт, показывающий номер текущего шага в профиле программного задатчика. Этот байт может принимать значения 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F и 10, представляя 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 **Direct**SOFT упрощает этот процесс.

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

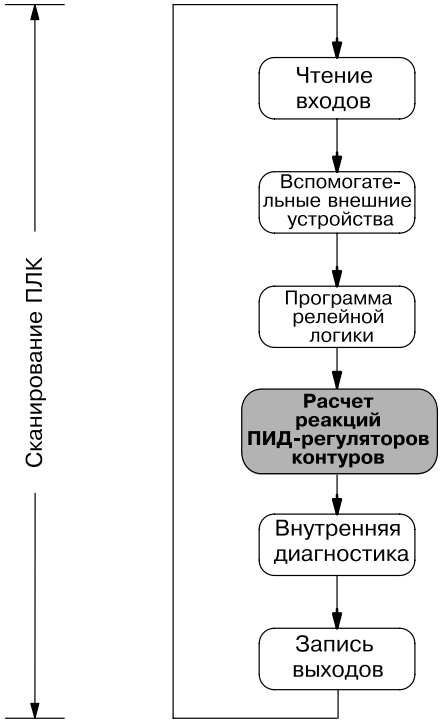
В следующей таблице перечислены определения отдельных битов флагов ошибок программирования таблицы программного задатчика (Addr+35). Дальнейшие подробности можно найти в разделах «Режимы контура с ПИД-регулятором» и «Аварийные сигналы».

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

Частота опроса и планирование контура

Частота опроса контура

Основные задачи процессора делятся на категории, как показано справа. В списке представлены задачи, решаемые процессором в рабочем режиме при каждом сканировании ПЛК. Обратите внимание, что расчеты контуров выполняются после задачи программы. С точки зрения пользователя, контуры могут работать, когда программа не работает.



Частота опроса контура управления — это просто частота расчета реакции контура ПИД-регулирования. Результатом каждого расчета является новое значение управляющего выхода. Для процессора DL450 частоту опроса контура можно задавать в диапазоне от 50 мс до 99.99 секунд. Для большинства контуров расчет реакции ПИД-регулятора выполняется не для каждого сканирования ПЛК. Реально для некоторых контуров расчеты могут понадобиться только раз в 1000 проходов.

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

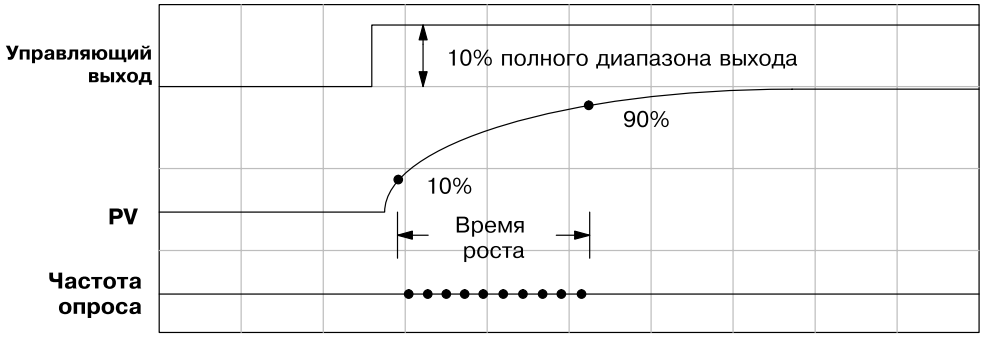
Выбор наилучшей частоты опроса

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

- Нужная частота пропорциональна времени отклика PV на изменение управляющего выхода. Обычно частота опроса сильно инерционного процесса будет мала, а для слабо инерционного процесса нужна большая частота.
- Большие частоты обеспечивают более гладкий управляющий выход и точное значение PV, но используют больше процессорного времени. При частоте, **большей необходимой**, вычислительные ресурсы процессора расходуются вхолостую.
- Меньшие частоты обеспечивают более грубый управляющий выход и менее точное значение PV, но используют меньше процессорного времени.
- Слишком малая частота приводит к нестабильности системы, особенно при изменении уставки или при возникновении возмущений.

Для начала мы можем приравнять частоту опроса любому значению, достаточно большому, чтобы избежать нестабильности управления (что чрезвычайно важно). Для определения начальной частоты воспользуйтесь следующей процедурой:

1. Запустите контур процесса (к этому моменту контур даже необязательно должен быть сконфигурирован). Переведите процессор в рабочий режим (а контур — в ручной режим, если он уже настроен). Вручную установите значение управляющего выхода так, чтобы переменная PV устойчиво находилась в середине безопасного диапазона.
2. Попробуйте выбрать время, когда процесс испытывает незначительные внешние возмущения. Затем вызовите резкое 10% изменение шага управляющего значения.
3. Зарегистрируйте время роста или падения PV (время между точками 10% и 90%).
4. Разделите записанное время роста или падения на 10. Это и будет начальное значение периода опроса, которое можно использовать для начала настройки вашего контура.

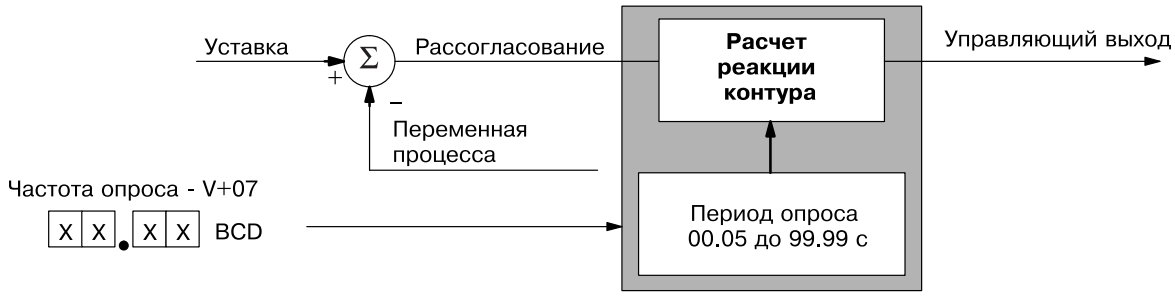


Предположим, что на приведенном выше рисунке измеренное время роста PV равно 25 секундам. Получаемая по этому измерению частота соответствует 2.5 секунды. Для иллюстрации на оси частоты в течение периода роста показано десять опросов. Они показывают частоту расчетов реакции ПИД-регулятора как функцию PV. Конечно, в ходе работы вычисления частоты и ПИД-регулятора проводятся постоянно.

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

Задание частоты опроса

Таблица параметров контуров содержит ячейки данных для частоты каждого контура. В соответствии с приведенным ниже рисунком ячейка V+07 содержит BCD-число от 00.05 до 99.99 (с подразумевающейся десятичной точкой). Оно соответствует значениям от 50 мс до 99.99 секунд. Это число можно запрограммировать с помощью экрана Настройка ПИД-регулятора (PID Setup) **Direct**SOFT или любым другим способом записи в V-память. Для правильной работы контура это число должно быть задано.



Влияние контура ПИД-регулирования на время сканирования процесса

Так как расчеты контуров ПИД-регулирования представляют собой задачи, выполняемые процессором, использование контуров ПИД-регулирования увеличивает среднее время сканирования. Увеличение времени сканирования пропорционально количеству используемых контуров и частоте отсчета каждого контура.

Время выполнения одного расчета контура зависит от количества выбранных опций, например, аварийные сигналы, квадратичная ошибка и т.п. Приведенная справа диаграмма содержит диапазон ожидаемых значений.

Время расчета ПИД-регулятора

Минимум	150 мкс
Типично	250 мкс
Максимум	350 мкс

Для вычисления увеличения времени сканирования мы также должны узнать (или оценить) время сканирования программы (без контуров), так как время быстрого прохода вырастет на меньшую долю, чем время медленного прохода при добавлении в каждом случае одинаковых затрат на расчет реакции ПИД-регулятора. Вот как выглядит формула вычисления среднего времени прохода:

Среднее время прохода с ПИД-контуром = (время прохода без контура)/(период отсчетов контура) x (время расчета реакции ПИД-регулятора) + время прохода без контура

Например, пусть оценочное время сканирования без расчета контуров составляет 50 мс, а период опроса контура — 3 с. Вычислим новое время прохода:

Среднее время прохода с ПИД-контуром = (50 мс)/(3 с) x (450 мс) + 50 мс = 50.004 мс

Как показывают вычисления, добавление только одного контура с небольшой частотой мало влияет на время прохода. Расширим приведенное уравнение, чтобы показать влияние добавления любого числа контуров:

Среднее время прохода с ПИД-контуром = S(время прохода без контуров)/(период отсчетов контура) x (время расчета реакции ПИД-регулятора)+ время прохода без контуров

В приведенном новом уравнении мы должны вычислить суммируемый член (в скобках) для всех контуров от 1 до L (последний контур) и один раз в конце добавить крайний правый член «время прохода без контуров». Предположим, что DL450 управляет четырьмя контурами. В таблице приведены значения данных и суммируемых членов для каждого контура.

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

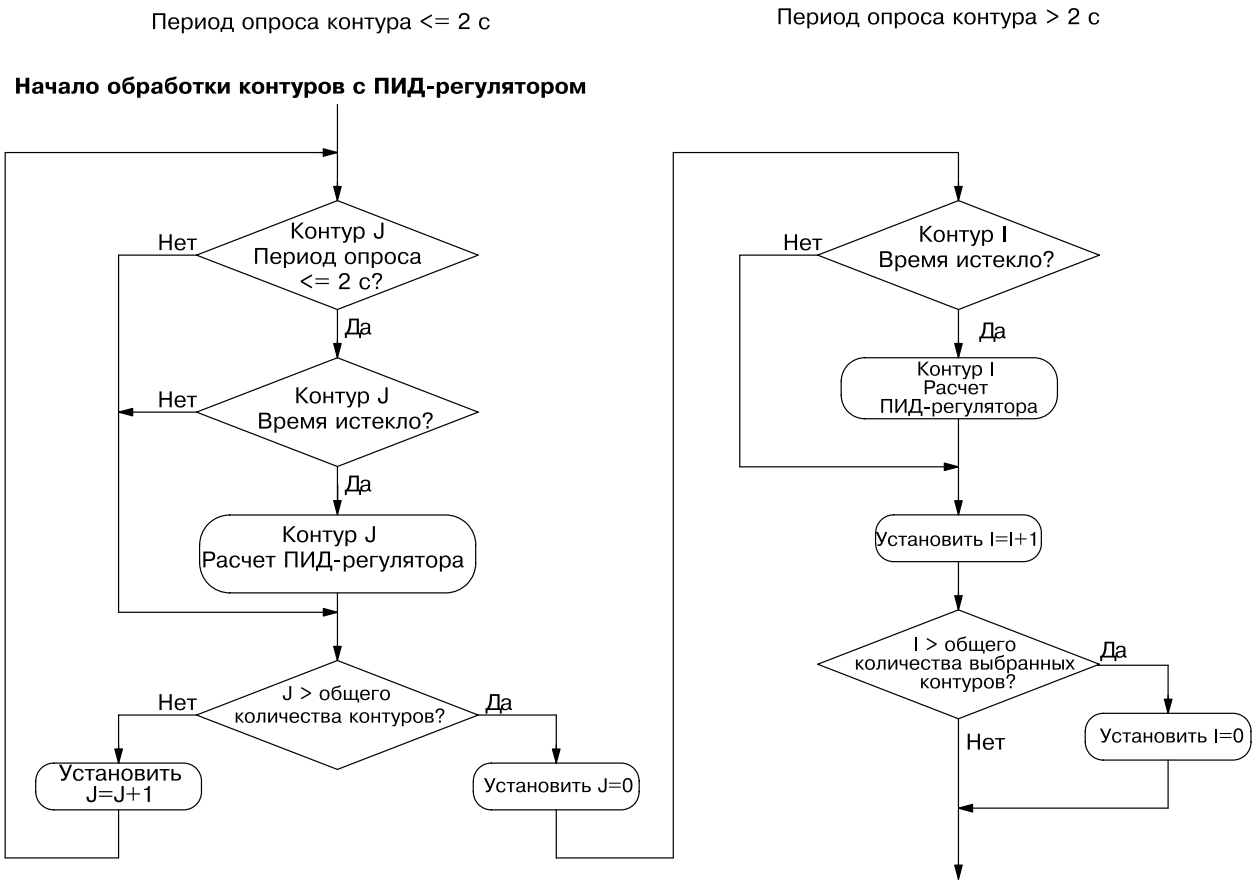
Теперь, складывая суммируемые члены и первоначальное значение времени сканирования, получаем:

Среднее время прохода с ПИД-контуром = (50 мкс + 0.42 мкс + 1.25 мкс + 8.3 мкс) + 50 мс = 50.06 мс

Процессор DL450 выполняет расчет реакции ПИД-регулятора на конкретном сканировании для контура (контуров), период опроса которых требует выполнения обновления (расчета). Встроенный планировщик контура использует следующие правила:

- За одно сканирование обрабатывается столько контуров с периодами опроса <= 2 секунд, сколько нужно для обеспечения частоты опроса каждого контура. Задание контуров с большими частотами увеличит время прохода ПЛК. Следовательно, используйте эту возможность только при необходимости!
- Контуры с периодами опроса > 2 секунд обрабатываются с частотой один или менее контуров за проход, (с минимальной частотой, требуемой для поддержания частоты опроса каждого контура).

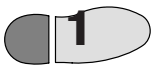
Планирование расчета контура показано на приведенной ниже схеме. Это более подробное представление содержимого задачи «Рассчитать контуры с ПИД-регулятором» схемы сканирования процессора. Указатели «I» и «J» соответствуют медленному (> 2 с) и быстрому (<= 2 с) контурам. Схема позволяет указателю J инкрементироваться от 1 до номера последнего контура, если определены быстрые контуры. Указатель I инкрементируется только один раз за сканирование, а затем только при обновлении следующего медленного контура. Таким образом, оба указателя циклов I и J меняются от 1 до максимального номера используемого контура, но с разными скоростями. Их совместное изменение обеспечивает правильное обновление всех контуров.



Десять шагов к успешному управлению процессом

Современные электронные контроллеры, подобные процессору DL450, обеспечивают функции управления сложными процессами. Автоматические системы управления могут с трудом поддаваться отладке, так как у конкретного симптома может быть множество различных причин. При оперативном введении новых контуров управления рекомендуется использовать осторожный, пошаговый подход:

Шаг 1. Знание рецепта



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

- Определить все существенные переменные процесса, например, температуру, давление, расход и т.п., требующие точного управления.
- Спланировать значения уставок для всех переменных процесса.

Шаг 2. Планирование стратегии управления контуром

Этот шаг подразумевает выбор метода, который будет использоваться машиной, 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 ходе производства и многое другое. Необходимо также определить, как будут генерироваться в ходе процесса значения уставок, и может ли оператор машины вручную изменять эти значения.

Шаг 3. Определение параметров компонентов контура

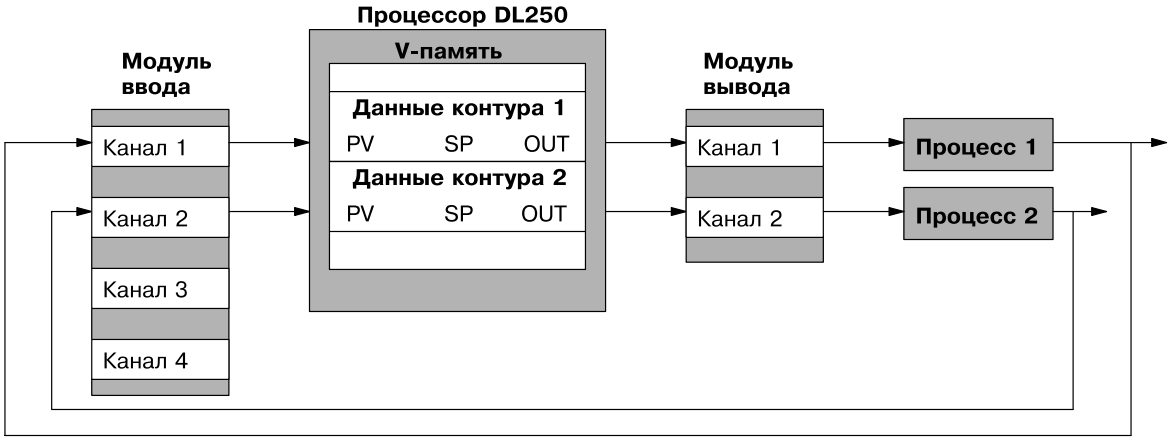
После выбора стратегии управления ключевое действие состоит в том, чтобы правильно определить размеры исполняющих механизмов и чувствительность датчиков.

- Выберите исполнительный механизм (нагреватель, насос и т.п.), соответствующий параметрам нагрузки. Исполнительный механизм с завышенными характеристиками после изменения задания (SP) будет оказывать на процесс слишком сильное воздействие. Однако заниженные характеристики исполнительного механизма при изменения SP или возмущении процесса приведут к отставанию или возможному отклонению переменной (PV) от SP.
- Выберите для вашего процесса датчик PV, соответствующий нужному диапазону (и управлению). Определите требуемую для PV разрешающую способность управляющего воздействия (например, в пределах 2°C) и убедитесь, что разрешающая способность датчика (на уровне младшего значащего бита) не менее чем в 5 раз лучше этого значения. Однако слишком чувствительный датчик может привести к возникновению осцилляций управления и т.п. DL450 обеспечивает 12-битовый и 15-битовый, униполярный и биполярный форматы данных. Этот выбор влияет на SP, PV, управляющий выход и результат интегратора.

Шаг 4. Выбор модулей ввода/вывода

После выбора количества контуров, измеряемых PV и значений SP нужно выбрать соответствующие модули ввода/вывода. Взгляните на рисунок на следующей странице. Во многих случаях модули ввода или вывода можно использовать совместно в нескольких контурах управления. В приведенном примере сигналы PV и управляющего вывода для двух контуров посылаются через один и тот же набор модулей.

Помните, что PLC**Direct** предлагает для различных типов и диапазонов сигнала аналоговые модули с 2, 4 и 16 каналами на модуль. Подробную информацию о конкретных модулях можно найти в каталоге продаж. Для аналоговых модулей есть отдельное руководство, которое понадобится при монтаже и написании программ.

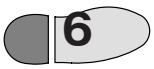


Шаг 5. Монтаж проводов и установка

После выбора и приобретения всех компонентов контура и модулей ввода/вывода, мы можем выполнить монтаж проводов и установку. Основные принципы монтажа проводов можно найти в главе 2 настоящего Руководства и, при необходимости, в Руководстве по модулям аналогового ввода/вывода DL450. Наиболее частыми ошибками при монтаже проводов в ходе установки управляющих элементов ПИД-контуров являются следующие:

- Легко перепутать полярность подключения проводов датчика.
- Обратите внимание на соединение сигнальной земли между компонентами контура

Шаг 6. Параметры контура



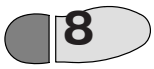
После монтажа проводов и установки можно выбрать параметры настройки контура. Самым простым способом запрограммировать таблицы контура является использование диалогов настройки ПИД-регулятора (PID Setup) в **Direct**SOFT (версии 2.0 и выше). Перед выбором вводимых значений обязательно разберитесь со всеми параметрами контура, описываемыми в данной главе.

Шаг 7. Проверка работы разомкнутого контура

После подключения датчиков и исполняющих механизмов и завершения ввода параметров цикла требуется тщательно проверить новую систему управления вручную (используя ручной режим).

- Убедитесь в правильности значения переменной PV, выдаваемого датчиком.
- Если это можно сделать безопасно, постепенно увеличивайте управляющий выход от 0%, отслеживая, как меняется PV *(и в правильном направлении ли происходит изменение)*.

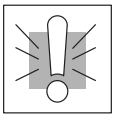
Шаг 8. Настройка контура.



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

Шаг 9. Выполнение цикла процесса

Если проверка замкнутого контура показывает, что PV отслеживает малые изменения SP, мы можем рассмотреть выполнение реального цикла процесса. Теперь необходимо действительно запрограммировать контур, чтобы генерировать требуемое значение SP в реальном времени. На этом этапе можно пропустить небольшую партию продукции через машину, изменяя SP в соответствии с технологией.



Внимание. Убедитесь в возможности отключения питания и остановки в экстренной ситуации на случай, если процесс выйдет из-под контроля. Потеря управления частью процессов может привести к поломке оборудования и/или причинению серьезного вреда персоналу.

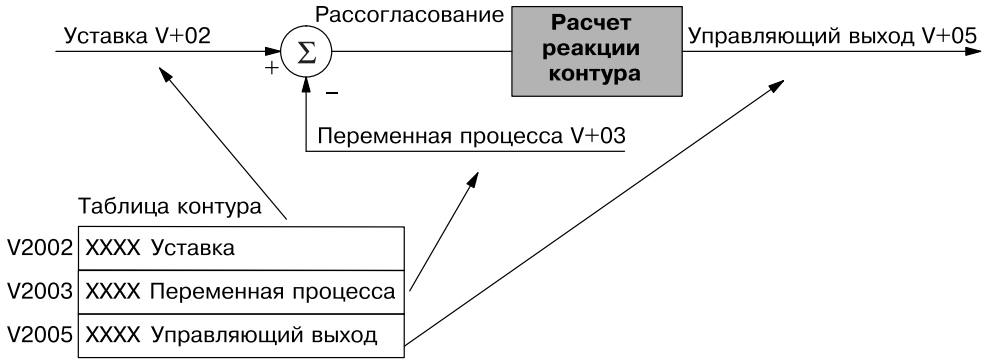
Шаг 10. Сохранение параметров контура

После завершения сеансов проверок и настройки контура не забудьте сохранить все параметры контура на диск. Параметры контура являются результатом немалой работы и заслуживают бережного отношения.

Основы работы контура

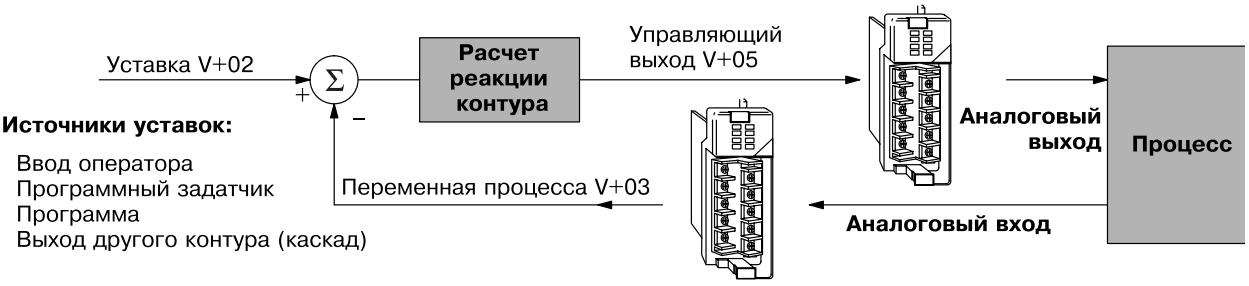
Местонахождение данных

Каждый контур ПИД-регулирования полностью зависит от команд и данных, находящихся в соответствующей таблице контура. На следующем рисунке показаны ячейки таблицы контура, соответствующие трем основным переменным ввода/вывода контура: SP, PV и управляющий выход. В приведенном примере таблица контуров начинается с V2000 (произвольная ячейка, выбираемая пользователем). Уставка (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 DL405 и произвольной ячейкой V-памяти, можно найти в руководстве по аналоговому модулю.



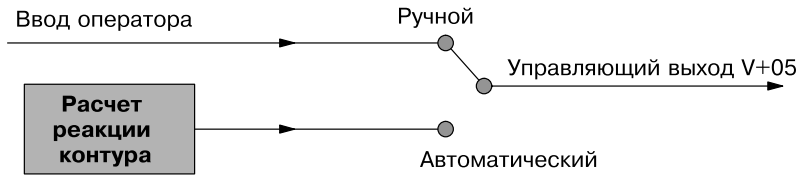
Как показано на рисунке, у уставки может быть несколько источников. Многие приложения в зависимости от режима цикла в разное время будут использовать два или более источников. Кроме того, способ генерирования уставки также определяется топологией контура и методом программирования. При использовании встроенного программного задатчика ПИД-контроллер автоматически записывает данные уставки в ячейку V+02. Однако **при генерации уставки из любого другого источника ее значение в ячейку таблицы должно записываться программно**.

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

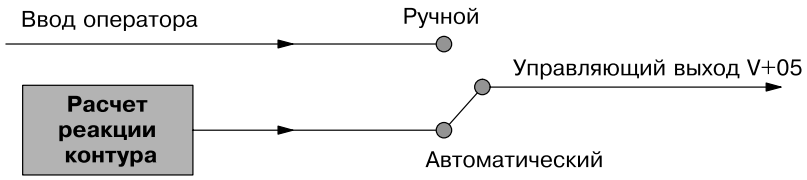
Режимы контуров

В промышленности часто возникают типовые ситуации использования контуров ПИД-регулирования. В каждом сценарии мы слегка изменяем источник данных для основных трех переменных: SP, PV и управляющего выхода, создавая имя режима для каждого сценария. Для процессора DL450 возможны следующие режимы — ручной, автоматический и каскадный. После введения в режимы мы рассмотрим, как переходить от режима к режиму.

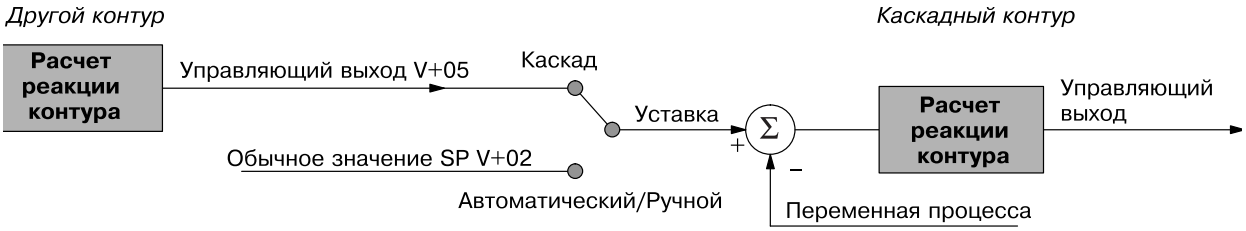
В ручном режиме контур не выполняет расчет реакции ПИД-регулятора (однако, аварийные сигналы контура действуют). Для такого контура процессор прекращает записывать значения в ячейку V+05 таблицы контура. Считается, что оператор (или другой интеллектуальный источник) вручную управляет выходом, отслеживая PV и записывая в V+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 V+05. Ниже приведена эквивалентная схема работы.



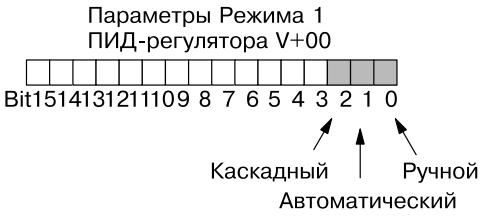
В каскадном режиме контур работает так же, как и в автоматическом с одним важным отличием. Источником данных для SP в этом случае является не ячейка V+02, а значение управляющего выхода другого контура (назначение каскадных контуров описано ниже в данной главе). Итак, в автоматическом или ручном режимах для расчета контура используются данные V+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.

Как изменять режимы контуров

Первые три бита слова V+00 Режима 1 ПИД-регулятора запрашивают режим работы соответствующего контура.

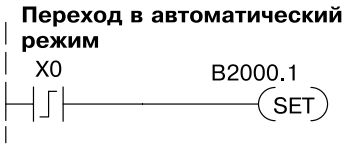


Примечание. Эти биты представляют собой запрос режима, а не команду на переход в него (некоторые условия могут помешать изменению конкретного режима- см. на следующей странице).

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:

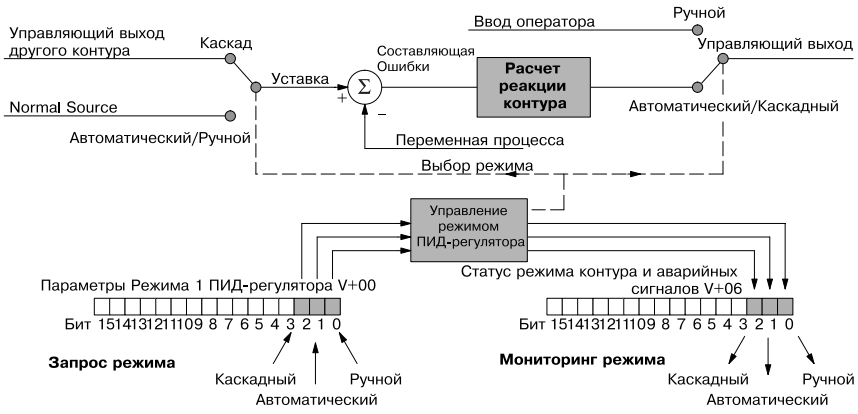
- **PID View** (просмотр ПИД-регулятора) в **Direct**SOFT. Самый простой способ. Щелкните по одной из кнопок, и **Direct**SOFT установит соответствующий бит.
- **ручной программатор.** Используйте статус слова (WD ST) для отслеживания содержимого V+00, которое является BCD/шестнадцатеричным значением из 4 цифр. Необходимо рассчитать и ввести новое значение для V+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 обмотки реле RLL). 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.

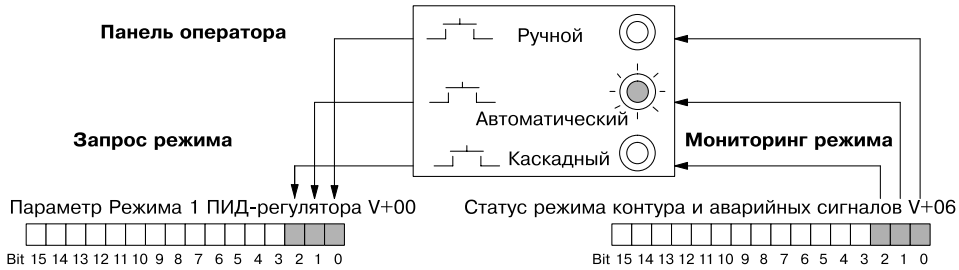
Так как изменение режима только запрашивается, контроллер контура ПИД-регулирования решает, когда разрешить изменение режима, и обеспечивает статус режима контура. Он устанавливает текущий режим в битах 0, 1 и 2 слова Статуса режима контура и аварийных сигналов, ячейка V+06 в таблице контура. Параллельные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



Управление режимами ПИД-регулятора с панели оператора

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

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



Влияние режимов работы ПЛК на режимы контура

Режимы ПЛК (программирование, рабочий) взаимодействуют с контурами как с группой. Вот как выглядит это взаимодействие:

- Когда ПЛК находится в программном режиме, все контуры переводятся в ручной режим, и расчеты контуров не производятся. Однако обратите внимание, что в программном режиме ПЛК отключаются модули вывода (включая аналоговые). Поэтому, если ПЛК находится в программном режиме, действенное ручное управление невозможно.
- Процессор позволяет изменять режим контура только, когда ПЛК находится в рабочем режиме. По существу, процессор записывает режимы всех 16 контуров как желаемые режимы работы. Если, пока ПЛК находился в режиме исполнения, 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 контуры. Новые контуры автоматически начинают работать в ручном режиме.

Подавление режима контура

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

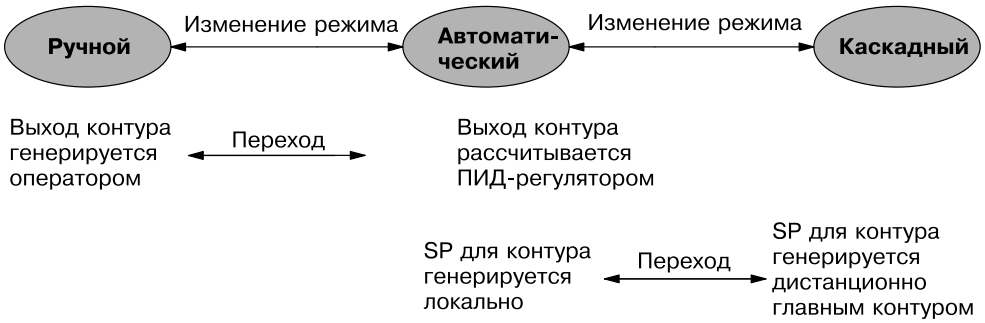
- Главный контур каскадной пары контуров не может перейти из ручного в автоматический режим, пока подчиненный контур находится в каскадном режиме.

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

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

Безударные переходы

В применении к управлению процессом слово «переход» имеет конкретное значение. Переход контура возникает при изменении его режима работы, как показано ниже. Когда мы меняем режимы контура, на самом деле мы вызываем переход от одного источника управления некоторым параметром контура к другому. Например, при изменении режима контура с 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 плавного перехода контроллера контура DL450: плавный переход I и плавный переход II. Тип перехода выбирается с помощью диалогового окна PID Setup (Настройка ПИД-регулятора) **Direct**SOFT. Или можно, как показано, использовать бит 3 V+00 режима 1 ПИД-регулятора.

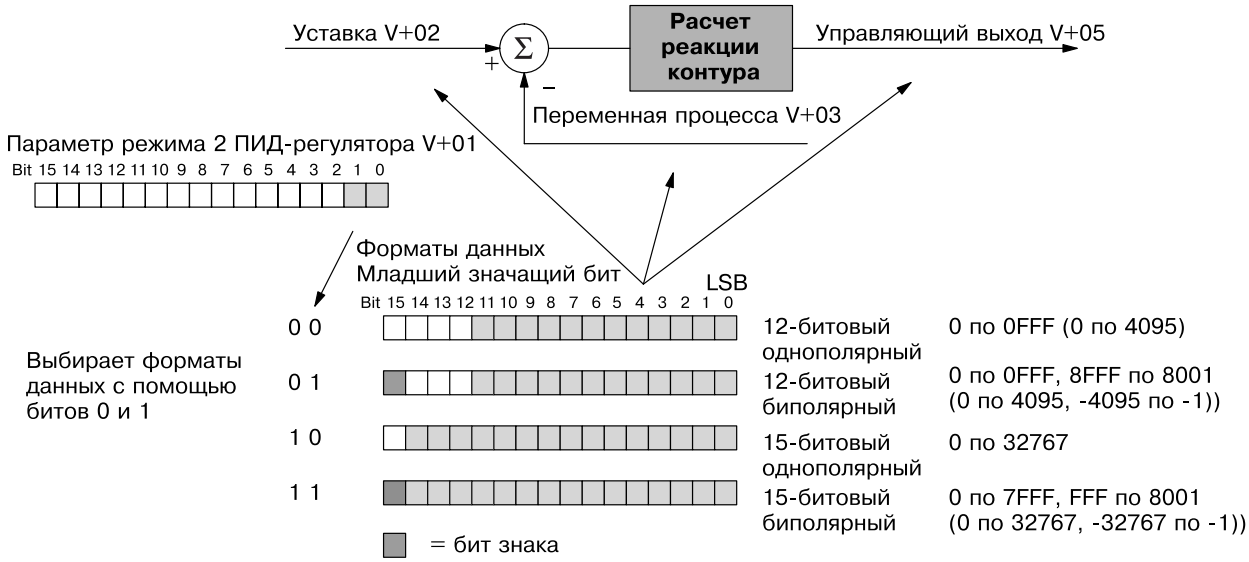
Характеристики типов безударных переходов I и II приведены в следующей таблице. Обратите внимание, их работа также зависит от используемого алгоритма ПИД-регулятора, формы ПИД-уравнения (позиционная или в скоростная). Заметим, что при использовании уравнения ПИД-регулятора в скоростной форме необходимо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

Тип перехода	Бит выбора перехода	Алгоритм ПИД-регулятора	Ручной в Автоматический	Автоматический в Каскадный
Безударный переход I	0	Позиционный	Устанавливает смещение = управляющему выходу Устанавливает SP = PV	Устанавливает выход основного контура = PV подчиненного контура
		Скоростной	Устанавливает SP = PV	Устанавливает выход основного контура = PV подчиненного контура
Безударный переход II	1	Позиционный	Устанавливает смещение = управляющему выходу	отсутствует
		Скоростной	отсутствует	отсутствует

Конфигурирование данных контуров ПИД-регулирования

Форматы данных параметров контура

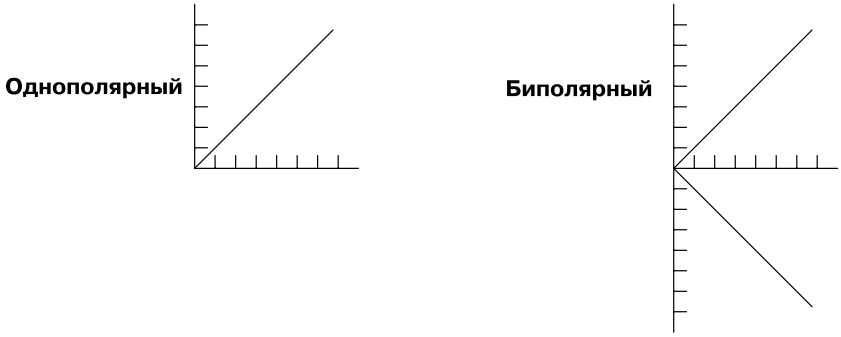
С выбором диапазона и разрешающей способности переменной процесса связан выбор формата данных трех основных переменных контура: уставки (SP),переменной (PV) и управляющего выхода (этот формат данных также используется суммой интегратора в V+04). Четырьмя возможными форматами данных являются 12- или 15-битовый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



Форматы данных — это очень мощный 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

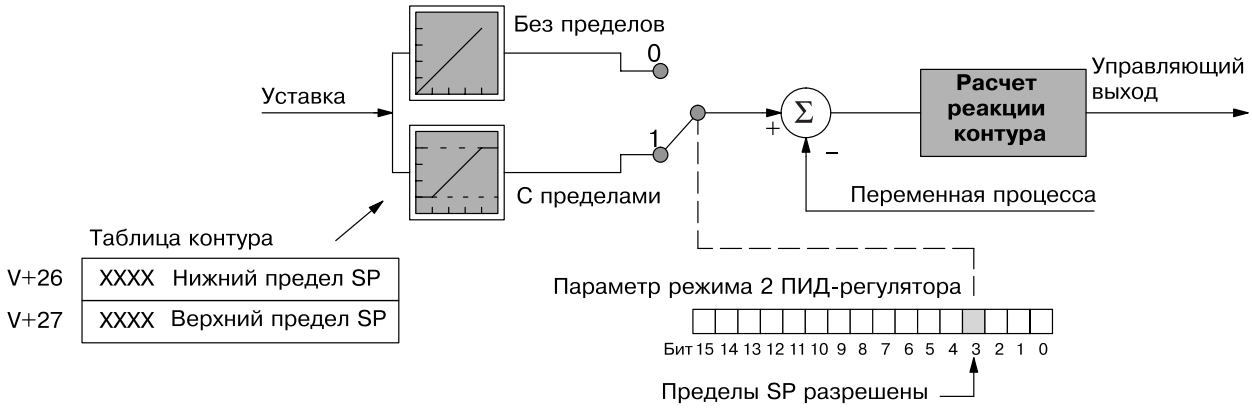
Во многих приложениях с периодическим технологическим процессом датчики или исполняющие механизмы взаимодействуют с аналоговыми модулями DL405, используя сигналы в диапазоне 4-20 мА. Этот тип сигнала обладает встроенным 20% смещением, так как нулевой точкой является 4 мА, а не 0 мА. Однако не забывайте, что аналоговые модули преобразуют сигналы в данные с одновременным устранением смещения. Например, сигнал 4-20 мА часто преобразуется в 0000 - 0FFF (шестнадцатеричное), или 0 - 4095 (десятичное). В этом случае потребуется только выбрать 12-битовый однополярный формат данных и убедиться, что программа правильно выполняет обмен данными между таблицей контура и аналоговыми модулями.

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

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

Пределы уставки (SP)

Уставка в ячейке V+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 контуры используют два и более возможных источника, в разное время записывающих значение уставки, и заданные пределы помогут защитить процесс от влияния ввода неправильного значения SP. На следующем рисунке SP показана работа выбираемых пределов, разрешаемая словом V+01 режима 2 ПИД-регулятора, бит 3. Если пределы включены, то ячейки V+26 и V+27 определяют нижний и верхний пределы SP, соответственно. Эти пределы используются при расчете реакции контура, поэтому в V+02 можно записывать любое значение.

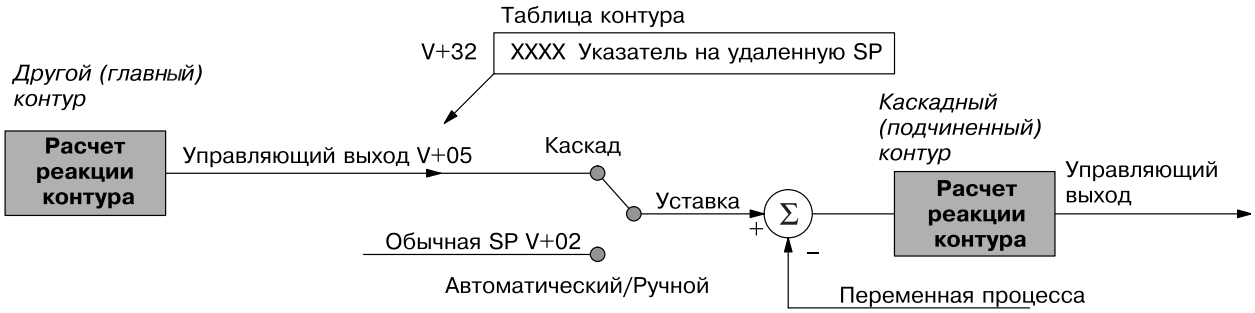


При расчете контура установленные верхний и нижний пределы SP проверяются перед каждым вычислением. Это означает, что программа в ходе выполнения процесса может изменить значения пределов, позволяя поддерживать узкий защитный диапазон входного значения SP.

Ячейка удаленной уставки (SP)

Вспомните, что в общем случае существует несколько возможных источников данных 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 V+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 V+32 таблицы контура. Для контуров, которые будут управляться каскадно (являясь подчиненным контуром), понадобится записать в эту ячейку адрес управляющего выхода главного цикла. Найдите начальную ячейку таблицы параметров основного цикла и добавьте к ее адресу смещение +05.

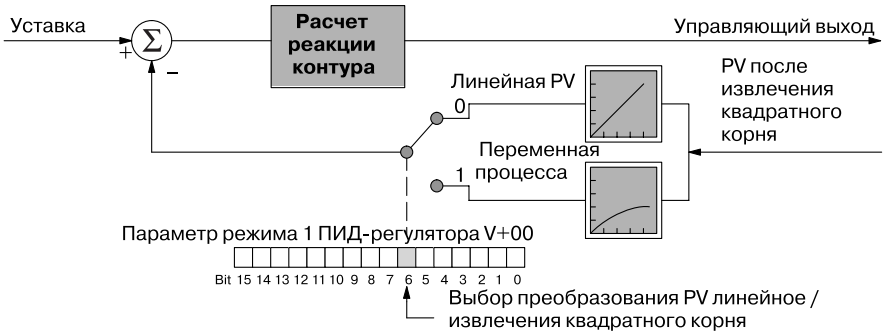


Диалоговое окно Loop Setup (Настройка контура) **Direct**SOFT позволит задать указатель на удаленную SP, если известен соответствующий адрес. В противном случае можно ввести значение указателя с помощью ручного программатора или задать его программно с помощью инструкции LDA.

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 отклика. Большинство температурных датчиков почти линейны в рабочем диапазоне. Однако измерения расхода, использующие измерительную диафрагму, дают сигнал, представляющий (приблизительно) квадрат потока. Следовательно, перед использованием сигнала в линейной системе управления (например, в контуре с ПИД-регулятором) из него необходимо извлечь квадратный корень.

Доступны некоторые 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 V+00 параметров режима 1 ПИД-регулятора.



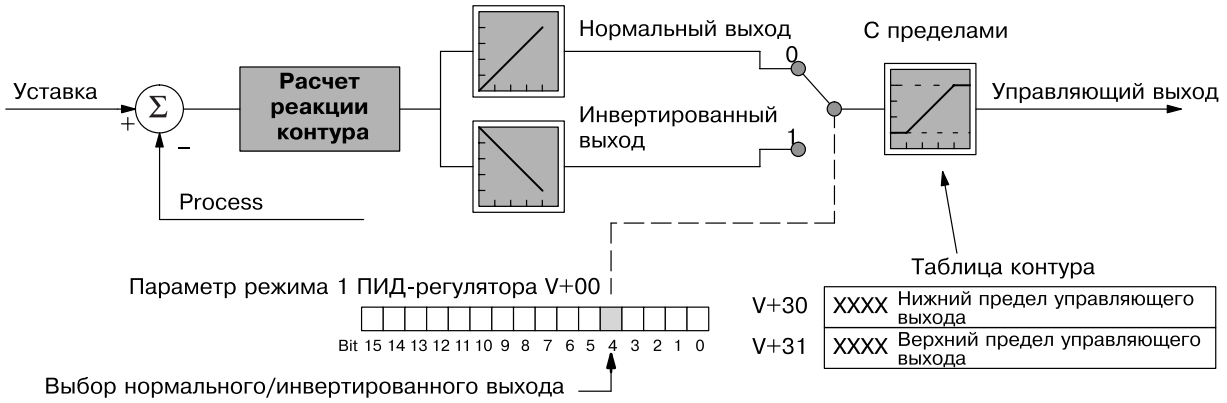
Важно. При использовании извлечения квадратного корня PV необходимо масштабировать шкалу SP, так как контур управляет выходом так, чтобы квадратный корень из PV равнялся входу PV. Разделите требуемое значение SP на квадратный корень из аналогового диапазона и используйте результат в ячейке V+02 для SP.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

Формат данны	Масштабирование SP	Диапазон SP	Диапазон PV
12-битовый	SP = вход PV / 64	0 – 64	0 – 4095
15-битовый	SP = вход PV / 181.02	0 – 181	0 – 32767

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 нижним и верхним пределами. Значения V+30 и V+31 могут быть установлены один раз с помощью диалогового окна PID Setup (настройка ПИД-регулятора) **Direct**SOFT. Или запись значений в ячейки таблицы и изменения пределов во время работы контура может выполняться программно.

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



Другим доступным вариантом является выбор нормального/инвертированного выхода (называемого в **Direct**SOFT «forward/reverse» — прямой/обратный). Для конфигурирования выхода используется бит 4 слова V+00 параметров режима 1 ПИД-регулятора. Независимо от используемого формата (однополярный/биполярный) нормальный выход увеличивается при положительных отклонениях и уменьшается при отрицательных (где отклонение = (SP-PV)). Инвертированный выход меняет направление изменения выхода. Выбор обычного/инвертированного выхода используется для настройки контуров с прямым/обратным действием.

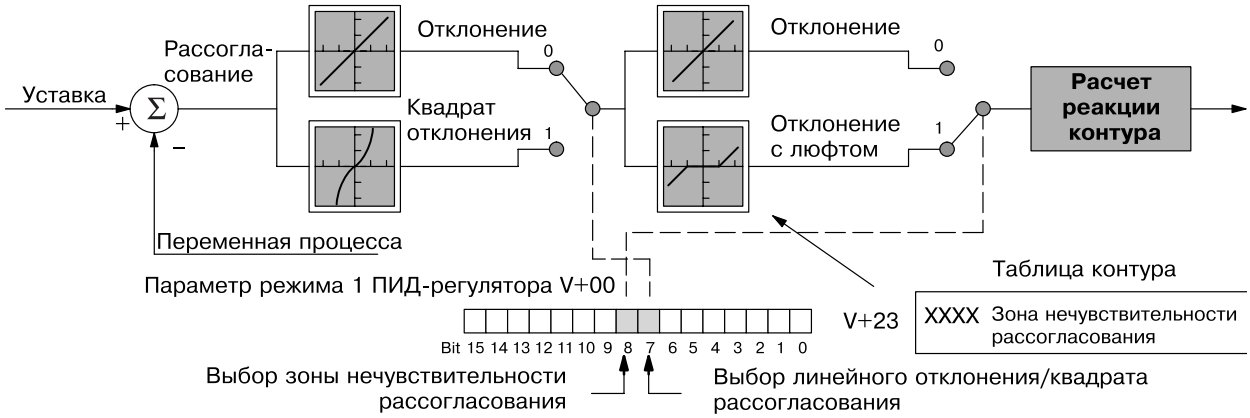
Этот выбор, в конце концов, определяется направлением отклика переменной процесса на изменение управляющего выхода в определенном направлении. Подробно контуры с прямым/обратным действием описаны в разделе «Алгоритмы ПИД-регулятора».

Конфигурирование рассогласования

Рассогласование 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 получить 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 V+00 режима 1 ПИД-регулятора позволяет выбрать линейную или квадратичную 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. 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 V+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.

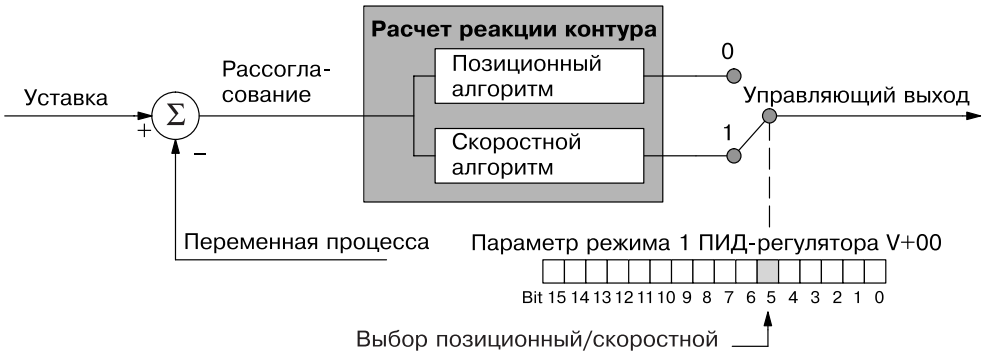
Алгоритмы ПИД-регулирования

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 DL450 реализует уравнение ПИД-регулирования цифровым образом, программно решая основное уравнение. Модули ввода/вывода служат только для преобразования электронных сигналов в цифровую форму (или наоборот).

DL450 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).
- ПИД-алгоритм в форме *скорости*. Управляющий выход рассчитывается так, чтобы он представлял скорость изменения (приращение) значения PV, чтобы PV было равно 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) **Direct**SOFT. Или используйте для выбора алгоритма, как показано ниже, бит 5 слова V+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:

$$Mx_o = M_o$$

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

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

$$M_n = K_c * e_n + K_r * (e_n - e_{n-1}) + Mx_n \ldots\ldots$$

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).

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

Управляющий выход — M_n
Пропорциональный член — $K_c * e_n$
Интегральный член — $K_i * \sum_{i=1}^{n} e_i$
Дифференциальный член — $K_r * (e_n - e_{n-1})$
Первоначальный вход — M_o
Член смещения — 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 алгоритма в форме скорости может быть получено путем преобразования формулы позиционного алгоритма путем вычитания 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.

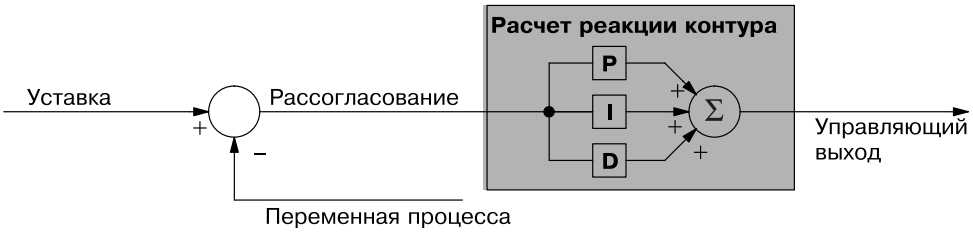
В контуре с обратным действием, как показано ниже, коэффициент усиления процесса отрицателен. Увеличение управляющего выхода приводит к уменьшению PV. Такие контуры обычно встречаются при управлении охлаждением, когда рост охлаждающего входа приводит к уменьшению PV (температуры). Соответственно, контуры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

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

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.

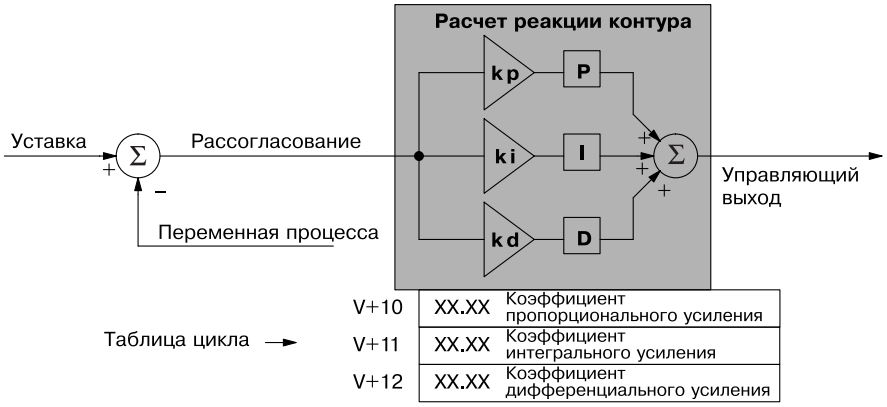


П-, И- и Д- составляющие при управлении играют следующие роли:

- **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: kp, ki и kd, 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.

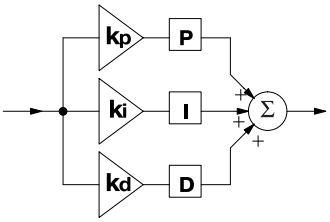


Таблица контура

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

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

Выбор единиц

В окне просмотра тренда пакета **Direct**SOFT 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 действием.

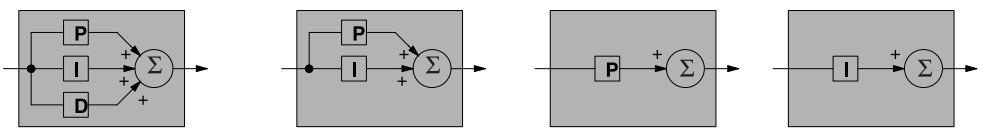
Интегральный коэффициент усиления. Диапазон значений от 0001 до 9998, но внутри значения используются как xx.xx. Задание «0000» или «9999»обнуляет интегральный коэффициент усиления, удаляя интегральную составляющую из уравнения ПИД-регулятора. Это позволяет подстроиться под приложения, для которых требуются 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 действием. 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. Это очень важно помнить при настройке контура.

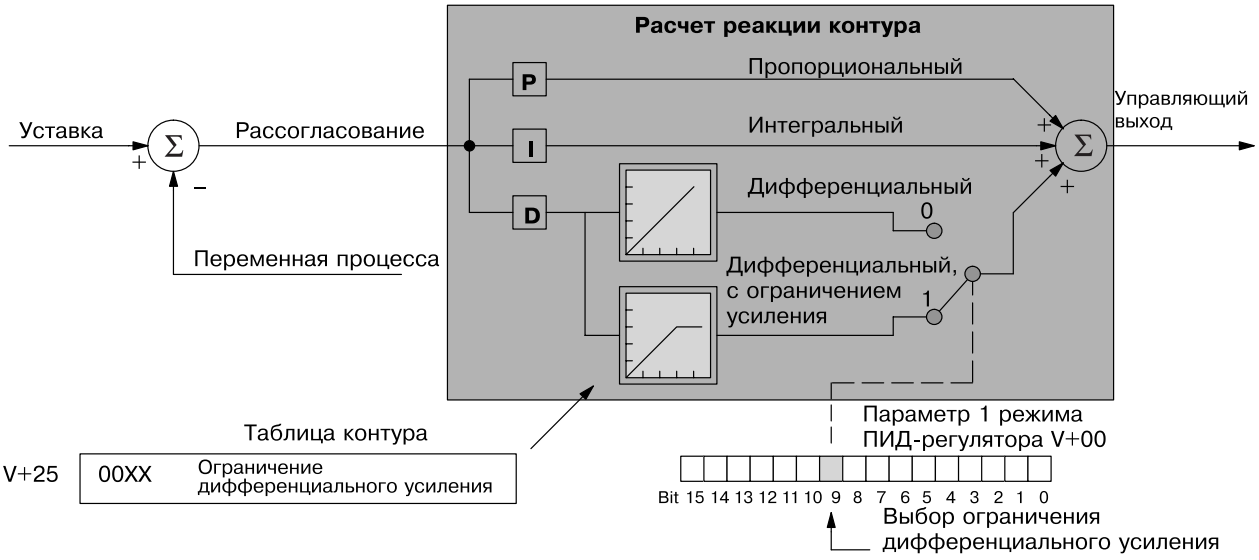
Использование подмножества управляющих элементов ПИД-регулятора

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 DL450. 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 V+00 режима 1 ПИД-регулятора.



Ограничение дифференциального усиления в ячейке V+25 должно иметь значение от 0 до 20, в формате BCD. Эта установка 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, показанная ниже. В таблице контура она находится в ячейке V+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; Пропорциональная составляющая; Интегральная составляющая; Дифференциальная составляющая; Начальный выход

Таблица контура

V+04 | XXXX | Составляющая смещения

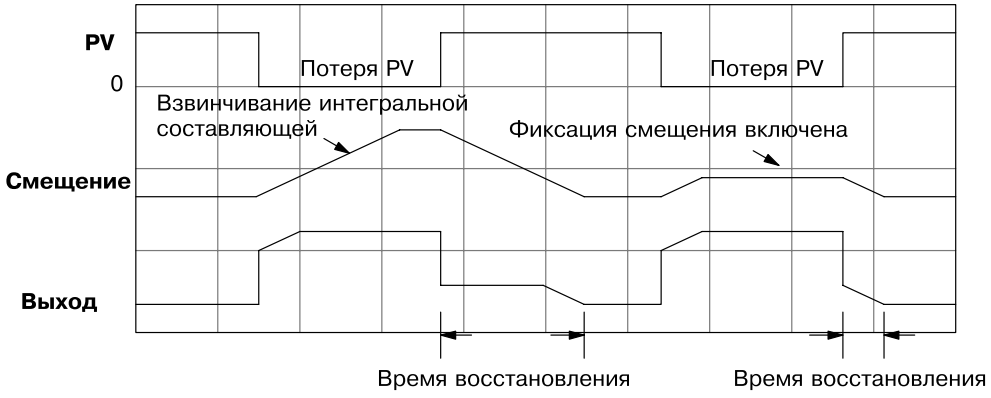
Член смещения

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

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

Фиксация смещения

Термин «взвинчивание интегральной составляющей» или «интегральная яма» (reset windup) относится к нежелательным характеристикам поведения интегратора, естественного при определенных условиях (см. следующий рисунок). Пусть сигнал PV пропадает, и значение PV становится равным нулю. Последствия этого серьезного сбоя — выход контура усиливается за счет роста 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 значение выхода 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 с предварением, обсуждаемом ниже в данной главе, релейная 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.

Тестирование разомкнутого контура

Очень важно проверить основные характеристики нового контура до его перевода в автоматический режим. 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 V+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 V+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.

Тестирование замкнутого контура

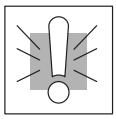
Наконец наступает волнующий момент, когда мы впервые действительно замкнем контур (перейдем в автоматический режим). Перед переключением в автоматический режим сверьтесь со следующим списком проверок:

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

Примечание. Рекомендуется использовать меню установки просмотра ПИД-регулятора для выбора «ручной» установки вертикального масштаба для областей SP/PV и смещение/управляющий выход. В противном случае функция автоматического масштабирования изменит вертикальный масштаб 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 (V+04). Если оно не равно нулю, установите его в ноль с помощью **Direct**SOFT или ручного программатора, и т.д.

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



Осторожно. 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.

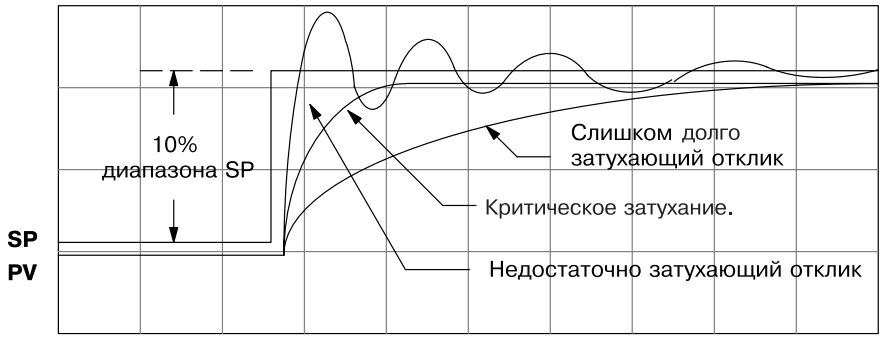
- В этот момент должно выполняться SP = PV благодаря функции безударного перехода. Немного увеличьте SP, создавая отклонение. Если активен только пропорциональный коэффициент усиления, и составляющая смещения равна 0, можно легко проверить значение управляющего выхода.

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 долго 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.
- Критическое затухание. 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

Функция автонастройки контроллера контура процессора DL450 выполняется только по команде инженера управления процессом. Таким образом, автонастройка не выполняется непрерывно в ходе процесса (такое управление было бы адаптивным). При возникновении в динамике контура заметных изменений (масса процесса, размер исполнительного механизма и т.п.), понадобится повторно выполнить процедуру настройки, чтобы получить новые коэффициенты усиления, требуемые для оптимального управления.

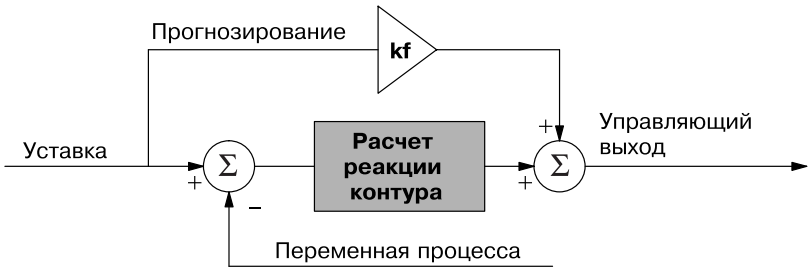
Настройка каскадных контуров

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, 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 отклик подчиненного контура в автоматическом режиме на 10% изменение SP будет быстро затухать. 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.

Управление по возмущению

Управление с упреждением (по возмущению) представляет собой улучшение стандартного управления в замкнутом контуре. Оно лучше всего подходит для уменьшения влияния *поддающегося количественному определению* и *предсказуемого возмущения* или резкого изменения уставки. DL450 поддерживает возможность управления по возмущению. Однако лучше реализовать и настроить контур без упреждения, добавляя его 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.

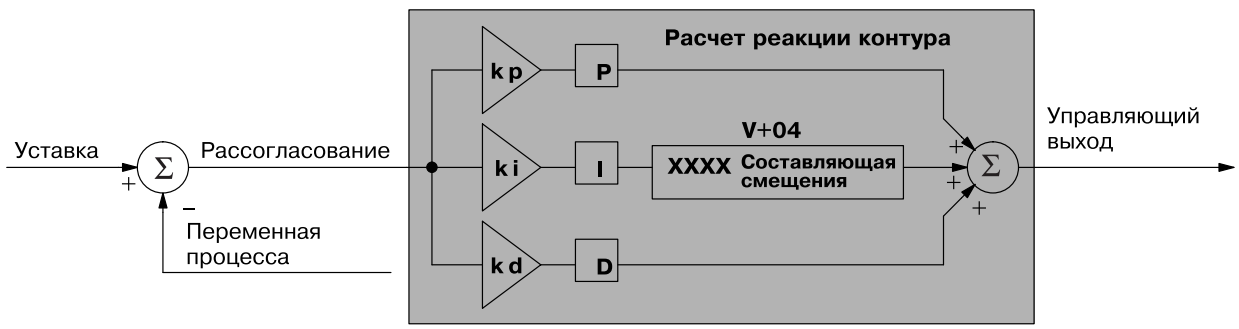


В предыдущем разделе при рассмотрении составляющей смещения, говорилось, что значение этой составляющей задает для управляющего выхода «рабочую область» или рабочую точку. *При колебании рассогласования вокруг нулевого значения, величина выхода колеблется вокруг значения смещения.* Теперь при изменении 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 DL450, как показано ниже, использовать упреждение очень легко. Пользователь получает доступ к составляющей смещения, расположенной в специальной ячейке чтения/записи, ячейке таблицы параметров ПИД-регулятора V+04.

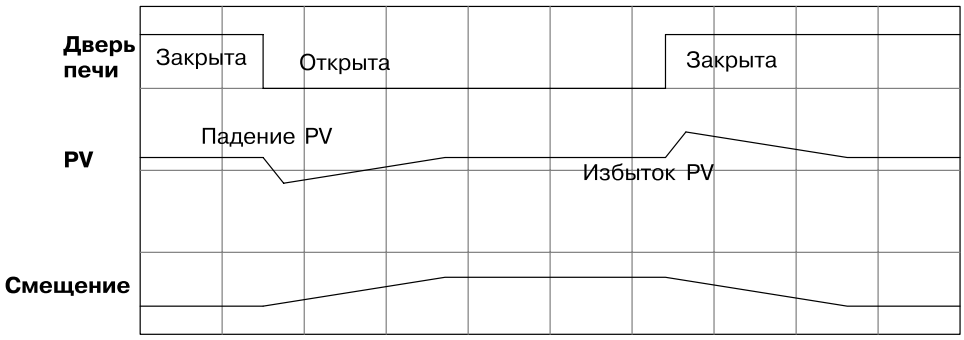


Для изменения смещения (рабочей точки) программе нужно только записать требуемое значение в V+04. При расчете контура с ПИД-регулятором значение смещения считывается из V+04 и изменяется в зависимости от текущего расчета коэффициента интегратора. Затем результат записывается обратно в ячейку V+04. Такой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

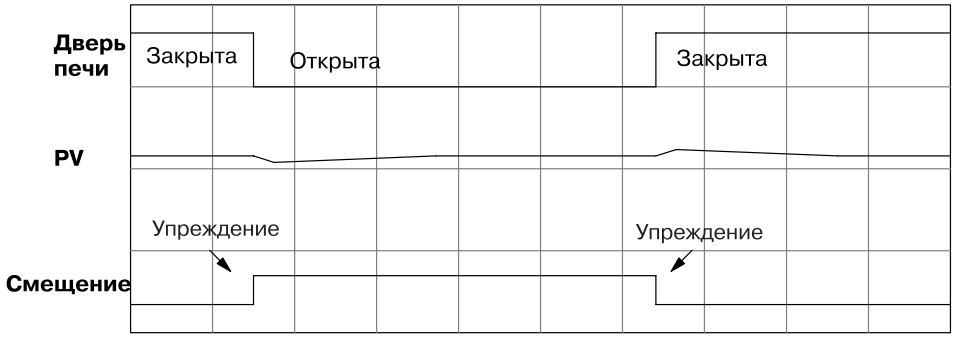
Примечание. При записи составляющей смещения нужно быть аккуратным, обеспечивая только однократную запись значения в момент возникновения новой рабочей точки смещения. Если программа записывает значение смещения при каждом сканировании, то интегратор контура по сути дела отключается.

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.



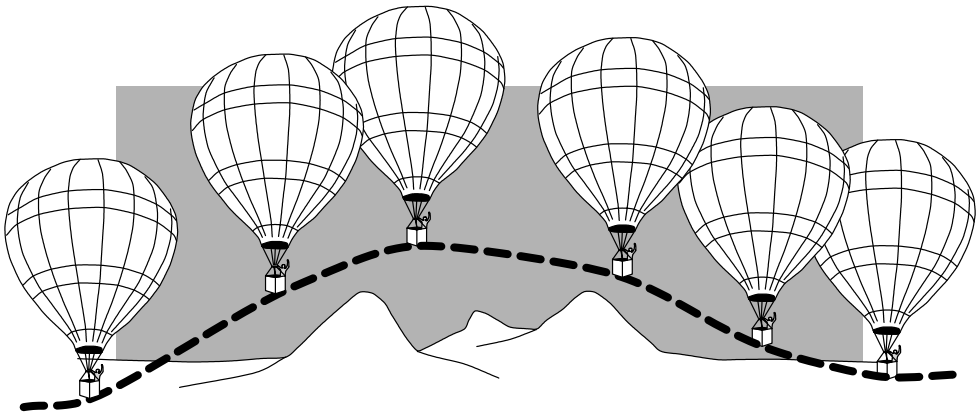
Сначала запишем величину изменения смещения, создаваемого контроллером контура при открытии или закрытии двери. Затем напишем программу, контролирующую положение концевого выключателя двери печи. При открытии двери наша программа считывает текущее значение смещения из V+04, добавляет нужное изменение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

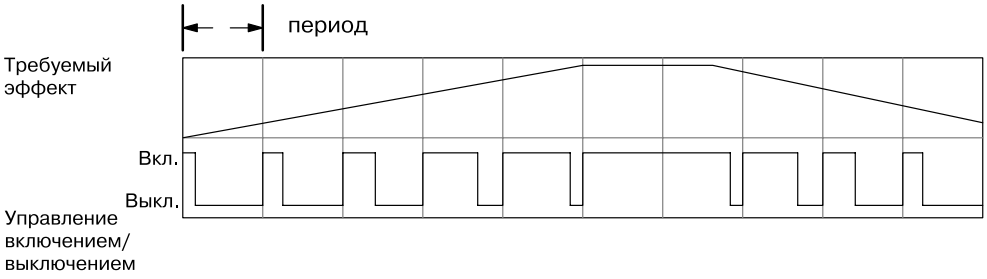
Широтно-импульсное управление

Контроллер контура ПИД-регулирования DL450 создает непрерывный 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*.

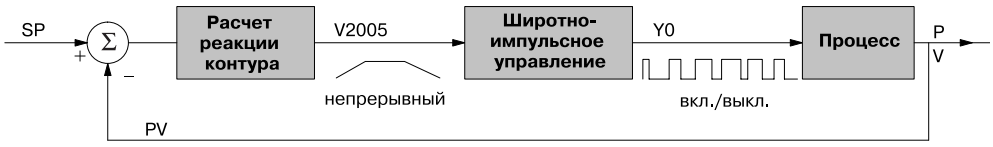
Широтно-импульсное управление является приближением непрерывного управления за счет особенностей рабочего цикла — отношения времени включения к времени выключения. На следующем рисунке приведен пример того, как рабочий цикл обеспечивает приближение к непрерывному уровню при усреднении с помощью большой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

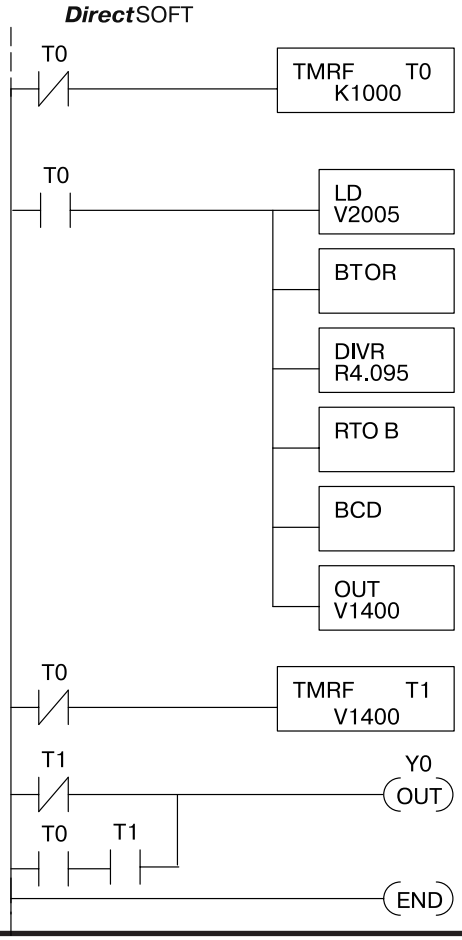
Пример программы управления с помощью включения/выключения

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



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

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



Использует для основной временной развертки быстрый таймер (с разрешением 0.01 с). K1000 обеспечивает предварительно установленное значение 10 с. Контакт Нет 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 это значение в действительное число, так как необходимо выполнить некоторое масштабирование.

Значение управляющего выхода делится на 4.095. Это преобразует используемый диапазон 0 - 4095 в 0 - 1000, приводя его в соответствие с 10-секундным диапазоном таймера.

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

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

Выводит наш результат в V1400. Это произвольно заданная ячейка, содержащая установку 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

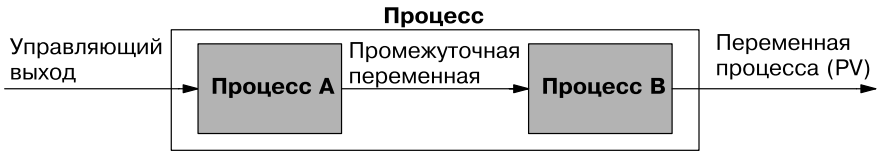
Каскадное управление

Введение

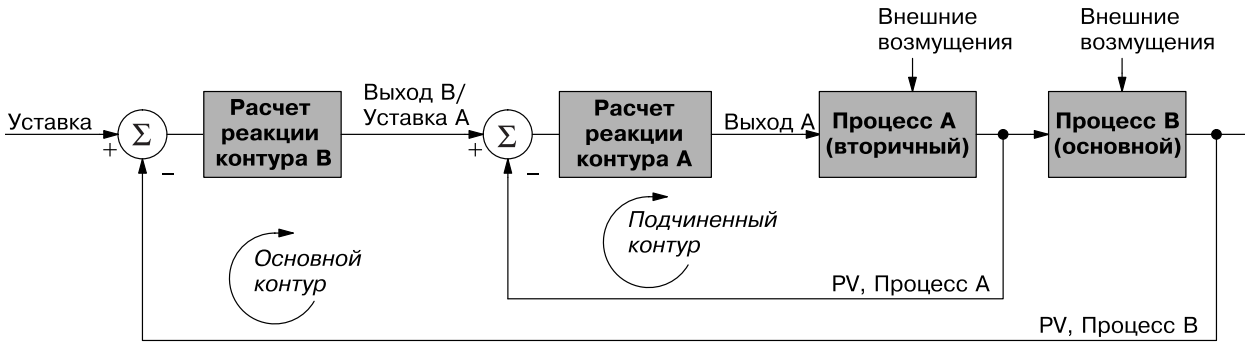
Каскадные контуры представляют собой усовершенствованный метод управления, в определенных ситуациях более выгодный чем управление с помощью отдельных контуров. Как следует из названия, каскад означает, что один контур подключен к другому. Помимо ручного (разомкнутый контур) и автоматического (замкнутый контур) режимов DL450 также обеспечивает каскадный режим.

Примечание. Каскадные контуры — это более сложный метод управления. Поэтому их использование рекомендуется только для опытных инженеров управления процессом. Если производственный процесс является сложным и включает задержку между управляющим выходом и изменением переменной процесса, использование даже самого идеально настроенного отдельного контура может приводить к медленному и неточному управлению. Такое может случиться, если исполнительный механизм, использующий одно физическое свойство, в конечном счете воздействует на переменную процесса, измеряемую с помощью другого физического свойства. Задание промежуточной переменной позволит разделить процесс на две части, как показано на следующем рисунке.

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



На следующем рисунке показана каскадная система управления, представляющая собой просто вложение одного контура в другой. Внутренний контур называется подчиненным, а внешний контур — главным. Для максимальной устойчивости из двух контуров более быстрым откликом должен обладать внутренний. Для измерения промежуточной переменной (PV процесса А) 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 А в подчиненном контуре, расчет ПИД-коэффициентов контура А может исправить получающуюся ошибку до того, как эффект будет обнаружен основным контуром.

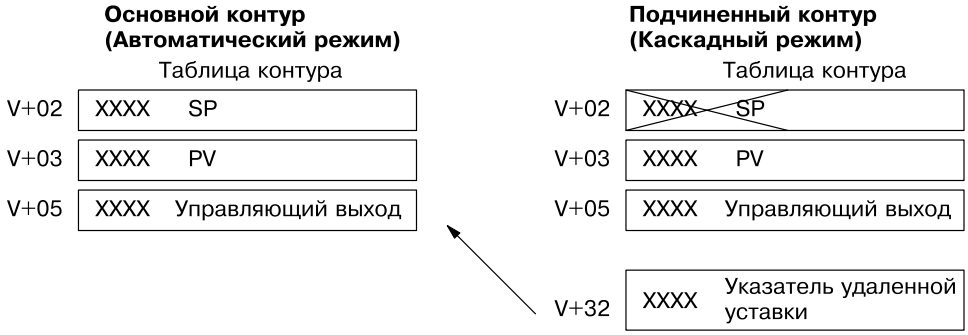
Каскадные контуры в процессоре DL450

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.

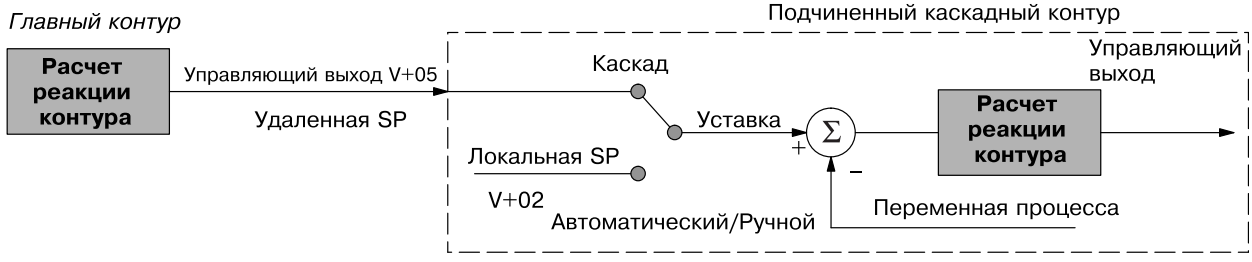
С помощью DL450 можно объединить в каскад любое нужное число контуров, а также создать несколько групп каскадных контуров. Для правильной работы каскадных контуров необходимо использовать для главного и подчиненного контуров одинаковые диапазоны данных (12/15 битовые) и настройки полярности.

Для подготовки подчиненного контура к работе в каскадном режиме необходимо, как показано ниже, задать в ячейке V+32 его таблицы контура значение Указателя удаленной уставки. Значением указателя должен быть адрес ячейки V+05 (управляющий выход) главного контура. В каскадном режиме второстепенный контур будет игнорировать собственный локальный регистр уставки SP (V+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) **Direct**SOFT автоматически считывает значение управляющего выхода главного контура и выводит это значение как SP подчиненного контура. Обычная ячейка SP подчиненного контура, V+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.

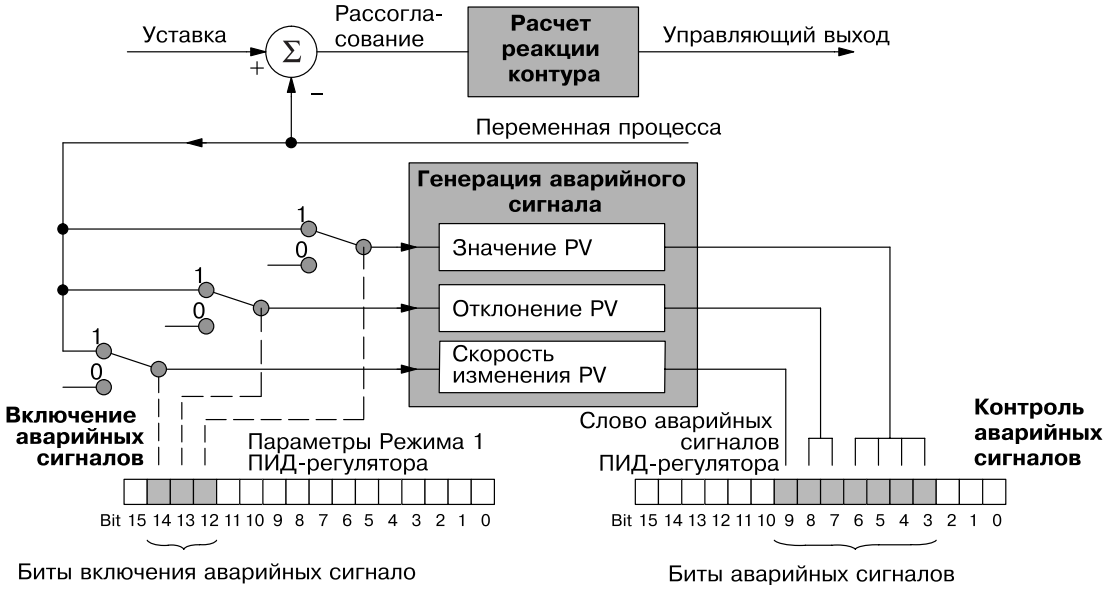
Аварийные сигналы процесса

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

Процессор DL450 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 значенияPV 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 отклонения, устраняя «дребезг» аварийного сигнала вблизи пороговых значений сигналов

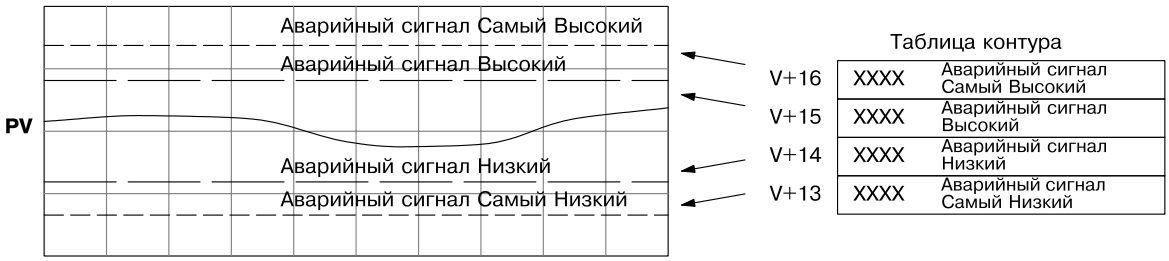
Пороги сигналов являются полностью программируемыми, и каждый тип аварийного сигнала может включаться и контролироваться независимо. На следующей схеме показана функция контроля PV. Аварийные сигналы включают/выключают биты 12, 13 и 14 слова V+00 параметров Режима 1 ПИД-регулятора в таблице параметров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



В отличие от расчета ПИД-коэффициентов аварийные сигналы работают всегда, когда процессор находится в рабочем режиме. Контур может находиться в ручном, автоматическом или каскадном режиме, и аварийные сигналы будут работать, если 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 ложный статус, 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.

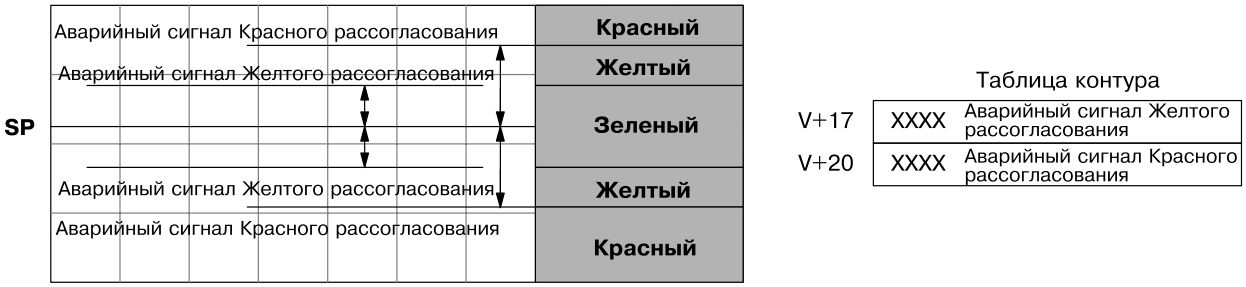


Если процесс останется какое-то время неуправляемым, PV в конце концов пересечет один из внешних порогов, которые называются Самый Высокий(High-High Alarm и Самый Низкий(Low-Low Alarm). Их пороговые значения задаются с помощью приведенных выше регистров таблицы контура. Аварийный сигнал Самый Высокий 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, что легко выполняется с помощью команд бит-из-слова. Кроме того, аварийные сигналы ПИД-регулятора можно контролировать, используя **Direct**SOFT.

Аварийные сигналы рассогласования PV

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

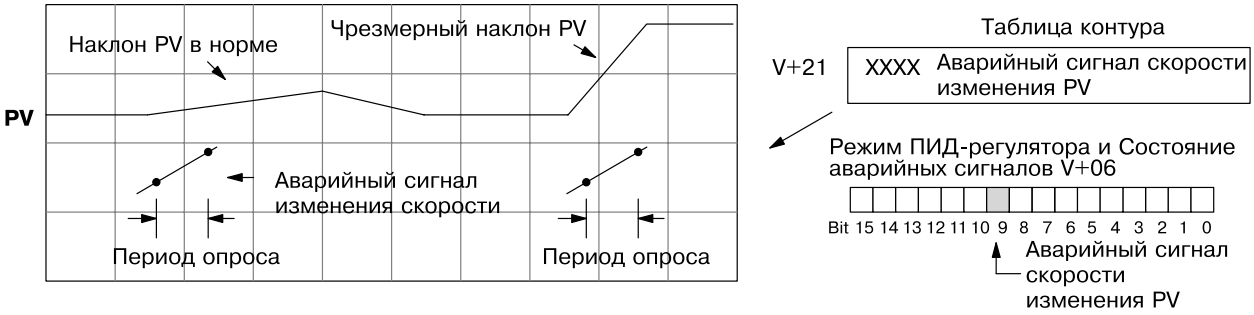
Аварийные сигналы рассоглосования PV индицируются двумя битами слова Режим ПИД-регулятора и Статус аварийных сигналов в таблице контура, как показано справа. Настоятельно рекомендуется программно контролировать эти биты, что легко выполняется с помощью команд бит-из-слова. Кроме того, аварийные сигналы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

Аварийный сигнал рассогласования PV может включаться и отключаться независимо от других аварийных сигналов PV с помощью бита 13 слова V+00 параметров режима 1 ПИД-регулятора. Помните, что вместе с аварийными сигналами рассогласования и абсолютного значения PV может включаться функция запаздывания сигнала, обсуждаемая в конце данного раздела.

Аварийный сигнал скорости изменения PV

Мощным средством раннего оповещения о сбое процесса является контроль *скорости изменения* переменной PV. Массы большинства технологических процессов велики, 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 точки разрушения производственного оборудования.

Контроллер контура DL450 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 V+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 V+03) обозначает 10 единиц счета на градус, а период отсчетов контура составляет 2 секунды. Для преобразования наших инженерных единиц в единицы счета/период отсчетов воспользуемся следующей формулой:

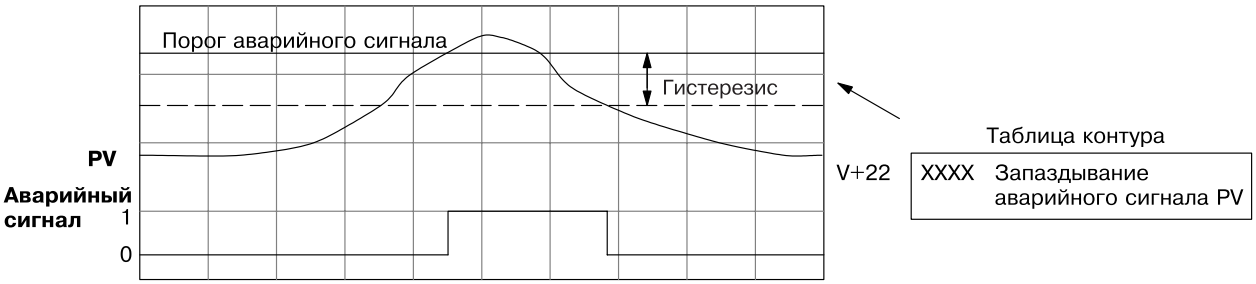
$$\text{Аварийное значение скорости изменения} = \frac{\text{15 градусов}}{\text{1 минута}} \times \frac{\text{10 единиц счета/градус}}{\text{30 опросов/минуту}} = \frac{150}{30} = \text{5 единиц счета/период опроса}$$

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

Гистерезис аварийных сигналов PV

Аварийные сигналы абсолютного значения PV и отклонения PV программируются с помощью пороговых значений. Когда абсолютное значение или отклонение 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 отклонения PV, заданное значение запаздывания должно быть меньше заданного значения отклонения. На следующем рисунке показано, как работает гистерезис, когда значение PV пересекает порог туда и обратно.



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

Ошибка программирования порогов аварийного сигнала

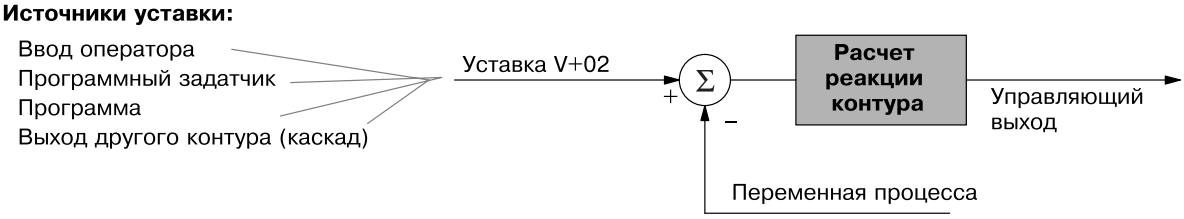
Чтобы значения порогов аварийных сигналов PV были заданы правильно, они должны подчиняться некоторым математическим соотношениям, перечисленным ниже. Если эти требования не выполняются, устанавливается, как показано справа, бит Ошибки программирования аварийного сигнала.

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

Программный задатчик

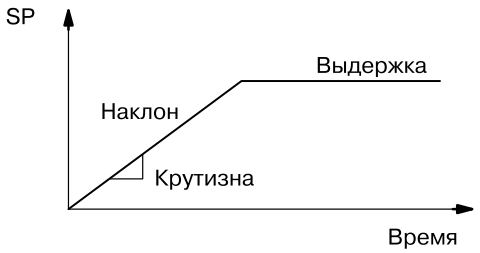
Введение

При обсуждении 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 V+02 значение SP, отвечает пользовательская программа.*



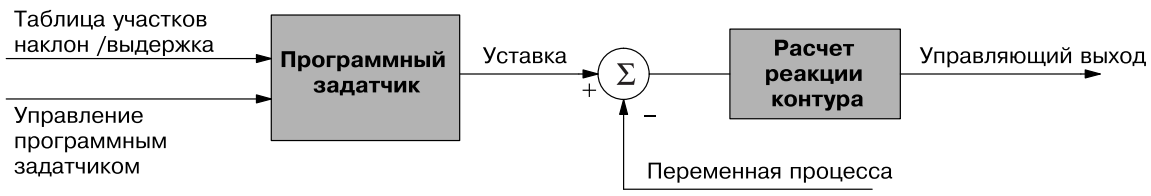
Если SP процесса меняется редко или допускает ступенчатые изменения, использование программного задатчика, возможно, не потребуется. Однако для некоторых процессов потребуется точно управлять изменениями значения SP. Программный задатчик может значительно снизить объем программирования, нужный для подобных 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 шестнадцати шагов программного задачника (16 участков).
- Программный задатчик может работать всегда, когда ПЛК находится в рабочем режиме. Его работа не зависит от режима контура (ручной или автоматический).
- Управляющие воздействия реального времени включают Start (запуск), Hold (остановка), Resume (продолжение) и Jog (толчок).
- Контроль сигнала программного задатчика включает Завершение профиля, Отклонение PV на участке-выдержке (SP минус PV) и номер текущего шага профиля наклон /выдержка .

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

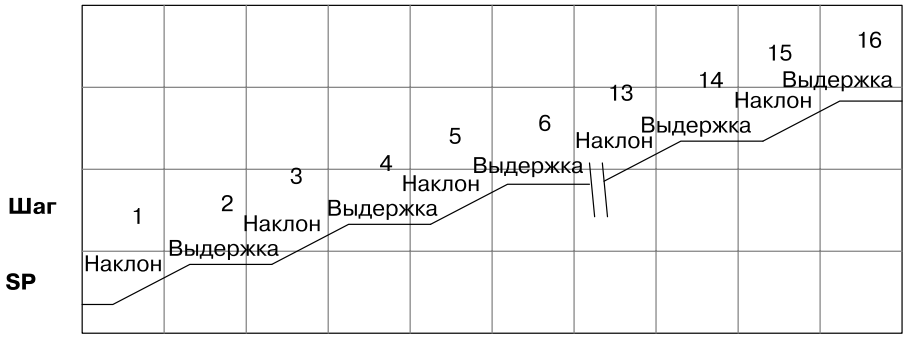
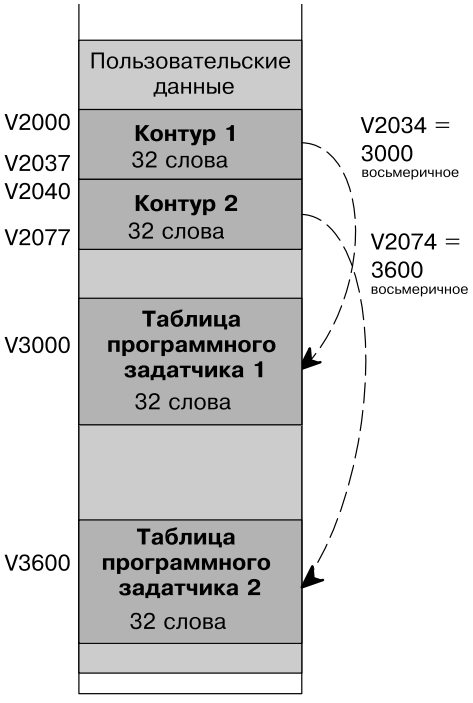


Таблица программного задатчика

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



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

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

- **Окончание участка наклона.** 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 наклон.** 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.** 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.** Необязательный параметр, задает допустимое отклонение PV вверх и вниз от значения SP во время выдержки. Бит состояния аварийного сигнала отклонения PV устанавливается программным задатчиком.

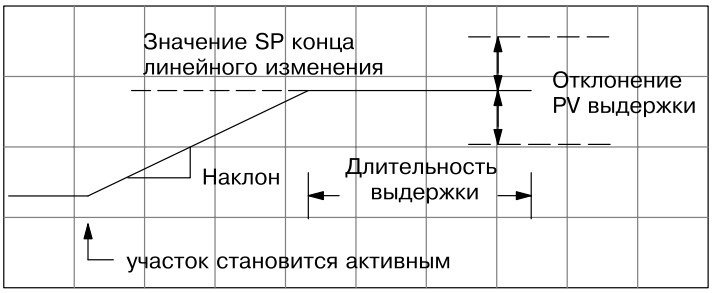


Таблица программного задатчика

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

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

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

Флаги таблицы программного задатчика

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

Бит	Описание битов флагов программного задатчика	Чтение/ Запись	Бит=0	Бит=1
0	Запуск программного задатчика	запись	–	0⇒1 Запустить
1	Остановка программного задатчика	запись	–	0⇒1 Удержать
2	Возобновление работы программного задатчика	запись	–	0⇒1 Продолжить
3	Толчок программного задатчика на шаг	запись	–	0⇒1 Толчок
4	Завершение профиля программного задатчика	чтение	–	Завершение
5	Отклонение входа PV от сигнала программного задатчика	чтение	Выкл	Вкл
6	Программный задатчик приостановлен	чтение	Выкл	Вкл
7	Зарезервирован	чтение	Выкл	Вкл
8–15	Текущий шаг профиля программного задатчика	чтение	Декодируется как байт (шестнадцатеричный)	

Включение программного задатчика

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

Средства управления программным задатчиком

Четыре основных управляющих воздействия на программный задатчик реализуются с помощью битов 0 - 3 слова параметров сигнала наклон/выдержка в таблице параметров контура. **Direct**SOFT непосредственно управляет этими битами, используя диалог параметров программного задатчика. Однако в ходе выполнения программы эти биты должны управляться программно. Рекомендуем использовать команды 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 бита слова используется команда Set (Установить) Bit-of-word.

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

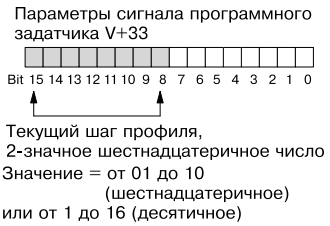
- **Старт**. Переход 0 - 1 запустит программный задатчик. Процессор должен быть в рабочем режиме, а контур может быть в ручном или автоматическом режиме. Если профиль не прерывается командой Остановка или Толчок, он нормально завершается.
- **Остановка.** Переход 0 - 1 остановит программный задатчик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

Мониторинг профиля сигнала наклон/выдержка

Отслеживать состояние профиля сигнала программного задатчика можно с помощью других битов слова параметров пилообразного сигнала V=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 V+33 параметров пилообразного сигнала. 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 **Direct**SOFT, который автоматически выполняет проверку диапазонов.

Тестирование профиля сигнала программного задатчика

Перед использованием профиля сигнала программного задатчика для управления процессом рекомендуется выполнить его проверку. Это несложно, так как программный задатчик будет работать, даже когда контур находится в ручном режиме. Сберечь время поможет окно Просмотр ПИД-регулятора (PID View) пакета **Direct**SOFT, так как в нем профиль сигнала выводится на экран. Не забудьте выбрать достаточно медленную временную развертку, чтобы полностью вывести пары участков сигнала наклон/выдержка 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 V+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 **Direct**SOFT 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 V+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 значение SP.
- Слишком низкое значение верхнего предела SP в ячейке V+27 таблицы контура.
- Проверьте свою программу, чтобы убедиться, что она ничего не записывает в ячейку SP (V+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 успешно считывать аналоговое значение и записывать его в ячейку V+03 таблицы контура. Убедитесь, что аналоговый модуль генерирует значение, и что программа работает.

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 (V+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 DirectSOFT, работают правильно, но эти же значения, записываемые программно, работают неправильно.

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

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 велика задержка передачи. Это означает, что PV медленно реагирует на изменения управляющего выхода. Причиной этого может быть слишком большое «расстояние» между исполнительным механизмом и датчиком PV, или способность исполнительного механизма передавать энергию процессу недостаточна.
- Может существовать возмущение процесса, которое контур не может преодолеть. Убедитесь, что PV относительно устойчиво при постоянном SP.

Библиография

Fundamentals of Process Control Theory, Second Edition
(Основы теории управления процессами, второе издание
Автор: Paul W. Murrill
Издатель: Instrument Society of America
ISBN 1- 55617- 297- 4
PID Controllers: Theory, Design, and Tuning, 2nd Edition
(ПИД-контроллеры. Теория, проектирование и настройка, 2-е издание)
Авторы: K. Astrom и T Hagglund
Издатель: Instrument Society of America
ISBN 1- 55617- 516- 7
Process / Industrial Instruments & Controls Handbook, Fourth Edition
Справочник по технологическим/промышленным инструментам и средствам управления, четвертое издание
Автор (главный редактор): Douglas M. Considine
Издатель: McGraw- Hill, Inc.
ISBN 0- 07- 012445- 0
Process Control, Third Edition
Instrument Engineer's Handbook
(Управление процессами, третье издание
Справочник инженера по инструментальным средствам)
Автор (главный редактор): Bela G. Liptak
Издатель: Chilton
ISBN 0- 8019- 8242- 1
Application Concepts of Process Control
(Концепции применения управления процессами)
Автор: Paul W. Murrill
Издатель: Instrument Society of America
ISBN 1- 55617- 080- 7
Fundamentals of Temperature, Pressure, and Flow Measurements, Third edition
(Основы измерений температуры, давления и потоков, третье издание)
Автор: Robert P. Benedict
Издатель: John Wiley and Sons
ISBN 0- 471- 89383- 8
pH Measurement and Control, Second Edition
(Управление и измерение pH, второе издание)
Автор: Gregory K. McMillan
Издатель: Instrument Society of America
ISBN 1- 55617- 483- 7
Process Measurement and Analysis, Third Edition
(Измерение и анализ процессов, третье издание)
Instrument Engineer's Handbook
Автор (главный редактор): Bela G. Liptak
Издатель: Chilton
ISBN 0- 8019- 8197- 2

Словарь терминов ПИД-регулирования

Автоматический режим	Режим работы контура, в котором контур выполняет расчет ПИД-алгоритма и изменяет управляющий выход.
Фиксация смещения	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	В позиционной форме уравнения ПИД-регулятора это сумма значений интегральной составляющей и начального значения управляющего 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 как выход другого контура. Каскадные контуры связаны соотношением главный/подчиненный и работают совместно, в конечном счете, управляя одной PV.
Каскадный режим	Режим работы контура, в котором контур получает значение SP как выход 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	Разность значений SP и PV, Рассоглосование = SP - PV
Зона нечувствительности отклонения	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	Метод оптимизации управляющей реакции контура на известное изменение уставки или возмущение, влияние которых на составляющую смещения поддается количественному определению.
Управляющий выход	Численный результат решения уравнения ПИД-регулятора, выдаваемый контуром с целью достижения нулевого текущего рассоглосования.
Дифференциальный коэффициент усиления	Константа, определяющая величину дифференциальной составляющей ПИД-регулятора, соответствующую текущему отклонению.
Интегральный коэффициент усиления	Константа, определяющая величину интегральной составляющей ПИД-регулятора, соответствующую текущему отклонению.
Главный контур	В каскадном управлении это контур, генерирующий уставки 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 SP от главного контура.

Дискретное управление Простой метод управления процессом с помощью включения/выключения подачи энергии в систему. Масса процесса усредняет влияние включения/выключения для относительно гладкой PV. Непрерывный выход DL450 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 в ходе процесса, которое влияет на качество окончательного продукта, которое требует контроля и управления.

Пропорциональный коэффициент усиления 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 рассогласования PV Программируемый аварийный сигнал, сравнивающий разность значений SP и PV с пороговым значением рассогласования.

Программный задатчик Набор значений SP, называемый профилем, генерируемых в реальном времени при каждом расчете контура. Профиль состоит из последовательности пар участков наклон/выдержка, заметно упрощающих задачу программирования ПЛК для генерации подобных последовательностей SP.

Скорость(Rate) Также называемая дифференциатором, составляющая скорости отвечает за изменения ошибочной составляющей.

Удаленная уставка Ячейка, из которой контур считывает свою уставку, если он является подчиненным контуром в каскадной топологии контуров.

Интегратор(Reset) Интегральная составляющая добавляет каждую дискретную ошибку к предыдущей, поддерживая вычисление текущей суммы, называемой смещением.

Взвинчивание(выбег) интегральной составляющей Условие, возникающее, когда контур не может найти равновесие, и постоянная ошибка приводит к чрезмерному увеличению 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 Интервал между вычислениями ПИД-регулятора. Метод, с помощью которого процессор управляет процессом, называется дискретным управлением, так как он выполняет расчеты только периодически.

Уставка (SP) Требуемое значение переменной процесса. Уставка (SP) или задание — это сигнал, подаваемый на вход контроллера контура при работе в режиме 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	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	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.
Скоростиой ПИД- алгоритм	Управляющий выход рассчитывается так, чтобы представлять изменение (скорость) PV, чтобы PV стала равной SP.

Обслуживание и поиск неисправностей

9

В этой главе...

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

Обслуживание технических средств

Нормальное обслуживание

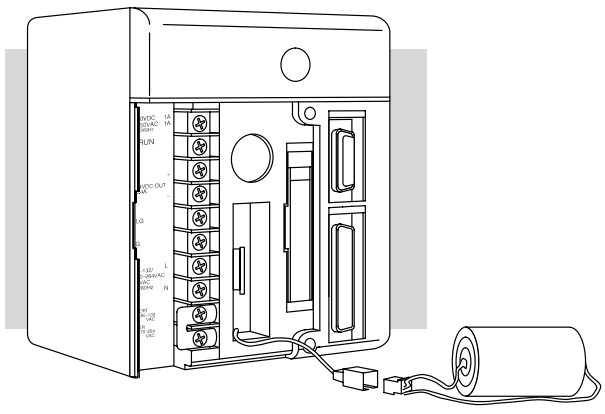
Хорошей практикой при обслуживании технических средств являются регулярные проверки вашего ПЛК и системы управления, которые включают следующие объекты контроля:

- Температура воздуха - Проверка температуры окружающей атмосферы в шкафу управления с тем, чтобы пределы рабочих температур любого компонента не были превышены.
- Воздушный фильтр - Если шкаф управления оснащен воздушным фильтром, то периодически очищайте его, а при необходимости заменяйте.
- Резервная батарея памяти - Убедитесь, что резервная батарея памяти процессора полностью не разрядилась. В процессоре используется один и тот же индикатор низкого напряжения на батарее процессора и на батареи картриджа памяти, если он применяется. Убедитесь, что Вы точно определили, какая из этих батарей требует замены.
- Предохранители или прерыватели - Убедитесь, что все предохранители и прерыватели находятся в исправном состоянии.
- Вентиляционные отверстия модуля DL405 - Убедитесь, что все вентиляционные отверстия процессора и всех модулей ввода/вывода чистые, а воздух для охлаждения проходит свободно.

Замена батареи процессора

Батареи Процессора используются для сохранения программную V-память и системных параметров. Предполагаемый срок службы батареи - пять лет.

ПРИМЕЧАНИЕ: Перед заменой вашей батареи процессора скопируйте V-память и системные параметры. Можно сохранить V-память и системные параметры либо на картридже памяти, либо на магнитной ленте, либо с помощью DirectSOFT, сохраняя программы на жестком/флоппи диске на персональном компьютере.
Чтобы предотвратить потерю информации в памяти, батарея процессора может заменяться при включенном питании системы. Если процессор был выключен, вы должны включить питание процессора, по крайней мере, за 5 секунд до замены батареи. Это делается для полной зарядки конденсатора, используемого для поддержания уровня напряжения, достаточного для сохранения памяти.



Замените батарею на новую D3-D4-BATT

Для замены батареи процессора необходимо:

- вынуть батарею из фиксаторов батареи,
- освободить зажим двухпроводного разъема батареи,
- отсоединить разъем батареи.

Для установки батареи процессора необходимо:

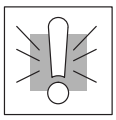
- соединить разъем батареи (маркированный) так, чтобы совпали красные провода;
- мягко вставить разъем батареи до щелчка зажима;
- вставить батарею в фиксаторы (заподлицо с углублением под батарею);
- отметить дату замены батареи.

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

Замена батареи картриджа памяти ОЗУ КМОП

Батарея картриджа памяти ОЗУ КМОП используется для сохранения содержимого ОЗУ КМОП картриджа памяти на время отключения питания. Предполагаемый срок службы этой батареи - три года.

ПРИМЕЧАНИЕ: Для дополнительной надежности Вы можете сохранить содержимое картриджа памяти на другом картридже памяти, магнитной ленте или на компьютерном диске, чтобы избежать потери содержимого картриджа памяти, когда батарея удалена. Однако картридж памяти имеет встроенный конденсатор для сохранения памяти в течение нескольких минут на время замены батареи. Если система была отключена, вы должны включить питание процессор, по крайней мере, на 5 секунд до замены батареи. Это делается для полной зарядки конденсатора, используемого для напряжения, необходимого для сохранения памяти.



ПРЕДУПРЕЖДЕНИЕ: Никогда не удаляйте картридж памяти из процессора при включенном питании. Если картридж памяти снимается при включенном питании, то он может стать ненадежным. Не пытайтесь перезарядить батарею и не располагайте ее вблизи огня. Батарея может взорваться или выделить опасные вещества.

Чтобы снять батарею картриджа памяти ОЗУ КМОП, необходимо:

1. Отключить питание процессора.

2. Вынуть картридж памяти из процессора, мягко надавливая на рычаг, пока картридж не освободится, затем вытяните его за рычаг.

3. Удалите винт крепления крышки.

4. Поднимите верхнюю крышку (поднимайте под небольшим углом, чтобы не поломать направляющую около разъема).

5. Возьмитесь за батарею и поднимите ее прямо вверх.

Чтобы установить батарею картриджа памяти ОЗУ КМОП, необходимо:

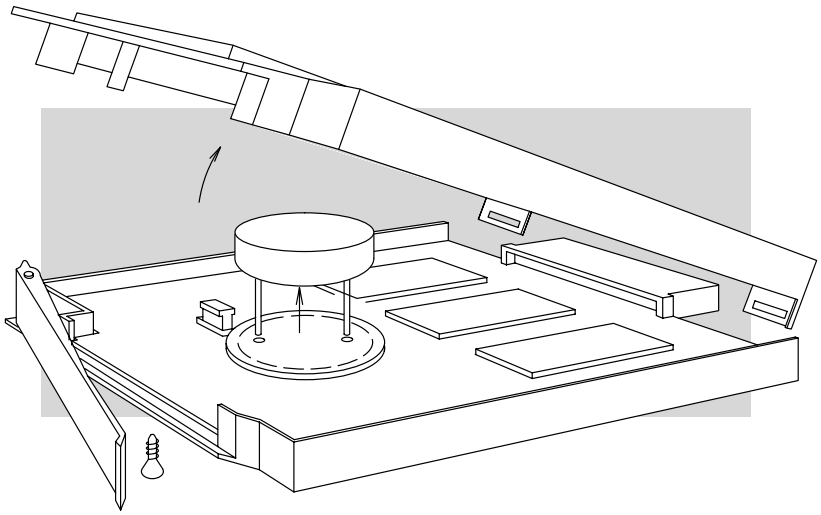
1. Совместить батарею с отверстиями в плате.

2. Прижать батарею так, чтобы она была заподлицо с платой.

3. Поместить крышку обратно в модуль (под небольшим углом, чтобы зафиксировать направляющие около разъема).

4. Установить прижимной винт крышки.

5. Отметить дату смены батареи.



Диагностика

Диагностика

DL405 выполняет свыше 90 программ диагностики в каждом цикле сканирования процессора. Диагностика разработана для обнаружения отказов в локальном каркасе процессора и локальных каркасах расширения. Ошибки определяются в следующих устройствах: процессорах, блоках расширения, вводе/выводе, каркасах, коммуникационных модулях картриджах памяти и в батареях. Существует два основные классы ошибок: критические и исправимые.

Критические ошибки (Fatal Error)

Критические ошибки - это ошибки, которые находит процессор и 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 он не войдет в РАБОЧИЙ режим до тех пор, пока ошибка не будет исправлена.

Примеры критических ошибок:

- Отказ источника питания каркаса процессора.
- Ошибка четности или сбой процессора.
- Определенные программные ошибки.

Исправимые ошибки (Non-fatal Error)

Исправимые ошибки - это такие ошибки, которые процессор помечает как требующие внимания. Они не могут вызвать переход процессора из РАБОЧЕГО режима в ПРОГРАММНЫЙ режим, также как препятствовать переходу процессора в РАБОЧИЙ режим. Имеются специальные реле, которые позволяют прикладной программе определить, имела ли место исправимая ошибка. Из прикладной программы можно затем правильно отключить систему или при необходимости вывести ПЛК из РАБОЧЕГО режима.

Примеров исправимых ошибок:

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

Получение диагностической информации

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

- Процессор автоматически заносит коды ошибок и все сообщения об ОШИБКАХ в две отдельные таблицы, которые можно просмотреть с помощью ручного программатора или DirectSOFT™.
- На ручной программатор выводятся номера ошибок и их краткое описание.
- С помощью DirectSOFT™ выводятся номера ошибок и сообщения об ошибках.
- В приложении B данного руководства приведен полный перечень сообщений об ошибках, упорядоченных по их номеру.

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

В следующих двух таблицах указаны конкретные ячейки памяти, соответствующие определенному типу сообщений об ошибках. Таблица специальных реле включает указатели состояний, которые могут использоваться при программировании. Обратитесь к приложению D для получения более детального описания этих специальных реле.

Класс ошибок	Категория ошибок	V-память для диагностики
Коммуникации	Номер каркаса / Номер слота	V7746
Коммуникации	Диагностический код ошибки	V7747
Определяемая пользователем	Код ошибки дефектного сообщения	V7751
Конфигурация ввода/вывода	Код идентификатора текущего модуля	V7752
Конфигурация ввода/вывода	Код идентификатора текущего модуля	V7753
Конфигурация ввода/вывода	Номер каркаса / Номер слота	V7754
Критическая	Код ошибки	V7755
Более существенная	Код ошибки	V7756
Менее существенная	Код ошибки	V7757
Модуль	Номер каркаса / Номер слота	V7760
Модуль	Код ошибки	V7762
Грамматическая	Адрес	V7763
Грамматическая	Код ошибки	V7764

Ячейки V-памяти, соответствующие кодам ошибок

Таблица специальных реле включает также индикаторы состояния, которые могут указывать ошибки. Обратитесь к приложению D за более детальной информацией по специальным реле.

Пусковое реле и реле реального времени		Реле состояния процессора		Реле текущего контроля системы		Реле состояния аккумулятора		Реле текущего контроля коммуникаций	
SP0	Реле первого сканировании	SP11	Реле принудительного рабочего режима	SP40	Реле критической ошибки	SP60	Реле значения меньше, чем	SP120	Реле занятости модуля, Слот 0
SP1	Всегда ВКЛЮЧЕННОЕ реле	SP12	Реле рабочего режима	SP41	Реле предупреждения	SP61	Реле значения равно	SP121	Реле ошибки связи, Слот 0
SP3	Реле на 1 минуту	SP13	Реле режима TEST-RUN	SP42	Реле низкого напряжения на батарее	SP62	Реле значения больше, чем	SP122	Реле занятости модуля, Слот 1
SP4	Реле на 1 секунду	SP14	Реле прерывания 1	SP44	Реле ошибки программной памяти	SP63	Реле нулевого значения	SP123	Реле ошибки связи, Слот 1
SP5	Реле на 100 миллисекунд	SP15	Реле режима TEST-PROGRAM	SP45	Реле ошибки ввода/вывода	SP64	Реле переноса части при вычитании	SP124	Реле занятости модуля, Слот 2
SP6	Реле на 50 миллисекунд	SP16	Реле режима TERM-PROGRAM	SP46	Реле ошибки связи	SP65	Реле переноса при вычитании	SP125	Реле ошибки связи, Слот 2
SP7	Реле чередующегося сканирования	SP17	Реле режима FORCED- STOP (принудительного останова)	SP47	Реле ошибки в конфигурации ввода/вывода	SP66	Реле переноса части	SP126	Реле занятости модуля, Слот 3
		SP21	Реле прерывания 2	SP50	Реле неправильной команды	SP67	Реле переноса	SP127	Реле ошибки связи, Слот 3
		SP22	Реле разрешения прерывания	SP51	Реле превышения времени математикой	SP70	Реле знака	SP130	Реле занятости модуля, Слот 4
		SP25	Реле блокирования батареи процессора	SP52	Реле синтаксической ошибки	SP71	Ошибка ссылки на указатель	SP131	Реле ошибки связи, Слот 4
		SP26	Реле запрета обновления ввода/вывода (DL440)	SP53	Реле ошибки в указателе таблицы/ математике	SP73	Реле переполнения	SP132	Реле занятости модуля, Слот 5
		SP27	Реле запрета селективного обновления ввода/вывода (DL440)	SP54	Ошибка связи	SP75	Реле ошибки в данных	SP133	Реле ошибки связи, Слот 5
		SP30	Реле состояния DIP-переключателя 1	SP56	Реле переполнения в табличной команде	SP76	Реле нулевой загрузки	SP134	Реле занятости модуля, Слот 6
		SP31	Реле состояния DIP-переключателя 2					SP135	Реле ошибки связи, Слот 6
		SP32	Реле состояния DIP-переключателя 3					SP136	Реле занятости модуля, Слот 7
		SP33	Реле состояния DIP-переключателя 4					SP137	Реле ошибки связи, Слот 7

Коды модулей ввода/вывода

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

Код (16-ричный)	Тип компонента	Код (16-ричный)	Тип компонента
01	Процессор	21	Вход на 8 точек
02	Блок расширения	28	Выход на 16 точек, аналоговый выход серий FL
03	Каркас ввода/вывода	2B	Вход на 16 точек, аналоговый вход серий FL, прерывание
11	DCM, все модули со-процесора™	30	Выход на 32 точки, аналоговый выход серий DL
12	Удаленное ведущее устройство, секционное ведущее устройство	3F	Вход на 32 точки, аналоговый вход серий DL
18	Высокоскоростной счетчик, магнитный импульсный вход	7F	Непредусмотренный тип
20	Выход на 8 точек	FF	Модуль не обнаружен

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

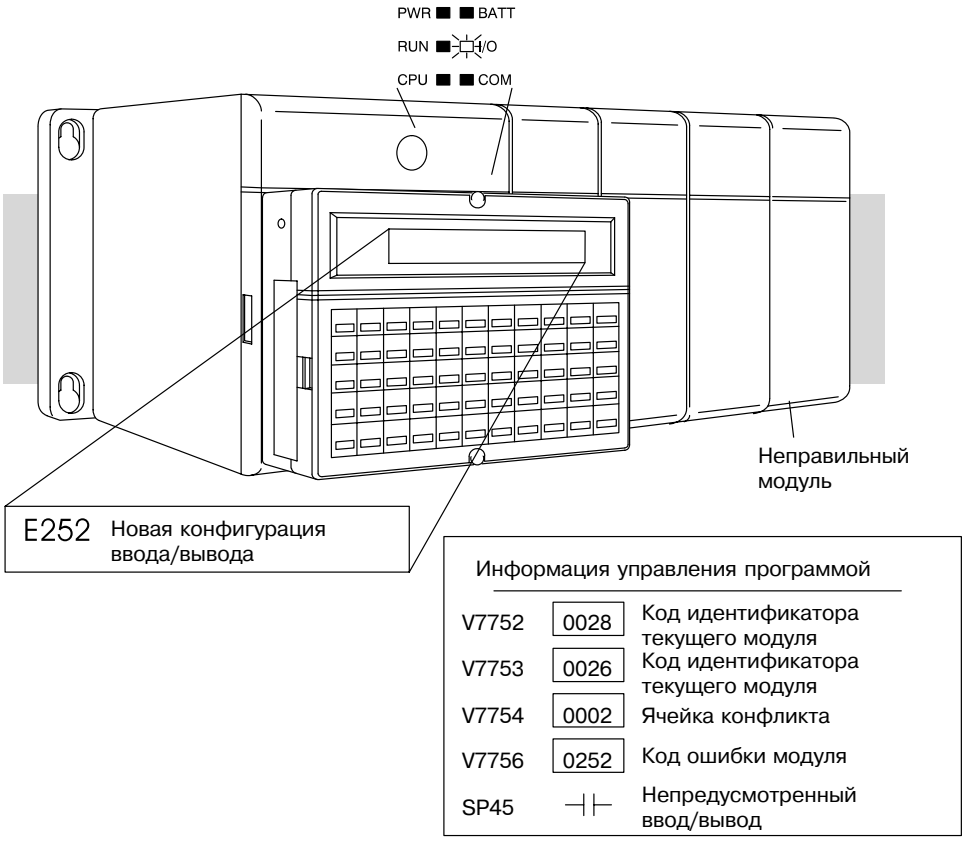


Таблица сообщений об ошибках

X	√	√
430	440	450

Процессоры DL440 и DL450 автоматически регистрируют коды любых системных ошибок, при этом вы можете создать любые специальные сообщения в вашей прикладной программе при использовании команд FAULT (См. главу 5 с подробным описанием команды FAULT). Процессор регистрирует код ошибки, дату и время, когда произошла ошибка. Имеются две отдельные таблицы, в которых хранится эта информация.

- Таблица системных ошибок - в этой таблице хранится до 32 ошибок.
- Таблица сообщений об ошибках - в этой таблице хранится до 16 сообщений.

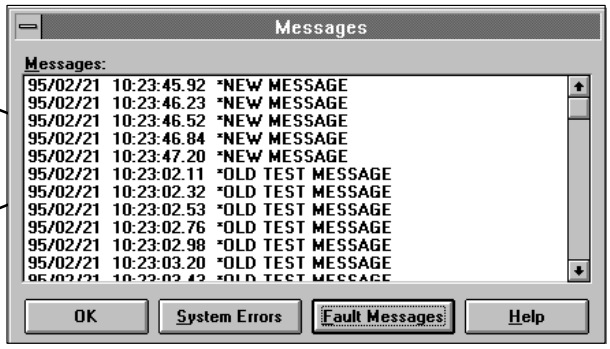
Когда возникает ошибка или сообщение, то оно помещается в первую доступную ячейку таблицы. Поэтому самое последнее сообщение об ошибке может не находиться в первой строке таблицы. Если таблица полностью заполнена при появлении новой ошибки, то самая старая ошибка выталкивается (стирается) из таблицы, а новая ошибка заносится в эту строку.

На следующем рисунке показан пример таблицы сообщений с ошибками, полученный с помощью DirectSOFT. Вы можете получить доступ к таблице кодов ошибок и к таблице сообщений об ошибках из DirectSOFT с помощью под- меню Диагностика ПЛК. Более подробную информацию по доступу к этим данным можно получить в руководстве по DirectSOFT.

Пример сообщений об ошибках

Последнее сообщение заносится сюда, а не в верхнюю строку таблицы

Следующее сообщение будет записано в этой строке, в ней сейчас находится самое раннее сообщение



Вы не можете просмотреть эту таблицу целиком на ручном программаторе. Сообщения об ошибках автоматически появляются на экране ручного программатора по мере их возникновения. Сообщение будет находиться на экране столько времени, сколько выполняется команда FAULT. На следующих примерах показывается, как использовать ручной программатор и функцию AUX 5C, чтобы просмотреть коды ошибок. Самый последний код ошибки или сообщение всегда выводятся. Вы можете использовать клавиши PREV или NXT, чтобы просмотреть путем прокрутки эти сообщения.

Используйте AUX 5C для просмотра таблиц

CLR AUX 5 SHFT C ENT

```
AUX 5C SHOW ERR/MSG
      ERR OR MSG
```

Используйте клавиши со стрелками для выбора ошибок или сообщений

SHFT → ENT

```
AUX 5C SHOW ERR/MSG
      ERR OR MSG
```

Пример представления сообщения на экране

```
PUMP 3 FAILED
04/22/93 17:30:00
```

Коды системных ошибок

X	√	√
430	440	450

Журнал регистрации системных ошибок содержит 32 последние обнаруженные ошибки. Ошибки, заносимые в журнал ошибок, составляют только часть всех сообщений об ошибках, которые вырабатываются системами DL405.

Следующие ошибки фиксируются в журнале регистрации системных ошибок при первом или повторном их появлении.

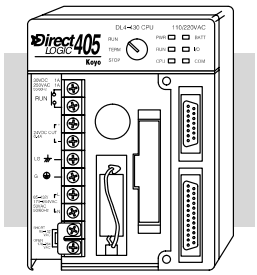
Код ошибки	Описание	Код ошибки	Описание
E003	Превышение времени программным обеспечением	E155	Сбой оперативной памяти (ОЗУ)
E004	Неправильная команда (ошибка четности ОЗУ в процессоре)	E201	Снятие клеммника блока
E041	Низкое напряжение на батарее процессора	E202	Отсутствие модуля ввода/вывода
E043	Низкое напряжение на батарее картриджа памяти	E203	Перегорание предохранителя
E099	Превышена программная память	E206	Отказ источника питания на 24 В пользователя
E101	Отсутствие картриджа памяти	E250	Отказ коммуникации в цепи ввода/вывода
E104	Сбой при записи	E251	Ошибка четности ввода/вывода
E151	Неправильная команда	E252	Новая конфигурация ввода/вывода

Следующие ошибки заносятся в журнал регистрации системных ошибок, если они имеют место, когда процессор пытается перейти в РАБОЧИЙ режим.

Код ошибки	Описание	Код ошибки	Описание
E401	Отсутствие оператора END	E431	Неправильный адрес ISG/SG
E402	Отсутствие оператора LBL	E432	Неправильный адрес перехода (GOTO)
E403	Отсутствие оператора RET	E433	Неправильный адрес SBR
E404	Отсутствие оператора FOR	E434	Неправильный адрес RTC
E405	Отсутствие оператора NEXT	E435	Неправильный адрес RT
E406	Отсутствие оператора IRT	E436	Неправильный адрес INT
E412	SBR/LBL > 64	E437	Неправильный адрес IRTC
E413	FOR/ NEXT> 64	E438	Неправильный адрес IRT
E421	Дублированная ссылка на стадию	E440	Неправильный адрес данных
E422	Дублированная ссылка на SBR/ LBL	E441	ACON/NCON
E423	Вложенные циклы		

Индикаторы состояния процессора

На передней панели процессоров DL405 расположены индикаторы, с помощью которых вы можете диагностировать неисправности в системе. В приведенной ниже таблице приводится краткий перечень возможных неисправностей, соответствующих каждому состоянию индикатора. В последующем тексте дается подробное описание каждого индикатора.



Индикатор	Состояние	Возможные причины
PWR	OFF	1. Неправильное входное напряжение для выбранного рабочего режима, перемычка на клеммном блоке для выбора напряжения 110/220 В переменного тока неправильно установлена. 2. Внешний источник питания отключен или отсоединен (проверьте предохранители, прерыватели). 3. Неисправен источник питания/процессор. 4. В источнике питания другого компонента, например, модуля ввода/вывода имеется короткое замыкание. 5. Превышена мощность используемого процессора
RUN	OFF	1. Ошибка в программах процессора. 2. Переключатель находится в положении STOP.
CPU	ON	1. Действие электрических помех. 2. Неисправен процессор.
BATT	ON, или мерцание	1. Мерцание с частотой 2 Гц: низкое напряжение батареи процессора. 2. Мерцание с частотой 0.5 Гц: низкое напряжение батареи картриджа памяти (только DL440/ DL450). 3. Постоянно включен: Обе батареи (процессора и картриджа памяти) имеют низкое напряжение. 4. Отсутствует или отсоединена батарея процессора или картриджа памяти.
DIAG (DL450)	ON	1. Отказ внутренней диагностики процессора. 2. Локальная шина на системной плате имеет ошибку связи.
I/O	ON	1. Отказ модуля ввода/вывода. 2. Отказ внешнего источника питания. 3. Ошибка в конфигурации. 4. Отказ блока расширения каркаса.
COM (DL430/ DL440)	ON	1. Неправильная настройка порта устройства. 2. Ошибка в кабельной разводке. 3. Неисправность в заземлении. 4. Электрические помехи. 5. Отказ порта устройства.
TXD (DL450)	OFF	1. Процессор не передает данные на вторичные порты (порты 1, 2 и 3) из-за программной ошибки. 2. Процессор не находится в рабочем режиме.
RXD (DL450)	OFF	1. Внешнее устройство не передает данные во вторичные порты процессора. 2. Коммуникационный кабель неисправен или отсоединен.

Индикатор PWR

√	√	√
430	440	450

В общем случае существуют четыре основных причины, при которых светодиод состояния питания процессора или блока расширения (PWR) находится в состоянии OFF (ВЫКЛЮЧЕН):

1. Напряжение питания процессора неправильное или установка перемычки диапазона напряжений (110/220 В) не соответствует входному напряжению.
2. Источник питания процессора неисправен.
3. Отключен источник питания другого компонента(ов). Возможно, модуль ввода/вывода в каркасе имеет короткое замыкание.
4. Превышена потребляемая мощность (при питании +5 В) процессора.

При неправильном напряжении на блоке питания ПЛК не сможет работать надлежащим образом, если вообще сможет работать. Используйте следующие указания для устранения неисправностей.

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

1. Сначала отключите внешнее питание системы.
2. Убедитесь, что все внешние прерыватели цепи и предохранители исправны.
3. Проверьте всю подводящую проводку на возможность ее обрыва. Если вы используете отдельный клеммный блок, проверьте правильность и сохранность подключений.
4. Если соединения правильны, включите питание системы и проверьте, удовлетворяет ли напряжение на входе блока питания процессора техническим требованиям. Если это напряжение неправильное, отключите систему и устраните неисправность.
5. Если все соединения правильны и питание на входе соответствует требуемым техническим условиям, то неисправен внутренний блок питания ПЛК.

При неправильном напряжении на блоке питания процессор не сможет работать надлежащим образом, если вообще сможет работать. Перед новой установкой версии процессора с питанием от переменного тока сначала проверьте установку перемычки на 110/220 В на клеммной колодке процессора. Если шунт выбора 110 В не установлен при использовании входного напряжения 110 В, то могут быть следующие последствия:

- Коммуникационные порты не будут функционировать.
- Процессор будет работать только тогда, когда модули не установлены.

Если шунт выбора 110 В установлен при использовании входного напряжения 220 В, то блок питания процессора может быть поврежден. Если это случится, то вам придется заменять процессор. Лучшим способом проверки повреждения блока питания процессора является его замена заведомо исправным процессором.

Если перемычка установлена правильно для используемых вами версий переменного или постоянного тока, то вам надо измерить напряжение на клеммнике, чтобы убедиться в том, что оно соответствует техническим требованиям процессора.

Существует вероятность того, что некоторый неисправный модуль или внешнее устройство, использующее системное питание 5В, может отключить источник питания. Это питание 5В может получаться с каркаса или с верхнего порта процессора. Чтобы проверить устройство, вызвавшее эту неисправность:

- Отключите питание процессора.
- Отсоедините от процессора все внешние устройства (например, коммуникационные кабели).
- Снова подайте питание в систему.

Если источник питания работает нормально, то у вас либо короткое замыкание в устройстве, либо в кабеле. Если источник питания работает ненормально, то определение модуля, вызвавшего эту неисправность, проводится в соответствии со следующими шагами:

- Отключите питание процессора.
- Выньте процессор из каркаса, оставляя подключенным шнур питания.
- Снова включите питание процессора.

Если светодиод PWR будет нормально работать, то неисправность наиболее вероятна в одном из модулей в локальном каркасе. Для того чтобы выделить этот модуль удаляйте по очереди по одному модулю, пока индикатор PWR не отключится. Перед поиском неисправного модуля вам необходимо установить процессор снова в каркас и выполнить следующую процедуру:

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

Причиной неисправности могут быть согнутые контакты разъема каркаса для этого модуля, поэтому проверьте разъем. Напоминаем, что превышение мощности является наиболее часто встречающейся ошибкой, при которой индикатор PWR либо не будет включаться, либо будет включаться периодически.

Проблемы превышения мощности обычно скорее возникают при запуске системы, чем при длительной ее работе. Если возникают какие-либо сомнения, то следует повторно проверить факт превышения мощности.

ПРЕДУПРЕЖДЕНИЕ: Если мощность превышена, то ПЛК может произвести сброс и перезагрузку системы. Если у вас есть какие-либо сомнения в части превышения мощности, то проверьте нагрузку в этот момент. Превышение мощности может привести к непредсказуемым последствиям, к повреждению оборудования и нанесению вреда персоналу. Проверьте потребляемую модулями мощность в выбранном вами каркасе. Соответствующие таблицы вы можете найти в главе 4 ”Проектирование и конфигурирование системы”.

Индикатор RUN

Если процессор невозможно перевести в рабочий режим (индикатор RUN выключен), то обычно имеется ошибка в прикладной программе, если только в самом процессоре нет критической ошибки. Однако при появлении критической ошибки светодиод процессора должен быть включен.

√	√	√
430	440	450

- Если Вы пытаетесь войти в рабочий режим с помощью программирующего устройства, а процессор в рабочий режим не переходит, то сначала убедитесь, что переключатель режимов находится в положении TERM, затем попытайтесь снова войти в рабочий режим.
- Если Вы используете переключатель режимов, чтобы изменить режим на рабочий, а процессор не реагирует на это, вам необходимо подсоединить программирующее устройство, чтобы получить диагностику ошибки.

Оба программирующих устройства, ручной программатор и DirectSOFT, выдают сообщение об ошибке и в зависимости от вида ошибки могут также рекомендовать AUX функцию для дополнительной диагностики неисправности. 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

√	√	√
430	440	450

Если индикатор CPU включен, то в процессоре возникла критическая 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.

Индикатор BATT

√	√	√
430	440	450

Если индикатор BATT включен или мерцает, то либо одна, либо обе батареи процессора/картриджа памяти имеют низкое напряжение (2.5 В и меньше). DL430 не имеет картриджа памяти. Напряжение на батарее отслеживается непрерывно, даже когда питание подается в систему.

Состояние светодиода BATT	Условие ошибки
Мерцание с частотой 2 Гц Мерцание с частотой 0.5 Гц	Напряжение на батарее процессора низкое Напряжение на батарее картриджа памяти низкое
Включен постоянно	Напряжение на батареях процессора и картриджа памяти низкое

Методы замены батарей можно найти выше в данной главе в разделе "Обслуживание технических средств".

Индикатор DIAG

X	X	√
430	440	450

Индикатор диагностики нормально выключен. Если он включается, то это значит, что процессор обнаружил ошибку при выполнении программ диагностики. Ошибку при диагностике могут вызвать критические электрические помехи, поэтому сначала повторите цикл включения питания процессора. Если индикатор DIAG по-прежнему будет включен, то вероятно отказал процессор. Чтобы убедиться в этом замените его на заведомо исправный.

Индикатор I/O (вода/вывода)

√	√	√
430	440	450

Если этот индикатор включен, то обнаружена неисправность в цепи локального, расширенного или удаленного ввода/вывода. Любая из следующих неисправностей может стать причиной включения светодиода I/O.

- Перегоревший предохранитель внутри модуля ввода/вывода.
- Потеря контакта в клеммном блоке.
- Отказ источника питания 24 В постоянного тока.
- Отказ модуля или блока расширения.
- Проверка конфигурации ввода/вывода обнаружила изменение в конфигурации ввода/вывода.

Обнаружение ошибок для удаленного ввода/вывода рассматривается в Руководстве по удаленному и секционированному вводу/выводу.

Для помощи вам при дополнительной диагностике каждый модуль имеет светодиоды для указания наличия ошибок. Модули дискретного ввода/вывода, охватываемые данным Руководством, могут иметь сочетание следующих индикаторов ввода/вывода.

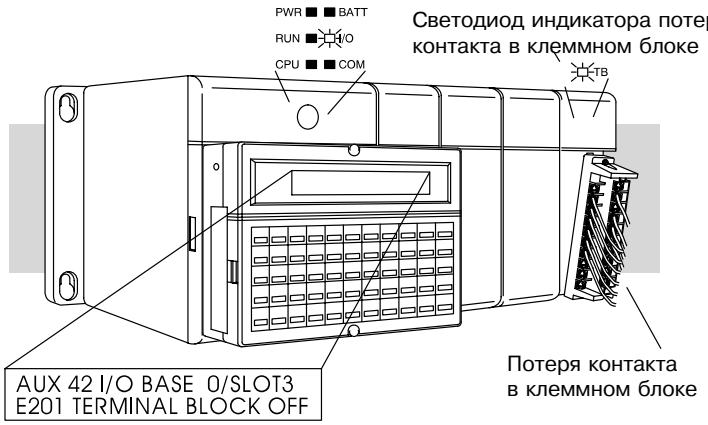
Индикатор	Условие ошибки
TB	Потеря контакта или отсутствие клеммного блока
24V	Внешний источник питания 24 В не обеспечивает нужного напряжения
FU	Предохранитель модуля перегорел (посмотрите спецификации модулей ввода/вывода, чтобы узнать заменяем ли предохранитель)

Многие другие специальные модули также имеют индикаторы. Руководства по этим модулям содержат информацию по индикаторам и по светодиодам состояний.

Если модули не дают какого-либо ключа для выяснения причины неисправности, запустите AUX 42 с ручного программатора или диагностику ввода/вывода в DirectSOFT. В обоих случаях Вы получите номер каркаса, номер слота и неисправность в модуле. Сразу после исправления неисправности индикаторы возвращаются в исходное состояние.

Ошибка ввода/вывода не вызывает переключение процессора из рабочего режима в программный, однако специальные реле (SP), доступные процессору, дают возможность прочитать эту ошибку в программе релейной логики. Прикладная программа может предусматривать необходимые действия, например, переход в программный режим или инициализацию правильного отключения. На следующем рисунке показан пример индикаторов отказов.

Внутренняя диагностика		
V7756	0201	Код идентификатора текущего модуля
V7760	0003	Местоположение ошибки (каркас 00, слот 03)
V7762	0201	Код ошибки модуля
SP45	⊣⊢	Ошибка ввода/вывода



Индикаторы TXD и RXD

430	440	450
√	√	X

Индикатор COM в процессорах DL430 и DL440 включается, когда процессор обнаруживает ошибку связи в одном из двух коммуникационных портов, встроенных в процессор. Наиболее частыми причинами такой ошибки чвляются:

- Кабель отсоединен.
- В кабеле имеется обрыв провода или он неправильно подключен.
- Кабель неправильно заделан или заземлен.
- Устройство, подсоединенное к порту, посылает неправильные данные.
- Между двумя устройствами существует разность потенциалов заземления.
- Электрические помехи вызывают периодические ошибки.
- В процессоре плохой коммуникационный порт, процессор должен быть заменен

Если возникает ошибка, то индикатор включается и остается включенным пока сеанс связь не будет успешно установлена. Если кабель и его соединения в порядке, попытайтесь повторить цикл включения питания устройств на обоих концах коммуникационного кабеля.

Индикатор COM

430	440	450
X	X	√

Индикация TXD и RXD в процессоре DL450 работает также как одноименные светодиодами индикаторов в модемах. Индикаторы TXD и/или RXD включаются, когда процессор передает или соответственно получает данные. Если индикатор(ы) не включаются, когда Вы предполагаете наличие обмена данными, то это - неисправность.

Индикаторы TXD и RXD включаются, когда данные передаются по любому из четырех портов DL450. Поэтому когда DirectSOFT или ручной программатор, или интерфейс оператора, например DV-1000, подключены к процессору, то TXD и RXD включены постоянно. Если Вы хотите обнаружить обмены, инициированные самой программой релейной логики, то следует отсоединить устройства для программирования или интерфейс оператора. В этом случае подсоединенным остается только кабель для коммуникаций, которые Вы отлаживаете.

Поиск неисправностей в модулях ввода/вывода

Возможные причины

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

- Ошибка конфигурации ввода/вывода с модулями, например, аналогового ввода/вывода, высокоскоростного счета, специализированных коммуникаций и т. д.
- Перегоревший предохранитель в модуле ввода/вывода, в вашей машине или шкафу.
- Плохой контакт в клеммном блоке.
- Отказ источника питания на 24 В постоянного тока
- Отказ входных или выходных точек.

Несколько быстрых шагов

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

- В выходных модулях нельзя автоматически обнаружить выходные точки с коротким замыканием или с обрывом. Если вы полагаете, что одна или несколько точек выходного модуля неисправны, то измерьте перепад напряжений между общим проводом и предполагаемой точкой. Напоминаем, что при использовании Цифрового Вольтметра необходимо рассматривать ток утечки из выходного устройства, такого как триодный тиристор или транзистор. Отключенная точка может дать такой же ток, как включенная точка при отключенной нагрузке.
- Индикаторы состояния точек ввода/вывода в модулях являются логическими индикаторами. Это значит, что светодиод, указывающий состояние "включено" или "выключено", отражает состояние логической части ввода/вывода. Индикаторы состояния на выходном модуле могут нормально функционировать, в то время как реальное выходное устройство (например, триодный тиристор или транзистор и др.) может быть неисправно. Что касается входных модулей, то, если светодиод индикатора включен, входные цепи должны работать правильно. При проверке правильного функционирования, убедитесь, что светодиод выключается, когда входной сигнал снимается.
- Ток утечки может стать проблемой при подсоединении полевых устройств к модулям ввода/вывода. Могут генерироваться ложные входные сигналы, когда ток утечки выходного устройства большой, достаточный для включения подсоединенного входного устройства. Для того чтобы это устранить, установите 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 активная нагрузка для вашего приложения правильно рассчитана.
- Индикатор перегорания предохранителя в выходном модуле будет показывать неисправность только в случае, если выходная точка соединена с нагрузкой и эта точка включена. Этот индикатор работает на основе измерения падения напряжения на предохранителе, поэтому должно быть напряжение, приложенное к предохранителю, и нагрузка, приложенная к выходу, чтобы создать этот перепад напряжения, который может быть зафиксирован модулем.
- 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.

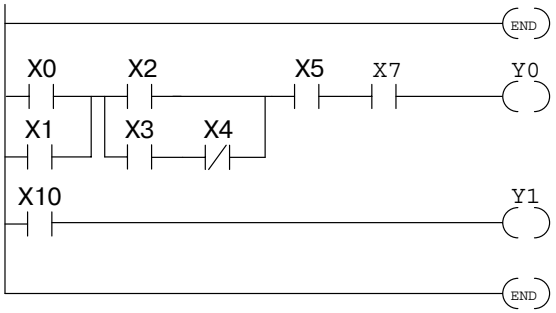
Тестирование выходных точек

В процессорах серий DL405 выходные точки могут находиться в состоянии "включено" или "выключено", но Вы не можете воздействовать на эти точки, чтобы заменить решение релейной логики. Если вы хотите провести независимую от прикладной программы проверку ввода/вывода, используйте следующую процедуру:

Шаг	Действие
1	Установите переключатель процессора в положение TERM
2	Используйте ручной программатор или DirectSOFT для оперативной связи с ПЛК.
3	Измените режим на программный.
4	Перейдите в адрес 0.
5	Вставьте по адресу 0 оператор END (При этом программа будет выполняться только до адреса 0 и не сможет включать или выключать точки ввода/вывода).
6	Измените режим на Рабочий, используя ручной программатор или DirectSOFT. Мы должны сделать это, чтобы разрешить изменение выходов.
7	Используйте ручной программатор для включения или выключения точек, которые вы хотите протестировать.
8	После завершения тестирования точек ввода/вывода удалите оператор END в адресе 0.

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

Работа с клавишами ручного программатора при тестировании выходной точки



Вставьте оператор END в начало программы. Это заблокирует выполнение оставшейся части программы.

При пустом экране дисплея нажмите следующие клавиши

Y(OUT) 0 BIT ST

```
        Y  10   Y   0
16P MON □□□□□□□□□□□□□□□□
```

Включите (или отключите) выход

Y(OUT) 0 SHFT ON

Y0 сейчас включено

```
        Y  10   Y   0
16P MON □□□□□□□□□□□□□□■□
```

Поиск и устранение помех

Проблемы электрических помех

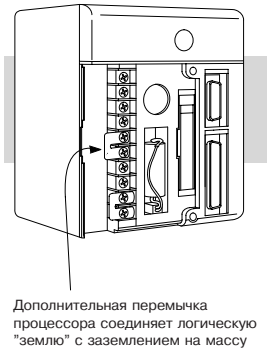
Помехи являются одной из самых трудных проблем при диагностике. Электрические помехи могут вводиться в систему разными способами, их можно разделить на две категории, введенные и излученные.

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

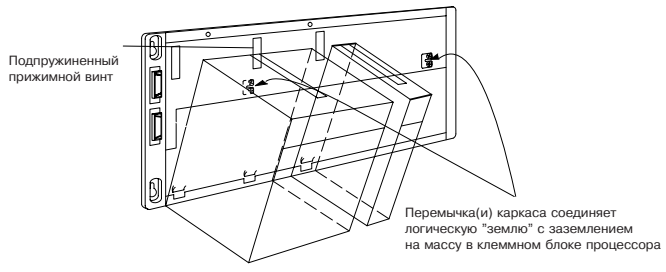
Уменьшение электрических помех

Следующие подсказки помогут уменьшить электрические помехи до уровня, который не отражается на работе системы.

- Большинство проблем с помехами возникает из-за неправильного заземления системы. Хорошее заземление может оказаться единственным эффективным способом устранения помех. Если заземление недостаточно, установите заземляющий стержень по возможности ближе к системе. Обеспечьте, чтобы все провода заземления подсоединялись к одной точке ”земля”, Чтобы не было ”гирляндных” цепей - из одного устройства в другое. Заземлите металлическое ограждение системы. Свободный провод - это большая антенна для приема помех в систему, поэтому вы должны затянуть все соединения в вашей системе. Свободный провод заземления более чувствителен 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 процессора или подсистемы 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.
- Для повышения помехозащищенности Вы можете дополнительно установить шунт заводского производства между логической ”землей” (LG) и заземлением на массу (G) в клеммном блоке процессора, как показано на рисунке слева.
- В редких случаях Вам может понадобиться изоляция логической ”земли” от заземления на массу. Для этого имеется простая перемычка в каркасе на 4 слота и две перемычки в каркасах на 6 и 8 слотов. Для такой изоляции нужно удалить эту перемычку(ки), как показано на рисунке ниже.



Дополнительная перемычка процессора соединяет логическую "землю" с заземлением на массу



Запуск машин и поиск ошибок в программах

Процессоры DL405 предоставляют несколько возможностей для отладки вашей программы до и после запуска производственных машин. В данном разделе рассматриваются следующие полезные для вас темы.

- Проверка синтаксиса программ
- Проверка дублированных ссылок
- Режимы тестирования
- Специальные команды
- Редактирование в рабочем режиме (RUN TIME EDIT)
- Специальные команды.

Проверка синтаксиса

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

Используйте AUX 21 для проверки синтаксиса

CLR AUX 2 1 ENT ENT

```
AUX 21 CHECK PROGRAM
1:SYN 2:DUP REF
```

Выберите проверку синтаксиса

1 ENT (Это может занять минуту или около этого)

```
BUSY
```

Появится одна из двух картинок на экране

Представление ошибки (пример)

```
$ 8 E401 MISSING END
TMRA   T 002   K00050
```

(показывает ячейку с ошибкой)

Изображение в случае отсутствия синтаксической ошибки

```
NO SYNTAX ERROR
?
```

Если вы получили ошибку, обратитесь к разделу Коды Ошибок с полным списком кодов программных ошибок. Исправьте ошибку и продолжайте проверку синтаксиса, пока не появится сообщение NO SYNTAX ERROR (синтаксических ошибок нет).

Проверка дублированных ссылок

Вы можете проверить также многократное использование одной и той же выходной обмотки. Оба устройства для программирования предоставляют возможность проверки этого условия. Например, вы можете использовать вспомогательную функцию AUX 21, CHECK PROGRAMM, для проверки дублированных ссылок с ручного программатора, либо вы можете воспользоваться опцией меню Диагностика ПЛК в DirectSOFT. Эта проверка позволяет найти разнообразные программные ошибки. На следующем примере показывается, как выполнить проверку дублированных ссылок с ручного программатора.

Используйте AUX 21 для проверки синтаксиса

AUX 2 1 ENT ENT

```
AUX 21 CHECK PROGRAM
1:SYN 2:DUP REF
```

Выберите проверку синтаксиса

2 ENT (Это может занять минуту или около этого)

```
BUSY
```

Появится одна из двух картинок на экране

Представление ошибки (пример)

```
$ 12 E471 DUP COIL REF
OUT    Y 0000
```

(показывает ячейку с ошибкой)

Изображение в случае отсутствия синтаксической ошибки

```
NO DUP REFS
?
```

Если вы получили ошибку дублированной ссылки исправьте ошибку и продолжайте проверку дублированных ссылок, пока таких ссылок не будет найдено.

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

Режимы TEST-PGM и TEST-RUN

В режиме тестирования процессор может начать работу в режиме TEST-PGM, перейти в режим TEST-RUN, отработать заданное число циклов сканирования и вернуться в режим TEST-PGM. Вы можете задать от 1 до 65 525 циклов сканирования. Вы можете выбрать режим тестирования либо с помощью ручного программатора (используя AUX 12), либо из DirectSOFT с помощью опции меню Режимы.

Если на ручном программаторе вы первый раз выбираете режим тестирования, то фактический режим, в который вы входите, зависит от режима работы процессора в момент запроса. Если процессор находится в рабочем режиме, то Вы переходите в режим TEST-RUN. Если - в программном режиме, то переходите в режим TEST-PGM. DirectSOFT более гибок при выборе режимов с помощью различных опций меню. В следующем примере показывается, как вы можете пользоваться ручным программатором при выборе режимов тестирования.

Используйте AUX 12 , чтобы войти в режим тестирования

AUX 1 2 ENT ENT

```
MODE = TEST-PGM
```

Определить число сканирований

CLR 1 SHFT TEST

(Процессор выполнит сканирования и вернется в режиме TEST-PGM)

```
NO. OF SCANS?
```

Переключиться в режим TEST-PGM

CLR 2 SHFT TEST

ENT (подтвердить возврат в режим TEST-PGM)

```
STOP SCAN?
```

Переключиться в режим TEST-RUN

CLR 3 SHFT TEST

ENT (подтвердить возврат в режим TEST-RUN)

```
START SCAN?
```

Вы можете получить некоторые преимущества, используя ручной программатор в режиме тестирования.

- На ручном программаторе отображение состояний более детальное.
- Вы можете дать возможность процессору сохранить состояния выходов.

Отображения на экране в режиме тестировании: При некоторых командах отображение состояний в режиме TEST-RUN более детальное, чем в рабочем режиме. На следующей схеме показан пример отображения команды таймера в режиме TEST-RUN.

Рабочий режим

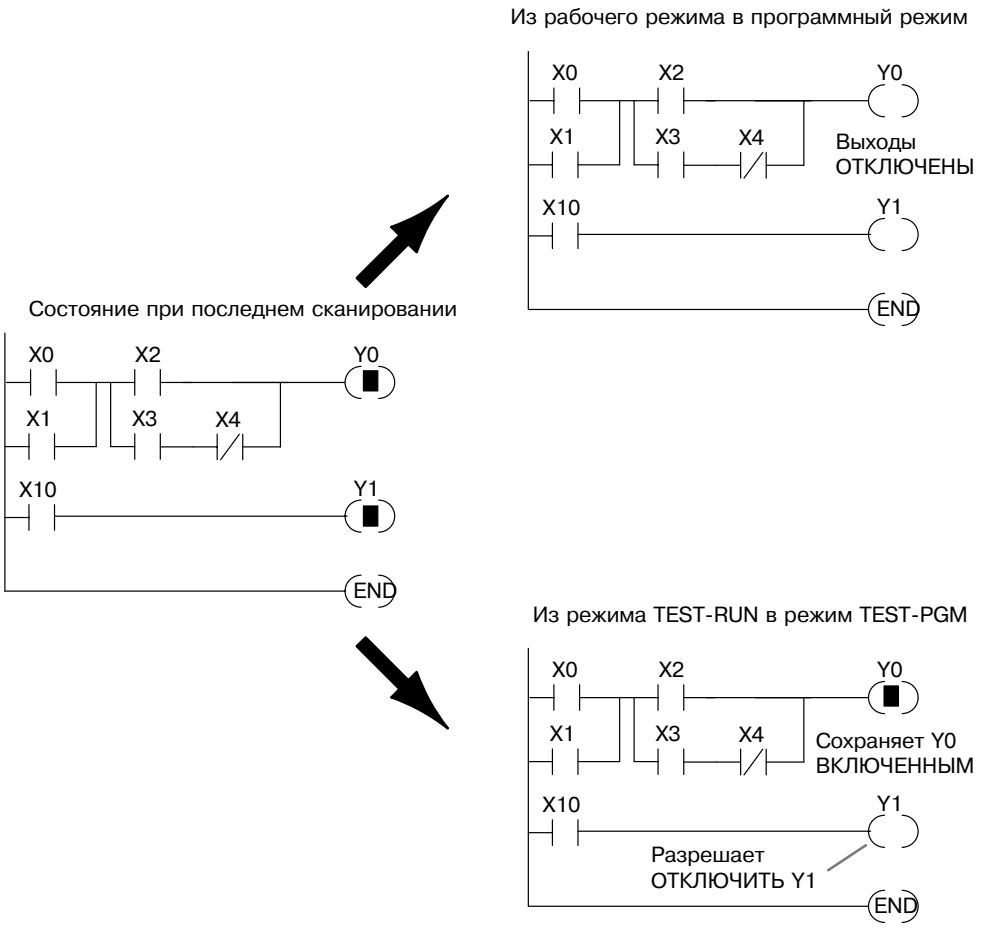
```
$    3                 ■
TMR    T 000   K0500
```

Режим TEST-RUN

```
$    3   T=510         ■
TMR    T 000   K0500
```

Текущее значение

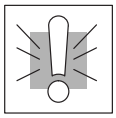
Сохранение выходных состояний: В обычном рабочем режиме выходы отключаются при переходе в программный режим. В режиме TEST-RUN при переходе в режим TEST-PGM вы можете выбрать для каждого конкретного выхода - либо отключение, либо сохранение его последнего состояния. Возможность сохранения выходных состояний весьма полезна, так как позволяет вам сохранить ключевые системные состояния точек ввода/вывода для последующей проверки. На следующей схеме показано различие между рабочим режимом и режимом TEST-RUN



Вы можете использовать AUX 58 на ручном программаторе для задания действия по каждому конкретному выходу.

Редактирование в рабочем режиме

X	√	√
430	440	450



Процессоры DL440 DL440 позволяют вам 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 в новой команде содержится синтаксическая ошибка, 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 принимаются при редактировании в рабочем режиме. Поэтому, когда вы используете высокоскоростную операцию и критический вход включен, процессор может не заметить изменение.

Для редактирования программ в рабочем режиме Вы можете использовать ручной программатор или DirectSOFT. На следующих страницах приведен краткий пример того, как это делается с ручного программатора. Вы используете AUX 14 для редактирования программ в рабочем режиме. Вам уже было показано, как задавать различные AUX-функции, но AUX 14 имеет несколько отличий ее применения:

- Сразу после задания AUX 14 светодиод RUN на ручном программаторе начинает мерцать. Это указывает на то, что редактирование в рабочем режиме действует.
- Если непосредственно перед выбором AUX 14 н экране отображался адрес, то этот адрес автоматически появится. Поэтому желательно найти необходимый адрес или команду до выбора AUX 14 или задавать его после выбора AUX 14.

Выберите AUX 14, редактирование в рабочем режиме

AUX 1 4 ENT

```
AUX 1* OPERATING MODE
AUX 14 RUN TIME EDIT
```

Нажмите ENT для выбора AUX 14 и отображения адреса

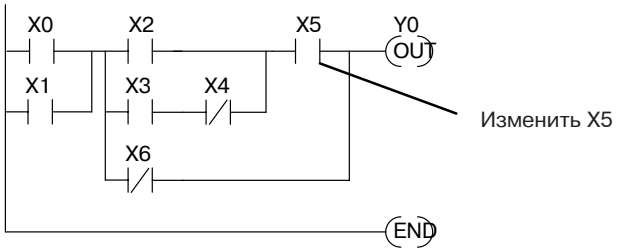
ENT

```
$xxxxx
STR    X 0001
```

Изменение команды в рабочем режиме

Как только Вы нашли команду, Вы можете очень легко изменить ее. В следующем примере показано, как изменить контакт X5 на X10.

Релейное представление



Идентифицируйте команду

АДРЕС	КОМАНДА	ОПИСАНИЕ
0	STR X0	Начинает ветвь 1 с X0
1	OR X1	Соединяет X1 параллельно X0
—	—	—
—	—	—
6	AND X5	Начинает ветвь 4 с X5
—	—	—
—	—	—
10	END	Конец программы

НАЙДИТЕ адрес

AND X(IN) 5 FIND

```
          SEARCHING
AND X5
```

```
$    6
AND    X 0005
```

Измените команду

AND X(IN) 10 ENT

```
$    6
WANT TO ALTER?
```

Нажмите CLR, чтобы прекратить редактирование или ENT, чтобы принять

ENT

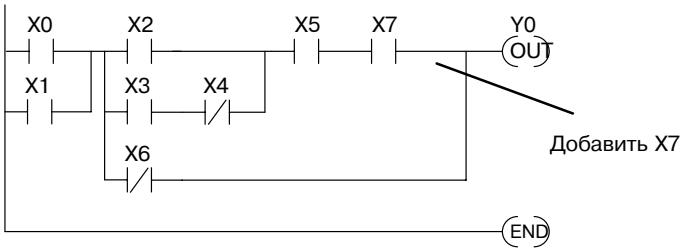
```
$    7
ORN    X 0006
```

(При нажатии ENT изменение принимается и высвечивается следующий адрес. При нажатии CLR отображается текущий адрес)

Вставить команду в рабочем режиме

Вставка команды в рабочем режиме работает почти также как и в программном режиме. Напоминаем, что INSERT добавляет команду перед командой, которая отображена, а оставшиеся адреса увеличиваются.

Релейное представление



Идентифицируйте команду

	АДРЕС	КОМАНДА	ОПИСАНИЕ
	0	STR X0	Начинает ветвь 1 с X0
	1	OR X1	Соединяет X1 параллельно X0
	—	—	—
Вставляется перед	6	AND X5	Начинает ветвь 4 с X5
		AND X7	Добавляет X7 последовательно с X5
	7	ORN X6	Соединяет X6 (NOT) параллельно
	—	—	—
	10	END	Конец программы

НАЙДИТЕ адрес

OR NOT X(IN) 6 FIND

```
            SEARCHING
ORN X6
```

```
$     7
ORN     X 0006
```

Вставьте новую команду

AND X(IN) 7 SHFT INS

```
$     7
WANT TO INSERT?
```

Нажмите CLR, чтобы прекратить редактирование или ENT, чтобы принять

ENT

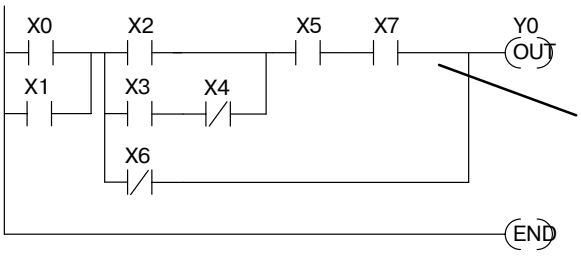
```
$     8
ORN     X 0006
```

(При нажатии ENT изменение принимается и высвечивается следующий адрес. При нажатии CLR отображается текущий адрес)

Удалить команду в рабочем режиме

Удаление команды в рабочем режиме работает почти также как и в программном режиме. Напоминаем, что в данной операции удаляется команда, которая в текущий момент отображена, а оставшиеся адреса уменьшаются.

Релейное представление



Идентифицируйте команду

АДРЕС	КОМАНДА	ОПИСАНИЕ
0	STR X0	Начинает ветвь 1 с X0
1	OR X1	Соединяет X1 параллельно X0
—	—	—
6	AND X5	Начинает ветвь 4 с X5
7	AND X7	Добавляет X7 последовательно с X5
—	—	—
11	END	Конец программы

Удалить

НАЙДИТЕ адрес

AND X(IN) 7 FIND

```
          SEARCHING
AND X7
```

```
$    7
AND    X 0007
```

Удалите команду

SHFT DEL

```
$    7
WANT TO DELETE?
```

Нажмите CLR, чтобы прекратить редактирование или ENT, чтобы принять

ENT

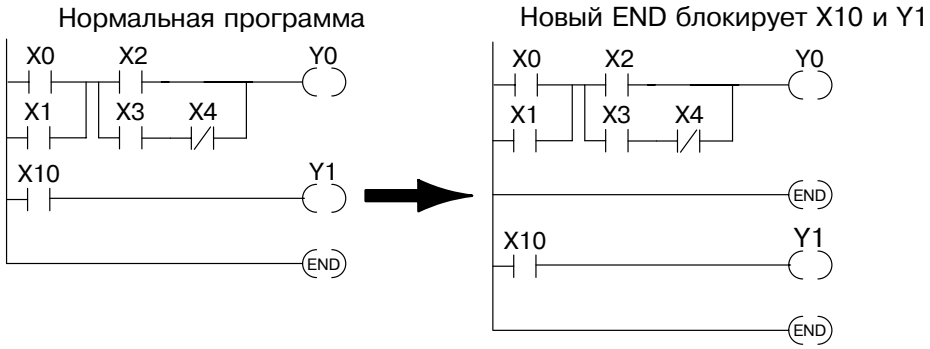
```
$    7
ORN    X 0006
```

(При нажатии ENT изменение принимается и высвечивается следующий адрес. При нажатии CLR отображается текущий адрес)

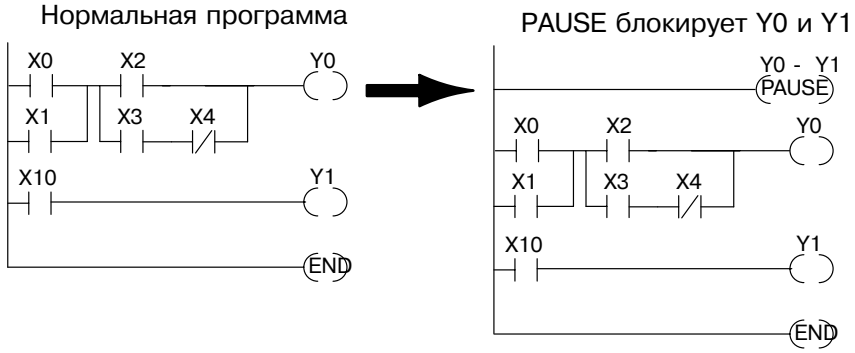
Специальные команды для отладки

Существует несколько команд, которые могут помочь вам при отладке программы во время операций после запуска машин: END, PAUSE, STOP и BREAK.

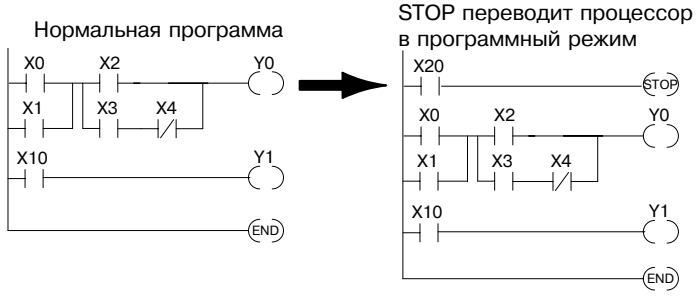
Команда END: Если вам необходимо быстро заблокировать часть программы, вставьте оператор END перед той частью, которую нужно заблокировать. Когда дроцессор встретит оператор END, он будет считать, что это конец программы. На следующей схеме приведен пример.



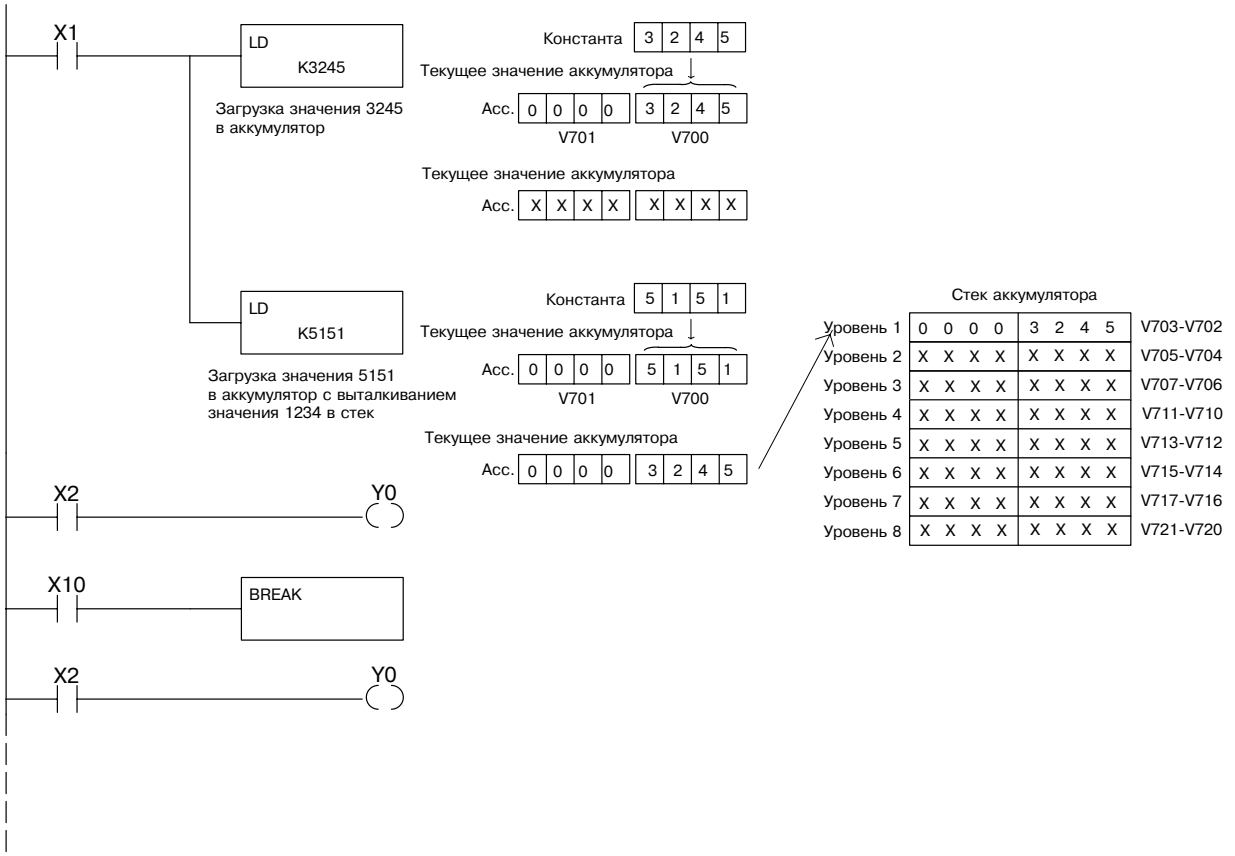
Команда PAUSE: Эта команда позволяет быстро установить режим, в котором входы (или другие логические переменные) функционируют при блокировании выбранных выходов. При этом регистр отображения выходов обновляется, но выходные состояния не записываются в модули. Например, вы можете сделать эту блокировку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



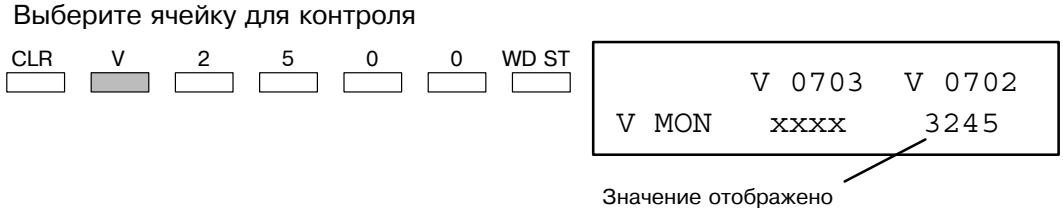
Команда STOP: Иногда после запуска машин вам необходимо быстро отключить все выходы и вернуться в программный режим. В дополнение к режимам тестирования вы можете использовать также команду STOP. При выполнении этой команды процессор автоматически выйдет из рабочего режима и перейдет в программный режим. В следующей программе показано, как это делается. Напоминаем, что все выходы в программном режиме отключаются.



Команда BREAK: Если у вас процессор DL440, то Вы можете использовать также команду BREAK для останова сканирования программ. Пока команда BREAK активна, сканирование остановлено процессор входит в специальный режим, TEST-HALT. Вы должны использовать либо ручной программатор, либо DirectSOFT для повторного запуска сканирования посредством перевода процессора обратно в рабочий режим. При возвращении процессора в рабочий режим сканирование возобновляется с точки прерывания. Например, если Вы используете некоторые команды обработки данных, может потребоваться приостановка сканирования и проверка аккумулятора и стека аккумулятора. На следующем рисунке показан пример.



В этом примере вход X10 запускает команду BREAK. В этой точке процессор останавливает сканирование программы. Сейчас Вы легко можете посмотреть, как команды программы влияют на аккумулятор или на стек аккумулятора. В следующем примере показано, как Вы можете использовать ручной программатор для проверки стека аккумулятора.



После того, как Вы используете ручной программатор для возвращения в рабочий режим, процессор начинает сканирование программы с цепочки, непосредственно следующей за BREAK.

Вспомогательные функции

A

В этой главе...

— Введение

— AUX 1* — Рабочие режимы

— AUX 2* — Операции в RLL

— AUX 3* — Операции с V-памятью

— AUX 4* — Конфигурирование ввода/вывода

— AUX 5* — Конфигурирование процессора

— AUX 6* — Конфигурирование ручного программатора

— AUX 7* — Операции с картриджем памяти

— AUX 8* — Операции с паролем

Введение

Что такое Вспомогательные функции ?

Многие задачи настройки процессора предполагают использование Вспомогательных (AUX) функций. AUX функции выполняют много различных операций, начиная от простого изменения рабочего режима и кончая копированием программ на картриджи памяти. Они разделены на категории, которые предназначены для различных системных ресурсов. Вы можете получить доступ к AUX функциям из DirectSOFT или с ручного программатора. Некоторые из этих AUX функций разработаны специально для ручного программатора, поэтому они не доступны для пакета DirectSOFT. Вы можете дополнительно к настоящему приложению использовать документацию по выбранному вами устройству для программирования.

AUX Функции и их описание		430	440	450	HPP
AUX 1* — Рабочие режимы					
11	Перейти в Рабочий Режим (RUN)	✔	✔	✔	-
12	Перейти в Режим Отладки (TEST)	✔	✔	✔	-
13	Перейти в Режим Программирования (PGM)	✔	✔	✔	-
14	Редактировать в Рабочем Режиме (Run Time Edit)	×	✔	✔	-
AUX 2* — Операции RLL					
21	Проверка Программы	✔	✔	✔	-
22	Изменение ссылки	×	✔	✔	-
23	Очистка памяти в диапазоне цепей	✔	✔	✔	-
24	Очистка памяти во всех цепях	✔	✔	✔	-
25	Выбрать Картридж памяти или Флэш-память	×	×	✔	-
26	Копировать содержимое Картриджа памяти во Флэш -память	×	×	✔	-
27	Копировать содержимое Флэш-памяти в Картридж памяти	×	×	✔	-
28	Сверить содержимое Флэш-памяти и Картридж памяти	×	×	✔	-
AUX 3* — Операции с V-Памятью					
31	Очистка V-памяти	✔	✔	✔	-
32	Стереть область V-памяти	✔	✔	✔	-
33	Найти значение в V-памяти	×	✔	✔	-
AUX 4* — Конфигурация ввода/вывода					
41	Показать конфигурацию ввода/вывода	✔	✔	✔	-
42	Диагностика ввода/вывода	✔	✔	✔	-
44	Проверка конфигурации ввода/вывода при включении питания	✔	✔	✔	-
45	Выбрать конфигурацию	✔	✔	✔	-
46	Конфигурировать ввод/вывод	×	✔	✔	-
47	Интеллектуальный ввод/вывод	✔	✔	✔	-

✔ — поддерживается

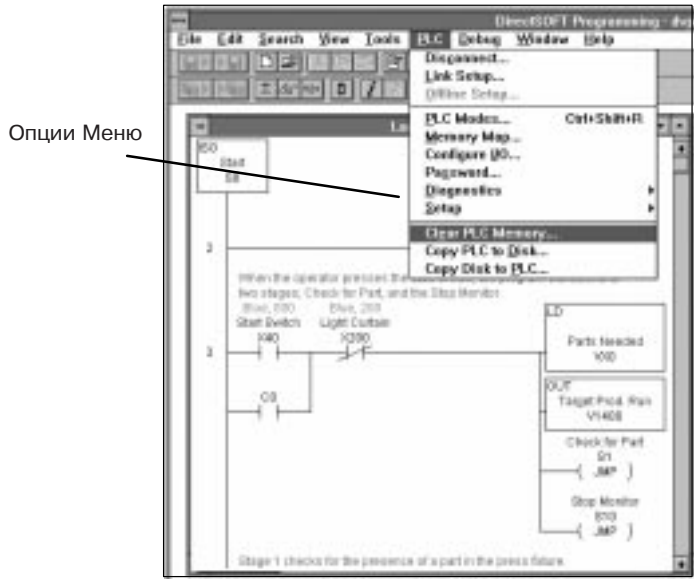
× — не поддерживается

– — не применяется

AUX Функции и их описание		430	440	450	HPP
AUX 5* — Конфигурация Процессора					
51	Изменить Имя программы	✔	✔	✔	-
52	Показать / Изменить Календарь	×	✔	✔	-
53	Показать Время сканирования	✔	✔	✔	-
54	Инициализировать Электронный блокнот	✔	✔	✔	-
55	Установить сторожевой таймер	✔	✔	✔	-
56	Конфигурировать последовательные порты	✔	✔	✔	-
57	Установить Области Сохранения	✔	✔	✔	-
58	Операции Тестирования	✔	✔	✔	-
5C	Показать архив ошибок	×	✔	✔	-
5D	Выбрать режим сканирования ПЛК	×	×	✔	-
AUX 6*— Конфигурация Ручного Программатора					
61	Показать номер версии	✔	✔	✔	
62	Включить/Отключить звуковой сигнал	×	×	×	✔
63	Отключить/включить подсветку индикации	×	×	×	✔
64	Выбрать режим работы с процессором /автономный режим	×	×	×	✔
65	Запустить самодиагностику	×	×	×	✔
AUX 7* — Операции с картриджем памяти					
71	Копировать память процессора в Картридж	✔	✔	✔	✔
72	Копировать память Картриджа в процессор	✔	✔	✔	✔
73	Сравнить память Картриджа с памятью процессора	✔	✔	✔	✔
74	Проверить очистку памяти на Картридже	×	×	×	✔
75	Очистить память Картриджа	×	×	×	✔
76	Показать тип Картриджа	✔	✔	✔	✔
77	Копировать магнитную ленту в Картридж	×	×	×	✔
78	Копировать Картриджа на магнитную ленту	×	×	×	✔
79	Сравнить магнитную ленту с Картриджем	×	×	×	✔
AUX 8* — Операции с паролем					
81	Изменить пароль	×	✔	✔	-
82	Разблокировать процессор	×	✔	✔	-
83	Заблокировать процессор	×	✔	✔	-

Доступ к AUX функциям через DirectSOFT

DirectSOFT предоставляет различные опции меню как в диалоговом, так и в автономном режимах 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 работы с меню ПЛК в DirectSOFT.



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. На следующем рисунке показано, как можно быстро получить доступ к AUX функциям с ручного программатора.

CLR AUX

```
AUX FUNCTION SELECTION
AUX 1* OPERATING MODE
```

Используйте клавиши NXT или PREV для просмотра меню

NXT

```
AUX FUNCTION SELECTION
AUX 2* RLL OPERATIONS
```

Нажмите клавишу ENT для выбора под-меню

ENT

```
AUX 2* RLL OPERATIONS
AUX 21 CHECK PROGRAM
```

Вы можете также ввести точный номер AUX функции для непосредственного выхода в под-меню.

Введите непосредственно номер AUX функции

AUX 2 1 ENT

```
AUX 2* RLL OPERATIONS
AUX 21 CHECK PROGRAM
```

AUX 1* — Рабочие режимы

AUX 11 Переход в рабочий режим

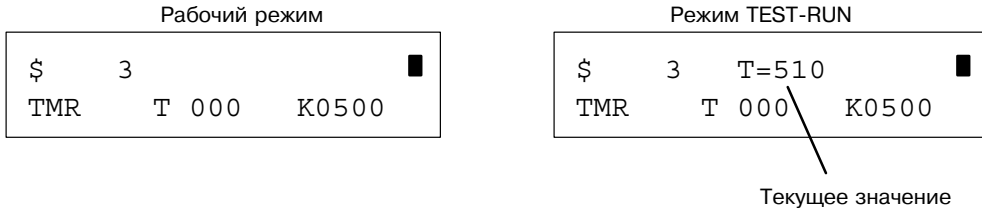
Вы можете использовать AUX 11 для перевода процессора в рабочий режим. В рабочем режиме процессор выполняет программы и обновляет модули ввода/вывода. Вы можете также изменять режим ПЛК из DirectSOFT, используя под-меню ПЛК/Режимы ПЛК.

AUX 12 Переход в режим тестирования

В режиме тестирования процессор может запустить режим TEST-PGM, войти в режим TEST-RUN, выполнить фиксированное число циклов сканирования (от 1 до 65535), затем вернуться в режим TEST-PGM. Вы также можете войти в режим тестирования из DirectSOFT, используя под-меню ПЛК/Режимы ПЛК. Используя режим тестирования, Вы получаете некоторые преимущества.

- На ручном программаторе состояние отображается более детально.
- Процессор имеет возможность сохранить состояние выходов.

Для некоторых команд на ручном программаторе состояние в режиме TEST-RUN отображается более детально, чем в рабочем режиме. На следующем рисунке показан пример отображения команды таймера в режиме TEST-RUN.



AUX 13 Переход в программный режим

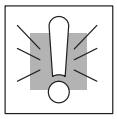
Вы можете использовать AUX 11 для перевода процессора в программный режим. В этом режиме Вы можете вводить или изменять программы. Процессор не выполняет программы и не обновляет выходные модули. Вы также можете войти в программный режим из DirectSOFT, используя под-меню ПЛК/Режимы ПЛК.

AUX 14 Редактирование в рабочем режиме

На процессоре DL440 или DL450 Вы можете редактировать программы в рабочем режиме и в режиме TEST-RUN с помощью AUX 14 или из DirectSOFT, используя под-меню ПЛК/Режимы ПЛК.

Большинство операций, которые Вы делаете в программном режиме, здесь также выполнимы. Например, можно использовать тот же самый метод поиск конкретной команды или адреса. Однако в рабочем режиме Вы не можете использовать функции Найти и Заменить.

Редактирование в рабочем режиме не является ”безударным”. Процессор сохраняет выходы по их последнему состоянию, пока принимается новая информации по программам. Если в новой программе будет найдена ошибка, то процессор сбросит все выходы и перейдет в программный режим.



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. Редактирование в рабочем режиме идеально подходит для внесения небольших изменений.

При больших изменениях в программе мы настоятельно рекомендуем переключить систему в программный режим, соблюдая те же предосторожности, как и при начальном запуске машины.

AUX 2* — Операции в RLL

AUX 21 Проверка программы

Как ручной программатор, так и DirectSOFT 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 обратитесь к приложению B с полным списком кодов ошибок в программах. После исправления ошибки продолжайте синтаксическую проверку, пока не появится сообщение ”NO SYNTAX ERROR” (синтаксических ошибок нет).

При проверке дублированных ссылок контролируется, не используете ли вы ссылку на одну и ту же выходную обмотку несколько раз. Следует заметить, что данная AUX функция будет определять одни и те же выходы даже тогда, когда они применяются с командой OROUT, где дублирование допустимо.

Данная AUX функция доступна в DirectSOFT из под-меню PLC Diagnostic (Диагностика ПЛК).

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 диапазона программной памяти

Очень часто при решении новых прикладных задач вам необходимо добавить или удалить часть существующих программ. При помощи AUX 23 вы можете выбрать и удалить часть программы. DirectSOFT 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 памяти процессора. Перед тем как вы вводите новую программу, вы всегда должны чистить программную память. Данная AUX функция доступна в DirectSOFT через под-меню ” PLC/ Clear PLC” (”Чистка ПЛК”).

AUX 25 Выбор картриджа памяти или Флэш-памяти

AUX 25 позволяет вам выбрать либо картридж памяти (необязательный), либо встроенную флэш-память процессора, как текущую рабочую память. ЗАМЕЧАНИЕ: Такие операции, как очистка диапазона программной памяти, очистка программной памяти, работают на выбранной текущей памяти устройства.

AUX 26 Копирование содержимого картриджа памяти во флэш-память

AUX 26 копирует все содержимое картриджа памяти в встроенную флэш-память процессора. Все данные во флэш-памяти будут затерты.

AUX 27 Копирование содержимого флэш-памяти на картридж памяти

AUX 27 копирует все содержимое встроенной флэш-памяти процессора на картридж памяти. Все данные в картридже памяти будут затерты.

AUX 28 Верификация содержимого флэш-памяти и картриджа памяти

AUX 28 проверяет, какие данные во внутренней флэш-памяти процессора совпадают с содержимым картриджа памяти. Эта функция полезна, когда проверяемая программа обновлена, и вообще при управлении программой в процессе ее выполнения.

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 из DirectSOFT через под-меню PLC/Clear PLC ("Очистить ПЛК").

AUX 32 Очистка области V-памяти

Сразу после того, как Вы использовали AUX 23 для удаления части программы управления, Вы должны также использовать AUX 32 для очистки части V-памяти. DirectSOFT обеспечивает доступ к этой AUX функции через под -меню Tools/Memory Editor (Инструментарий/Редактор памяти).

AUX 33 Найти значение в V-памяти

AUX 33 дает возможность выполнить поиск в области ячеек V-памяти, чтобы найти конкретное значение. Например, Вы хотите найти значение 1234 в диапазоне V1400 - V2000. Данная AUX функция доступна также из DirectSOFT через под-меню Tools/Memory Editor (Инструментарий/Редактор памяти).

AUX 4* — Конфигурирование ввода/вывода

AUX 41 Показать конфигурацию ввода/вывода

Эта AUX функция позволяет вам вывести на экран текущую конфигурацию ввода/вывода. На ручном программаторе вы можете прокрутить каждый каркас и слот ввода/вывода, чтобы просмотреть всю конфигурацию. В конфигурации показывается тип модуля, установленного в каждом слоте. В DirectSOFT выдается та же информация, но гораздо удобнее для просмотра, так как вы можете просмотреть весь каркас на одном экране.

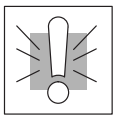
AUX 42 Диагностика ввода/вывода

Это одна из наиболее полезных AUX функций, имеющихся в системе DL405. Она покажет вам точное положение слота и каркаса любого модуля ввода/вывода, в котором возникла ошибка. Эта функция доступна также в DirectSOFT, вызывается из меню PLC Diagnostic (Диагностика ПЛК).

AUX 44 Проверка конфигурации при включении питания

При выборе этой функции вы можете быстро обнаружить все изменения, которые могли произойти в тот период, когда питание системы было отключено. Например, если кто-то поместил выходной модуль в слот, ранее предназначавшийся для входного модуля, то при проверке конфигурации это изменение немедленно будет выявлено.

Когда система при включении питания обнаруживает изменение в конфигурации ввода/вывод, то генерируется код ошибки E252 NEW I/O CONFIGURATION (НОВАЯ КОНФИГУРАЦИЯ ВВОДА/ВЫВОДА). Вы можете воспользоваться AUX 42 для определения точного положения каркаса и слота, в котором произошло это изменение.

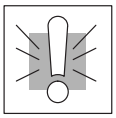


ПРЕДУПРЕЖДЕНИЕ: Вы всегда должны исправлять ошибки в конфигурации ввода/вывода перед переходом в рабочий режим. Неисправленные ошибки могут вызвать непредсказуемую работу аппаратуры, что может привести к риску нанесения вреда персоналу или к повреждению оборудования.

Эту функцию можно вызывать также из DirectSOFT через под-меню PLC/Setup (ПЛК/Настройка).

AUX 45 Выбрать конфигурации

Хотя процессор и находит автоматически изменения в конфигурации, вы можете установить новую конфигурацию ввода/вывода. Например, вы можете специально изменить модуль ввода/вывода, чтобы поработать с новой программой. Вы можете использовать AUX 45 для выбора новой конфигурации или сохранить существующую конфигурацию, сохраненную в памяти. Эту функцию можно вызывать также из DirectSOFT через под-меню PLC/Setup (ПЛК/Настройка).



ПРЕДУПРЕЖДЕНИЕ: Убедитесь в том, что выбранная конфигурация правильно работает с программами процессора. Всегда исправляйте ошибки в конфигурации ввода/вывода перед переходом в рабочий режим. Неисправленные ошибки могут вызвать непредсказуемую работу аппаратуры, что может привести к риску нанесения вреда персоналу или к повреждению оборудования.

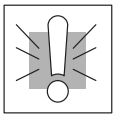
AUX 46 Сконфигурировать ввод/вывод

Процессор DL440 позволяет вам использовать функцию AUX 46 для ручного присвоения адресов ввода/вывода для любого или всех слотов ввода/вывода в локальном каркасе или каркасе расширения, хотя эта функция может никогда вам не понадобиться. Операции конфигурирования гораздо легче выполнять из DirectSOFT. Программный пакет предусматривает по-настоящему хороший экран, вызываемый из под-меню

PLC/Configure I/O (Конфигурирование ввода/вывода).

Эта функция полезна, если Вы имеете стандартную конфигурацию, которую иногда должны немного изменить, чтобы приспособить ее для специальных запросов. Например, Вы можете потребовать, чтобы два соседних входных модуля имели адреса, начинающиеся соответственно с X10 и X200.

При автоматическом конфигурировании адреса назначаются на границах 8 точек. При ручном конфигурировании предполагается, что все модули, по крайней мере, на 16 точек, поэтому Вы можете присваивать адреса, кратные 20 (восьмеричным). Например, X30 и Y50 не являются правильными начальными адресами. X20 и Y40 - примеры правильных начальных адресов при ручном конфигурировании. Это не означает, что при ручном конфигурировании Вы можете использовать только модули на 16 или 32 точки. Можете использовать модули на 8 точек, но из 16 назначаемых адресов 8 не будут использованы.



ПРЕДУПРЕЖДЕНИЕ: Когда Вы вручную конфигурируете слот ввода/вывода, то адресация других модулей может измениться. Это связано с тем, что DL405 не допускает повторных адресов ввода/вывода. Всегда исправляйте все ошибки в конфигурации ввода/вывода перед переводом процессора в рабочий режим. Неисправленные ошибки могут вызвать непредсказуемую работу аппаратуры, что может привести к риску нанесения вреда персоналу или к повреждению оборудования.

После ручного конфигурирования адресов для слотов ввода/вывода система автоматически сохранит эти значения даже после перезапуска системы. Вы можете убрать изменения ручного конфигурирования, просто выполнив автоматическое конфигурирование.

AUX 47 Оперативный контроль интеллектуального ввода/вывода

Эта AUX функция дает вам возможность оперативно контролировать совместно используемые данные ОЗУ, связанные со многими специальными модулями ввода/вывода. Например, Вы можете использовать эту AUX функцию для оперативного контроля данных, доступных в совместно используемом ОЗУ для высокоскоростного счетчика.

Эту функцию можно вызывать также из DirectSOFT через под-меню PLC/Setup/Global I/O (ПЛК/Настройка/глобальный ввод/вывод).

AUX 5* — Конфигурирование процессора

AUX 51 — Изменить имя программы

В системах DL405 используются имена программ, хранимых в картридже памяти или на магнитных лентах. Имена программ особенно полезны для магнитных лент, так как на них может храниться много программ. 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 из DirectSOFT через под-меню PLC/Setup (ПЛК/Настройка).

AUX 52 Показать/ изменить календарь

Процессор DL440 имеет функцию часов и календаря. Если вы пользуетесь ими, то вы можете с помощью ручного программатора и функции AUX 52 установить время и дату. Используются следующие форматы:

- Дата - Год, Месяц, Дата, День недели (0 - 6, с воскресенья по субботу)
- Время - формат 24 часа, Часы, Минуты, Секунды.

Вы можете использовать эту AUX функцию, чтобы изменить любой элемент даты и времени. Однако процессор не будет автоматически корректировать расхождение между датой и днем недели. Например, если вы изменили дату на 15-ое число месяца, 15-ое число является четвергом, то вы должны также изменить день недели (если только процессор уже не показывает дату как четверг).

Вы можете также выполнять эту операцию из DirectSOFT через под-меню PLC/Setup (ПЛК/Настройка).

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 DirectSOFT через под-меню PLC Diagnostic (Диагностика ПЛК).

AUX 54 Инициализировать системную оперативную память

Процессоры DL405 поддерживает системные параметры в области памяти, которую часто называют ”системной оперативной памятью”. В некоторых случаях вы можете вносить изменения в настройку системы, которая хранится в системной памяти. Например, если вы определяете область управляющих реле (CR) как сохраняемую, то эти изменения запоминаются..

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 из DirectSOFT через под-меню PLC/Setup (ПЛК/Настройка).

AUX 55 Установить сторожевой таймер

Процессоры DL405 имеют ”сторожевой” таймер, который применяется для отслеживания времени сканирования. Его значению по умолчанию, установленное в заводских условиях, равно 200 мс. Когда время сканирования превосходит установленный временной предел, процессор автоматически выходит из рабочего режима и переходит в программный режим. 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 DirectSOFT через под-меню PLC/Setup (ПЛК/Настройка).

AUX 56 Сетевой адрес процессора

Поскольку процессоры DL405 имеют встроенные порты DirectNET, вам необходимо использовать ручной программатор для установки сетевого адреса для порта и параметров связи порта. Параметрами настройки по умолчанию являются:

- Адрес станции "1"
- Шестнадцатеричный режим
- Контроль по нечетности

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 вы имеете нижний порт, который 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 DirectSOFT через подменю PLC/Setup (ПЛК/Настройка).

AUX 57 Установить области сохранения памяти

Процессоры DL405 обеспечивает по умолчанию определенные области сохранения памяти. Установленные по умолчанию области сохранения пригодны для многих приложений, но вы можете изменить их, если ваше приложение требуют дополнительных областей сохранения или вообще никаких областей сохранения. Областями сохранения по умолчанию являются

- Управляющие реле: C600-C737
- V- Память: V2000 - V7377
- Таймеры: По умолчанию не устанавливаются (хотя Вы можете сделать их сохраняемыми)
- Счетчики: CT0 - CT177
- Стадии: По умолчанию не устанавливаются (хотя Вы можете сделать их сохраняемыми).

Используйте AUX 57 для изменения областей сохранения. Вы можете также выполнять эту операцию из DirectSOFT через под-меню PLC/Setup (ПЛК/Настройка).

AUX 58 Операции тестирования

В нормальном рабочем режиме при переходе в программный режим все выходы сбрасываются. В режиме TEST-RUN вы можете установить каждый конкретный выход либо на сброс, либо на сохранение последнего состояния выхода при переходе в режим TEST-PGM. Способность сохранения состояний выходов чрезвычайно полезна, так как дает возможность сохранить для контроля ключевые системные параметры точек ввода/вывода.

Используйте AUX 58 для конфигурирования каждого отдельного выхода. Вы можете также выполнять эту операцию из DirectSOFT через под-меню PLC/Setup (ПЛК/Настройка).

AUX 5C Показать журнал ошибок

Процессоры DL440 и DL450 автоматически регистрирует коды любых системных ошибок и созданные вами с помощью команд FAULT сообщения по этим ошибкам. Процессор регистрирует код ошибки, дату и время, когда произошла ошибка, в двух отдельных таблицах:

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

Ниже показан пример таблицы сообщений об ошибках.

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

Вы можете использовать функцию AUX 5C для просмотра кодов и сообщений об ошибках. Вы можете также просмотреть эти ошибки и сообщения в DirectSOFT через под-меню PLC Diagnostic (Диагностика ПЛК).

AUX 5D Выбор режима сканирования ПЛК

Процессор DL450 имеет два режима сканирования программ: фиксированный и переменный. В фиксированном режиме время цикла сканирования задается вами (в миллисекундах). В переменном режиме процессор начинает каждое сканирование до полного завершения предыдущего цикла сканирования.

AUX 6* — Конфигурирование 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 программно - аппаратных средств процессора и ручного программатора. Эта информация (по процессору) доступна также в DirectSOFT через под-меню PLC Diagnostic (Диагностика ПЛК).

AUX 62 Включить/ отключить звуковой сигнал

Ручной программатор имеет устройство звуковой сигнализации для подтверждения нажатия клавиш. Это может раздражать людей, находящихся в офисе. Вы можете использовать вспомогательную функцию AUX 62 для отключения звуковой сигнализации.

AUX 63 Включить/ отключить подсветку

При необходимости Вы может использовать AUX 63 для отключения подсветки. Однако в большинстве случаев легче читать, если подсветка экрана включена.

AUX 64 Выбрать диалоговый/автономный режим

Вы может использовать AUX 64 для перевода ручного программатора в автономный режим работы. Ручной программатор не будет обмениваться данными с процессором, пока он не будет возвращен в диалоговый режим. Например, если выбран автономный режим, то Вы не можете иметь доступ к процессору.

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
- Порт для записи на магнитную ленту (требуется адаптер)
- Картридж памяти

AUX 7* — Операции с картриджем памяти

Переносимость областей памяти

Многие из указанных AUX функций позволяют вам копировать различные области памяти в процессор, картридж памяти и магнитную ленту и наоборот. В таблице ниже приведены области памяти, которые могут указываться.

Опция и тип памяти	Область памяти DL440	Область памяти DL430
1: PGM - Программная	$00000 - $07679 (7.5K программной памяти) $00000 - $015871 (15.5K программной памяти)	$00000 - $03583
2: V - V-память	$00000 - $37777	$00000 - $07777
3: SYS - Системная	Невыбираемые копии всех системных параметров	

AUX 71 Считать из процессора в картридж памяти

AUX 71 копирует информацию из памяти процессора в картридж памяти, установленный в ручном программаторе. Если картридж памяти отсутствует в ручном программаторе, Вы можете только изъять картридж памяти из процессора и установить его в ручном программаторе. Но если Вы сохранить процессор в рабочем состоянии, вам следует воспользоваться данной AUX функцией для копирования памяти процессора.

Вы можете копировать различные части памяти процессора на картридж памяти в соответствии с приведенной таблицей. Один картридж памяти не сможет поддерживать работу всей системы. Поэтом Вы должны иметь больше одного картриджа. Если это так, установите отдельно V-память в картридж памяти.

AUX 72 Записать картридж памяти в процессор

AUX 72 копирует информацию из картриджа памяти, установленном в ручном программаторе, в процессор. Если картридж памяти отсутствует в процессоре, Вы можете только снять картридж памяти из ручного программатора и установить его в процессоре. Вы можете копировать различные типы информации из памяти картриджа в соответствии с приведенной таблицей.

AUX 73 Сравнить память картриджа и процессора

AUX 73 сравнивает программы на ручном программаторе (на картридже памяти) с программами процессора. Вы можете сравнивать различные типы информации в соответствии с приведенной таблицей.

AUX 74 Проверить очистку картриджа

AUX 74 дает вам возможность проверить картридж памяти, чтобы убедиться, что она пустая. Эту функцию желательно использовать всякий раз, когда вы начинаете копировать программы в картридж памяти.

AUX 75 Очистить картридж памяти

AUX 75 позволяет вам удалить все данные из картриджа памяти. Это можно сделать с картриджами памяти ОЗУ и ЭППЗУ. Для очистки УФПЗУ необходимо использовать источник ультрафиолетового излучения.

AUX 76 Показать тип картриджа памяти

AUX 76 можно использовать для быстрого вывода на экран типа картриджа памяти, установленного как в процессоре, так и в ручном программаторе. Напоминаем, что существуют три типа картриджей памяти: ОЗУ КМОП, ЭППЗУ и УФПЗУ.

AUX 77 Записать с магнитной ленты на картридж памяти

Используйте AUX 77 для чтения программ с магнитной ленты. Перед началом этой процедуры убедитесь, что Вы знакомы с процедурой, определенной в Руководстве по ручному программатору. Эта процедура поможет вам обеспечить успешную передачу информации.

AUX 78 Записать с картридж памяти на магнитную ленту

Поскольку весьма удобно хранить много программ на одной кассете, очень важно иметь имя каждой программы. Вы уже знакомы с тем, как вводить имена программ процессора. Имена программ на кассете не должны повторяться. Например, имя процессора может быть PRESS1, а имя магнитной ленты STATION1.

Поскольку существует три области памяти процессора, которые могут передаваться, можно дать каждой из них отдельное имя программы. Например, можно использовать три программы для STATION1: STAT1PGM, STAT1V, STAT1SYS.

ПРИМЕЧАНИЕ: Напоминаем, что программы на магнитной ленте хранятся последовательно. Очень легко затереть существующие программы, если Вы не определили правильное позиционирование на магнитной ленте. Убедитесь перед началом данной операции, что Вы знакомы с процедурами, определенными в Руководстве по ручному программатору DL405.

AUX 79 Сравнить картридж памяти с магнитной лентой

Используйте AUX 79 для сравнения программ, находящимися на кассете с магнитной лентой, с программами процессора. Убедитесь перед началом данной операции, что Вы знакомы с процедурами, определенными в Руководстве по ручному программатору DL405.

AUX 8* — Операции с паролем

AUX 81 позволяет вам ввести особую меру защиты посредством введения пароля, с помощью которого предупреждаются несанкционированные машинные операции. Паролем должен быть 8-значный цифровой (0 -9) код. Введенный пароль может быть удален введением вместо него всех нулей (00000000). Это значение по умолчанию устанавливается в заводских условиях.

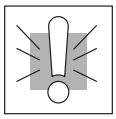
ПРИМЕЧАНИЕ: Процессор DL450 поддерживает многоуровневые пароли. Это дает возможность защитить с помощью пароля программы и в то же время не блокировать коммуникационный порт для интерфейсов оператора. Многоуровневый пароль можно ввести, если в пароле перед 7 цифровыми символами поставить прописную букву ”A” (например, A1234567).

Пароль хранится в картридже памяти. Если Вы устанавливаете картридж памяти в другой процессор или на ручной программатор, то защита с помощью пароля остается действующей.

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 изDirectSOFTTM через подменю PLC/Password (ПЛК/Пароль). В DirectSOFT 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. Вы также не можете стереть картридж памяти и начать все сначала.

AUX 82
Разблокировать процессор

AUX 82 может использоваться для разблокировки процессора, защищенного паролем. DirectSOFT автоматически попросит вас ввести пароль, если вы попытаетесь взаимодействовать с процессором, содержащим пароль.

AUX 83
Заблокировать процессор

AUX 83 может использоваться для блокировки процессора, который содержит пароль. При блокировке процессора вам необходимо вводить пароль, чтобы получить доступ. Напоминаем, что в DirectSOFT эта функция не является необходимой, так как процессор автоматически блокируется, как только вы выходите из программного пакета.

Коды ошибок DL405

B

В этой главе...

— Таблица кодов ошибок

Код ошибки	**Название**	**Описание**
E001	КРИТИЧЕСКАЯ ОШИБКА ПРОЦЕССОРА	Возможно вы устранили ошибку при повторном включении питания процессора. Если ошибка появилась вновь, замените процессор.
E003	ПРЕВЫШЕНИЕ ПРОГРАММАМИ УСТАНОВЛЕННОГО ВРЕМЕНИ	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 добавьте команды RSTWT в циклы и подпрограммы FOR NEXT или используйте AUX 55 для увеличения времени на сторожевом таймере.
E004	НЕПРАВИЛЬНАЯ КОМАНДА (DL440)	Прикладная программа по каким-то причинам изменена. SP44 включается, а код ошибки запоминается в V7755. Эта неисправность возможно вызвана электрическими помехами. Используйте AUX 21 для синтаксической проверки программы и исправьте ее, где это необходимо, или очистите память и перезагрузите программу. Устраните неисправности заземления. Если ошибка повторяется, замените процессор.
E041	СЛАБАЯ БАТАРЕЯ ПРОЦЕССОРА	Батарея процессора слабая, требуется ее замена. SP43 включается, а код ошибки записывается в V7757.
E042	БАТАРЕЯ ПРОЦЕССОРА ОТСУТСТВУЕТ (DL450)	Батарея процессора не установлена. SP43 включается, а код ошибки записывается в V7757.
E043	СЛАБАЯ БАТАРЕЯ ПРОЦЕССОРА КАРТРИДЖА ПАМЯТИ (DL440/ DL450)	Батарея картриджа памяти слабая, требуется ее замена. SP43 включается, а код ошибки записывается в V7757.
E044	БАТАРЕЯ КАРТРИДЖА ПАМЯТИ ОТСУТСТВУЕТ (DL450)	Батарея картриджа памяти не установлена. SP43 включается, а код ошибки записывается в V7757.
E099	ПРЕВЫШЕНИЕ ПРОГРАММНОЙ ПАМЯТИ	Эта ошибка возникает, когда длина скомпилированной программы превышает доступный объем ОЗУ процессора. SP52 включается, а код ошибки записывается в V7755. Уменьшите размер прикладной программы.
E101	ОТСУТСТВУЕТ КАРТРИДЖ ПАМЯТИ ПРОЦЕССОРА (DL440/ DL450)	Картридж памяти процессора отказал или отсутствует. SP44 включается, а код ошибки записывается в V7755. Установите или замените картридж памяти.

Код ошибки	Название	Описание
E104	ОТКАЗ ПРИ ЗАПИСИ (DL440/ DL450)	Запись в картридж памяти процессора оказалась неудачной. Картридж памяти может быть защищен от записи. Разберите и проверьте перемычку. Если ошибка повторяется замените картридж памяти.
E151	НЕПРАВИЛЬНАЯ КОМАНДА	В прикладной программе имеет место ошибка четности. SP44 включается, а код ошибки записывается в V7755. Возможно эта неисправность возникла из-за электрических помех.(исправьте любые неисправности в заземлении) Очистите память и снова загрузите программу. Если ошибка повторяется, замените картридж памяти или процессор.
E155	ОТКАЗ ОЗУ	В системном ОЗУ имеет место ошибка контрольной суммы. SP44 включается, а код ошибки записывается в V7755. Эта ошибка возможно появилась из-за слабой батареи, электрических помех или сбоя ОЗУ процессора. Очистите память и снова загрузите программу. Устраните неисправности в заземлении. Если ошибка повторяется, замените процессор.
E2**	ОТКАЗ МОДУЛЯ ВВОДА/ВЫВОДА	Отказ модуля ввода/вывода. Запустите AUX 42, чтобы определить фактическую ошибку.
E201	ОТСУТСТВУЕТ КЛЕММНЫЙ БЛОК	В клеммном блоке отсутствует контакт или он отсутствует в модуле ввода/вывода. SP45 включается, а код ошибки записывается в V7756
E202	ОТСУТСТВИЕ МОДУЛЯ ВВОДА/ВЫВОДА	Модуль ввода/вывода отказал при обмене данными с процессором или он отсутствует в каркасе. SP45 включается, а код ошибки записывается в V7756. Запустите AUX 42 для определения положение слота и каркаса модуля, по которому выдано сообщение об ошибке.
E203	ПЕРЕГОРЕЛ ПРЕДОХРАНИТЕЛЬ	Предохранитель перегорел в модуле ввода/вывода. SP45 включается, а код ошибки записывается в V7756. Запустите AUX 42 для определения положение слота и каркаса модуля, по которому выдано сообщение об ошибке.
E206	ОТКАЗАЛ ИСТОЧНИК ПИТАНИЯ 24 В ПОЛЬЗОВАТЕЛЯ	Отказал источник питания 24 В пользователя. SP45 включается, а код ошибки записывается в V7756. Запустите AUX 42 для определения положение слота и каркаса модуля, по которому выдано сообщение об ошибке.
E210	ОТКАЗ ПИТАНИЯ (DL440)	В линии сетевого питания, подающей напряжение на каркас, произошло кратковременное аварийное отключение питания.
E250	ОТКАЗ СВЯЗИ В ЦЕПИ ВВОДА/ВЫВОДА	В системе локального ввода/вывода произошел сбой. Неисправность может быть в источнике питания каркаса, в кабеле расширения или в блоке расширения ввода/вывода. SP45 включается, а код ошибки записывается в V7755. Запустите AUX 42 для определения положения слота и каркаса модуля, по которому выдано сообщение об ошибке.

Код ошибки	**Название**	**Описание**
E251	ОШИБКА ЧЕТНОСТИ ВВОДА/ВЫВОДА	Имеет место ошибка четности при обмене данными в коммуникационной цепи ввода/вывода.
E252	НОВАЯ КОНФИГУРАЦИЯ ВВОДА/ВЫВОДА	Эта ошибка выявляется, когда проводится автоматическая проверка конфигурации процессором, и когда фактическая конфигурация ввода/вывода изменена либо в результате перемещения модулей в каркасе, либо из-за изменения типа модулей в каркасе. Вы можете вернуть модули в первоначальное положение или к первоначальному типу, или запустить AUX 45 для установления новой конфигурации. SP47 включается, а код ошибки записывается в V7755.
E261	КОНФЛИКТ АДРЕСОВ ВВОДА/ВЫВОДА (DL440/ DL450)	При ручном конфигурировании ввода/вывода присвоены перекрывающиеся адреса. Исправьте присвоение адресов с помощью AUX 46. SP45 включается, а код ошибки записывается в V7755.
E262	ВНЕ ДИАПАЗОНА АДРЕСОВ ВВОДА/ВЫВОДА	Выход за диапазон адресов ввода/вывода может встретиться в прикладной программе. Исправьте в программе неправильный адрес. SP45 включается, а код ошибки записывается в V7755.
E263	СКОНФИГУРИРОВАННЫЕ АДРЕСА ВВОДА/ВЫВОДА ВНЕ ДИАПАЗОНА (DL440/ DL450)	При ручном конфигурировании ввода/вывода присвоены адреса вне диапазона. Исправьте присвоение адресов с помощью AUX 46. SP45 включается, а код ошибки записывается в V7755.
E264	ДУБЛИРОВАННЫЕ ССЫЛКИ ВВОДА/ВЫВОДА (DL440/ DL450)	При ручном конфигурировании ввода/вывода присвоены дублированные адреса. Исправьте присвоение адресов с помощью AUX 46.
E311	ОШИБКА 1 СВЯЗИ РУЧНОГО ПРОГРАММАТОРА (HP)	Запрос от ручного программатора не может быть обработан процессором. Сотрите ошибку и повторите запрос. Если ошибка появится вновь, смените процессор. SP46 включается, а код ошибки записывается в V7756.
E312	ОШИБКА 2 СВЯЗИ РУЧНОГО ПРОГРАММАТОРА (HP)	Обнаружена ошибка в данных при обмене с процессором. Очистите ошибочные данные и повторите запрос. Если ошибка появляется снова, проверьте кабели между устройствами, замените ручной программатор, а затем при необходимости и процессор. SP46 включается, а код ошибки записывается в V7756.
E313	ОШИБКА 3 СВЯЗИ РУЧНОГО ПРОГРАММАТОРА (HP)	Обнаружена ошибка в адресах при обмене с процессором. Очистите ошибочный адрес и повторите запрос. Если ошибка появляется снова, проверьте кабели между устройствами, замените ручной программатор, а затем при необходимости и процессор. SP46 включается, а код ошибки записывается в V7756.

Код ошибки	Название	Описание
E316	ОШИБКА 6 СВЯЗИ РУЧНОГО ПРОГРАММАТОРА (HP)	Обнаружена ошибка в режиме обмена с процессором. Удалите ошибку и повторите запрос. Если ошибка появляется снова, проверьте кабели между устройствами, замените ручной программатор, а затем при необходимости и процессор. SP46 включается, а код ошибки записывается в V7756.
E320	ТАЙМ-АУТ СВЯЗИ РУЧНОГО ПРОГРАММАТОРА	Процессор не отвечает на запрос связи ручного программатора. Проверьте кабель, убедитесь, что он исправлен и правильно подсоединен. Повторите цикл включения системы и, если ошибка продолжает повторяться, сначала замените процессор, затем, если это будет необходимо, и ручной программатор.
E321	ОШИБКА СВЯЗИ	Обнаружена ошибка в данных при обмене с процессором. Проверьте кабель, убедитесь, что он исправлен и правильно подсоединен. Повторите цикл включения системы и, если ошибка продолжает повторяться, сначала замените процессор, затем, если это будет необходимо, и ручной программатор.
E352	ФОНОВАЯ ОШИБКА СВЯЗИ	Ошибка связи между процессором и интеллектуальным модулем. Неправильная ссылка на слот при попытке использовать команды READ/WRITE, например, с интерфейса DCM. Номер слота с модулем, который дал ошибку ввода/вывода, хранится в V7660 - V7764. Повторите цикл включения питания, чтобы убрать эту ошибку.
E360	ТАЙМ-АУТ ПЕРИФЕРИЙНОГО ПОРТА РУЧНОГО ПРОГРАММАТОРА	Устройство, подсоединенное к периферийному порту, не отвечает на запросы связи ручного программатора. Убедитесь, что кабельная разводка правильна и не повреждена. Может быть поврежден периферийный порт или ручной программатор.
E4**	ДЕФЕКТ ПРОГРАММЫ	Прикладная 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	ОТСУТСТВИЕ ОПЕРАТОРА END	Прикладная программа должна заканчиваться оператором END. Введите оператор END в соответствующее место вашей прикладной программы. SP52 включается, а код ошибки записывается в V7755.
E402	ОТСУТСТВИЕ МЕТКИ LBL (DL440/ DL450)	Команды GOTO, GTS, MOV MC или LD LBL использованы без соответствующей метки. Обратитесь к разделу по программированию для получения информации по этим командам. SP52 включается, а код ошибки записывается в V7755.
E403	ОТСУТСТВИЕ RET (DL440/ DL450	Подпрограмма в программе не заканчивается командой RET. SP52 включается, а код ошибки записывается в V7755.

Код ошибки	Название	Описание
E404	ОТСУТСТВИЕ FOR (DL440, DL450)	Команда NEXT не имеет соответствующей команды FOR. SP52 включается, а код ошибки записывается в V7755.
E405	ОТСУТСТВИЕ NEXT (DL440, DL450)	Команда FOR не имеет соответствующей команды NEXT. SP52 включается, а код ошибки записывается в V7755.
E406	ОТСУТСТВИЕ IRT	Подпрограмма прерывания в программе не заканчивается командой IRT. SP52 включается, а код ошибки записывается в V7755.
E412	SBR/LBL >64 (DL440, DL450)	В программе содержится более 64 команд SBR, LBL или DLBL. Эта ошибка возникает также, если в программе более 128 команд GTS или GOTO. SP52 включается, а код ошибки записывается в V7755.
E413	FOR / NEXT >64 (DL440)	В прикладной программе содержится более 64 циклов FOR / NEXT. SP52 включается, а код ошибки записывается в V7755. (DL450 допускает неограниченное использование FOR- NEXT)
E421	ДУБЛИРОВАННАЯ ССЫЛКА НА СТАДИЮ	В прикладной программе содержится две и большее число меток SL 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	ДУБЛИРОВАННАЯ ССЫЛКА SBR/LBL (DL440, DL450)	В прикладной программе содержится две и большее число команд SBR или LBL с одним и тем же номером. Допустим только уникальный номер для каждой подпрограммы и метки. SP52 включается, а код ошибки записывается в V7755.
E423	ВЛОЖЕННЫЕ ЦИКЛЫ (DL440, DL450)	Вложенные циклы (когда один цикл FOR / NEXT находится внутри другого) не допустимы в семействах DL440. SP52 включается, а код ошибки записывается в V7755.
E431	НЕПРАВИЛЬНЫЙ АДРЕС ISG/SG	ISG или SG не должны при программировании ставиться после последнего оператора END, например, в подпрограмме. SP52 включается, а код ошибки записывается в V7755.
E432	НЕПРАВИЛЬНЫЙ АДРЕС ПЕРЕХОДА (GOTO) (DL440, DL450)	LBL, соответствующий команде GOTO, не должны при программировании ставиться после последнего оператора END, например, в подпрограмме. SP52 включается, а код ошибки записывается в V7755.
E433	НЕПРАВИЛЬНЫЙ АДРЕС SBR (DL440, DL450)	SBR не должен при программировании ставиться после последнего оператора END, ни в главной программе, ни в программе прерывания. SP52 включается, а код ошибки записывается в V7755.
E434	НЕПРАВИЛЬНЫЙ АДРЕС RTC (DL440, DL450)	RTC не должен при программировании ставиться после последнего оператора END, ни в главной программе, ни в программе прерывания. SP52 включается, а код ошибки записывается в V7755

Код ошибки	Название	Описание
E435	НЕПРАВИЛЬНЫЙ АДРЕС RT (DL440, DL450)	RT не должен при программировании ставиться после оператора END, ни в главной программе, ни в программе прерывания. SP52 включается, а код ошибки записывается в V7755
E436	НЕПРАВИЛЬНЫЙ АДРЕС INT	INT не должен при программировании ставиться в главной программе после оператора END. SP52 включается, а код ошибки записывается в V7755.
E437	НЕПРАВИЛЬНЫЙ АДРЕС IRTC	IRTC не должен при программировании ставиться в главной программе после оператора END. SP52 включается, а код ошибки записывается в V7755.
E438	НЕПРАВИЛЬНЫЙ АДРЕС IRT	IRT не должен при программировании ставиться в главной программе после оператора END. SP52 включается, а код ошибки записывается в V7755.
E440	НЕПРАВИЛЬНЫЙ АДРЕС ДАННЫХ (DL440, DL450)	Команда DLBL должна применяться либо в главной программе (но не после оператора END), либо в цепи, содержащей входной контакт(ы).
E441	ACON/NCON (DL440, DL450)	ACON или NCON не должны при программировании ставиться в главной программе после оператора END. SP52 включается, а код ошибки записывается в V7755.
E451	ПЛОХАЯ КОМАНДА MLS/MLR	Команды MLS должны нумероваться в возрастающем порядке сверху вниз.
E452	X ИСПОЛЬЗОВАН КАК ОБМОТКА	Тип данных X использован как выход обмотки.
E453	ОТСУТСТВИЕ ТАЙМЕРА/СЧЕТЧИКА	Использован контакт таймера или счетчика, хотя соответствующего таймера или счетчика не существует.
E454	ПЛОХАЯ КОМАНДА TMRA	В команде TMRA пропущен один из контактов.
E455	ПЛОХАЯ КОМАНДА CNT	В команде CNT или в команде UDC пропущен один из контактов.
E456	ПЛОХАЯ КОМАНДА SR	В команде SR пропущен один из контактов.
E461	ПЕРЕПОЛНЕНИЕ СТЕКА	В стеке записано более девяти логических уровней. Проверьте использование команд OR STR и AND STR.
E462	ПОТЕРЯ ЗНАЧИМОСТИ В СТЕКЕ	В стеке находится несогласованное число логических уровней. Убедитесь, что число команд OR STR и AND STR соответствует числу команд STR.

Код ошибки	Название	Описание
E463	ЛОГИЧЕСКАЯ ОШИБКА	Команда STR отсутствует в начале цепи релейной логики.
E464	ОТСУТСТВИЕ CKT	Цепь релейной логики не завершена надлежащим образом.
E471	ДУБЛИРОВАННАЯ ССЫЛКА НА ОБМОТКУ	Две или большее число команд OUT имеют ссылки на одну и ту же точку ввода/вывода.
E472	ДУБЛИРОВАННАЯ ССЫЛКА НА TMR	Две или большее число команд TMR имеют ссылки на один и тот же номер.
E473	ДУБЛИРОВАННАЯ ССЫЛКА НА CNT	Две или большее число команд CNT имеют ссылки на один и тот же номер.
E480	НЕПРАВИЛЬНЫЙ АДРЕС CV (DL440, DL450)	Команда CV использована в подпрограмме или в программе прерывания. Команда CV может использоваться только в главной программе (до оператора END).
E481	ПРОТИВОРЕЧИВЫЕ КОМАНДЫ (DL440, DL450)	Имеет место команда между сходящимися стадиями.
E482	ПРЕВЫШЕНО МАКСИМАЛЬНОЕ ЧИСЛО КОМАНД CV (DL440, DL450)	Число команд CV больше 17.
E483	НЕПРАВИЛЬНЫЙ АДРЕС CV JMP (DL440, DL450)	Команда CV JMP использована в подпрограмме или в программе прерывания.
E484	ОТСУТСТВИЕ КОМАНДЫ CV (DL440, DL450)	Перед командой CV JMP нет команды CV. Команда CV JMP должна непосредственно следовать за командой CV.
E485	ПРОПУЩЕНА НЕОБХОДИМАЯ КОМАНДА (DL440, DL450)	Команда CV JMP не находится между командой CV и командами {SG, ISG, BLK, BEND, END}.
E486	НЕПРАВИЛЬНЫЙ АДРЕС CALL BLK (DL440, DL450)	Команда CALL BLK использована в подпрограмме или в программе прерывания. Команда CALL BLK может использоваться только в главной программе (до оператора END).
E487	ОТСУТСТВИЕ КОМАНДЫ ST BLK (DL440, DL450)	За командой CALL BLK не следует команда ST BLK.

Код ошибки	Название	Описание
E488	НЕПРАВИЛЬНЫЙ АДРЕС КОМАНДЫ ST BLK (DL440, DL450)	Команда ST BLK использована в подпрограмме или в программе прерывания. Другая команда ST BLK использована между командами CALL BLK и END BLK.
E489	ДУБЛИРОВАННАЯ ССЫЛКА НА CR (DL440, DL450)	Управляющее реле, использованное в команде ST BLK, в другом месте использовано как выход.
E490	ОТСУТСТВИЕ КОМАНДЫ SG (DL440, DL450)	За командой ST BLK не следует непосредственно команда SG.
E491	НЕПРАВИЛЬНЫЙ АДРЕС КОМАНДЫ ISG (DL440, DL450)	Команда ISG находится между командами ST BLK и END BLK.
E492	НЕПРАВИЛЬНЫЙ АДРЕС КОМАНДЫ END BLK (DL440, DL450)	Команда END BLK использована в подпрограмме или в программе прерывания. За командой END BLK не следует команда ST BLK.
E493	ОТСУТСТВИЕ НЕОБХОДИМОЙ КОМАНДЫ (DL440, DL450)	Команда [CV, SG, ISG, ST BLK, END BLK] должна непосредственно следовать за командой END BLK.
E494	ОТСУТСТВИЕ КОМАНДЫ END BLK (DL440, DL450)	За командой ST BLK не следует команда END BLK.
E499	НЕПРАВИЛЬНАЯ КОМАНДЫ PRINT	Неправильное применение команды PRINT. Кавычки или пробелы не введены или введены неправильно.
E501	ПЛОХОЙ ВВОД	На ручном программаторе неправильно используется клавиша или группа клавиш.
E502	ПЛОХОЙ АДРЕС	На ручном программаторе введен неправильный адрес или адрес вне допустимого диапазона адресов.
E503	ПЛОХАЯ КОМАНДА	На ручном программаторе введена неправильная команда.
E504	ПЛОХАЯ ССЫЛКА REF/VAL	В команде неправильно введена ссылка или номер ссылки.
E505	НЕПРАВИЛЬНАЯ КОМАНДА	На ручном программаторе введена неправильная команда.
E506	НЕПРАВИЛЬНАЯ ОПЕРАЦИЯ	На ручном программаторе сделана попытка ввести неправильную операцию.

Код ошибки	Название	Описание
E520	ПЛОХАЯ ОПЕРАЦИЯ В РАБОЧЕМ РЕЖИМЕ	На ручном программаторе сделана попытка ввести операцию, которая неверна в рабочем режиме.
E521	ПЛОХАЯ ОПЕРАЦИЯ В РЕЖИМЕ TEST- RUN	На ручном программаторе сделана попытка ввести операцию, которая неверна в режиме TEST RUN.
E523	ПЛОХАЯ ОПЕРАЦИЯ В РЕЖИМЕ TEST - PGM)	На ручном программаторе сделана попытка ввести операцию, которая неверна в режиме TEST PROGRAM (Тестирования программ).
E524	ПЛОХАЯ ОПЕРАЦИЯ В ПРОГРАММНОМ РЕЖИМЕ	На ручном программаторе сделана попытка ввести операцию, которая неверна в программном (PROGRAM) режиме.
E525	ПЕРЕКЛЮЧАТЕЛЬ РЕЖИМОВ	На ручном программаторе сделана попытка выполнить операцию, когда переключатель режимов находится в положении, отличном от положения TERM.
E526	РЕЖИМ OFF LINE	Ручной программатор находится в АВТОНОМНОМ (OFF LINE) режиме Для изменения режима на ОПЕРАТИВНЫЙ (ON LINE) используйте AUX 64.
E540	ПРОЦЕССОР БЛОКИРОВАН (DL440, DL450)	Процессор блокирован паролем. Для разблокирования процессора используйте функцию AUX 82, позволяющую работать с паролем.
E541	НЕПРАВИЛЬНЫЙ ПАРОЛЬ (DL440, DL450)	Пароль, который используется для разблокирования процессора с помощью AUX 82, неверный.
E542	СБРОС ПАРОЛЯ (DL440, DL450)	Процессор включается с неправильным паролем и устанавливает его значение 00000000. Пароль может быть введен повторно с помощью AUX 81.
E601	ПАМЯТЬ НЕДОСТАТОЧНА	Выдается при попытке ввести команду, которая требует больше памяти, чем память, имеющаяся в процессоре.
E602	ОТСУТСТВИЕ КОМАНДЫ	Функция поиска выполнена, данная команда не найдена.
E603	ОТСУТСТВИЕ ДАННЫХ (DL440, DL450)	Функция поиска выполнена, данные не найдены.

Код ошибки	Название	Описание
E604	ОТСУТСТВИЕ ССЫЛКИ	Функция поиска не нашла указанную ссылку.
E610	НЕВЕРНЫЙ ТИП ВВОДА/ВЫВОДА	Прикладная программа обращается к модулю ввода/вывода с неправильным указанием типа модуля.
E620	ПРЕВЫШЕНИЕ ПАМЯТИ	Выдается при попытке передавать между процессором и ручным программатором больше данных, чем имеется в принимающем устройстве.
E621	КАРТРИДЖ ПАМЯТИ НЕ ПУСТ	Была сделана попытка записи в непустой картридж памяти. Очистите картридж памяти и затем повторите запись.
E622	НЕТ КАРТРИДЖА ПАМЯТИ В РУЧНОМ ПРОГРАММАТОРЕ	Была сделана попытка передачи данных в отсутствующий картридж памяти ручного программатора (или в его неисправный картридж памяти).
E623	СИСТЕМНЫЙ КАРТРИДЖ ПАМЯТИ	Была запрошена функция в картридже памяти, который содержит только системную информацию.
E624	ТОЛЬКО V-ПАМЯТЬ	Была запрошена функция в картридже памяти, который содержит только данные V-памяти.
E625	ТОЛЬКО ПРОГРАММЫ	Была запрошена функция в картридже памяти, который содержит только данные программ.
E626	КАРТРИДЖ ПАМЯТИ ППЗУ	Была сделана попытка передать данные с магнитной ленты в картридж памяти УФПЗУ. Такая передача должна делаться с использованием картриджа ОЗУ КМОП.
E627	ПЛОХАЯ ЗАПИСЬ	Была сделана попытка записи в картридж памяти, защищенный от записи или неисправный. Проверьте перемычку защиты от записи и, если необходимо, замените картридж памяти.
E640	ОШИБКА СРАВНЕНИЯ	При сравнении картриджа памяти и памяти процессора была обнаружена ошибка.
E641	ГРОМКОСТЬ ЗВУКА	Уровень звука на кассетном магнитофоне не установлена надлежащим образом. Отрегулируйте звук и повторите операцию.
E642	КОНТРОЛЬНАЯ СУММА	Обнаружена ошибка при передаче в картридж памяти ручного программатора. Проверьте разводка кабелей и повторите операцию.
E650	СИСТЕМНАЯ ОШИБКА В РУЧНОМ ПРОГРАММАТОРЕ	В ручном программаторе имеет место системная ошибка. Повторно включите ручной программатор. Если ошибка появится снова, замените ручной программатор.

Код ошибки	Название	Описание
E651	ОШИБКА ПЗУ В РУЧНОМ ПРОГРАММАТОРЕ	В ручном программаторе обнаружена ошибка ПЗУ. Повторно включите ручной программатор. Если ошибка появится снова, замените ручной программатор.
E652	ОШИБКА ОЗУ В РУЧНОМ ПРОГРАММАТОРЕ	В ручном программаторе обнаружена ошибка ОЗУ. Повторно включите ручной программатор. Если ошибка появится снова, замените ручной программатор.
E653	СЛАБАЯ БАТАРЕЯ КАРТРИДЖА ПАМЯТИ	Батарея картриджа памяти ОЗУ КМОП имеет низкое напряжение и должна быть заменена.

Времена выполнения команд

C

В этой главе...

Введение

Регистры данных V-памяти

В данном приложении приводятся таблицы времен выполнения команд для процессоров DL430, DL440 и DL450. Многие из приведенных времен зависят от типа используемых в команде данных. Регистры разделены на следующие типы:

- Регистры (слов) данных
- Регистры битов

Регистры битов V-памяти

Некоторые ячейки V-памяти рассматриваются как регистры данных, например, ячейки V-памяти, в которых хранятся текущие значения счетчика или таймера. Стандартная V-память пользователя может рассматриваться как регистры данных V-памяти. Следует отметить, что Вы можете загрузить битовую комбинацию в регистры этого типа, хотя их основное назначение - регистры данных. Следующие ячейки рассматриваются как регистры данных.

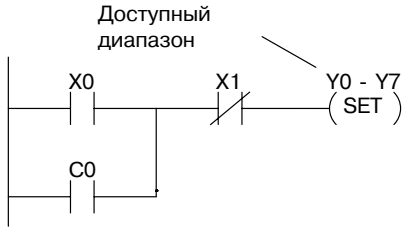
Регистры данных	DL430	DL440	DL450
Текущие значения счетчика	V00000 - V 00177	V00000 - V00377	V00000 - V00377
Текущие значения таймера	V01000 - V01177	V01000 - V01177	V01000 - V01377
Слова пользовательских данных	V1400 - V7377	V1400 - V7377 V10000 - V17777	V1400 - V7377 V10000 - V37777

Напоминаем вам, что некоторые из дискретных точек, например, X, Y, C и др., автоматически отображаются в V-памяти. Следующие регистры битов содержат такие данные:

Регистры данных	DL430	DL440	DL450
Входные точки (X)	V40400 - V 40423	V40400 - V 40423	V40400 - V 40477
Выходные точки (Y)	V40500 - V40523	V40500 - V40523	V40500 - V40577
Управляющие Реле (C)	V40600 - V40635	V40600 - V40677	V40600 - V40777
Биты Состояния Таймера	V41100 - V41107	V41100 - V41177	V41100 - V41117
Биты Состояния Счетчика	V41040 - V41147	V41040 - V41147	V41040 - V41 157
Стадии	V41000 - V41027	V41000 - V41077	V41000 - V41077
Удаленный ввод/вывод (Глобальные GX) (Глобальные GY)	V40000 - V40037	V40000 - V40077	V40000 - V40137 V40200 - V40377

Как читать таблицы

Некоторые команды могут иметь несколько параметров. Например, команда SET, показанная в программе справа может устанавливать одну точку или диапазон точек.



В этих случаях времена выполнения зависят от числа и типа параметров. В таблице времен выполнения указываются времена для различных ситуаций, как показано ниже:

SET	1st #: X, Y, C, S, GX, GY	20.8 µs
	2nd #: X, Y, C, S, GX, GY (N pt)	13.0µs+7.8µsxN
RST	1st #: X, Y, C, S, GX, GY	19.5 µs
	2nd #: X, Y, C, S, GX, GY (N pt)	11.7µs+7.8µsxN

Исполнение зависит от числа ячеек и типа используемых данных

ПРИМЕЧАНИЕ: Тип данных GY, указанный для некоторых команд в данном приложении действует только для процессора DL450, который использует оба типа данных GX и GY для точек удаленного ввода/вывода. Процессоры DL430 и DL440 используют тип данных GX для всех точек удаленного ввода/вывода.

Булевы команды

Булевы команды		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
STR	X, Y, C, T, CT	4.7 µs	4.7 µs	0.33 µs	0.33 µs	0.96 µs	0.96 µs
	S, SP, GX, GY	4.7 µs	4.7 µs	8.0 µs	8.0 µs	1.0 µs	1.0 µs
STRN	X, Y, C, T, CT	5.7 µs	5.7 µs	0.33 µs	0.33 µs	1.0 µs	1.0 µs
	S, SP, GX, GY	5.7 µs	5.7 µs	8.0 µs	8.0 µs	1.0 µs	1.0 µs
STRB	V:Register (Bit)	—	—	—	—	5.0 µs	??? µs
	P:Indirect (Bit)	—	—	—	—	15.3 µs	??? µs
STRNB	V:Register (Bit)	—	—	—	—	4.9 µs	??? µs
	P:Indirect (Bit)	—	—	—	—	15.1 µs	??? µs
OR	X, Y, C, T, CT	3.1 µs	3.1 µs	0.33 µs	0.33 µs	0.9 µs	0.9 µs
	S, SP, GX, GY	3.1 µs	3.1 µs	4.24 µs	4.24 µs	0.9 µs	0.9 µs
ORN	X, Y, C, CT	4.1 µs	4.1 µs	0.33 µs	0.33 µs	0.96 µs	0.96 µs
	S, SP, GX, GY	4.1 µs	4.1 µs	4.67 µs	4.67 µs	0.96 µs	0.96 µs
ORB	V:Register (Bit)	—	—	—	—	4.7 µs	??? µs
	P:Indirect (Bit)	—	—	—	—	15.0 µs	??? µs
ORNB	V:Register (Bit)	—	—	—	—	4.6 µs	??? µs
	P:Indirect (Bit)	—	—	—	—	14.8 µs	??? µs
AND	X, Y, C, T, CT	2.4 µs	2.4 µs	0.33 µs	0.33 µs	0.8 µs	0.8 µs
	S, SP, GX, GY	2.4 µs	2.4 µs	2.72 µs	2.72 µs	0.8 µs	0.8 µs
ANDN	X, Y, C, T, CT	3.4 µs	3.4 µs	0.33 µs	0.33 µs	0.9 µs	0.9 µs
	S, SP, GX, GY	3.4 µs	3.4 µs	3.14 µs	3.14 µs	0.9 µs	0.9 µs
ANDB	V:Register (Bit)	—	—	—	—	4.6 µs	??? µs
	P:Indirect (Bit)	—	—	—	—	14.8 µs	??? µs
ANDNB	V:Register (Bit)	—	—	—	—	4.4 µs	??? µs
	P:Indirect (Bit)	—	—	—	—	14.7 µs	??? µs
ANDSTR	None	1.8 µs	1.8 µs	0.33 µs	0.33 µs	0.5 µs	0.5 µs
ORSTR	None	1.8 µs	1.8 µs	0.33 µs	0.33 µs	0.5 µs	0.5 µs
OUT	X, Y, C	6.7 µs	6.7 µs	0.33 µs	0.33 µs	2.9 µs	2.9 µs
	GX, GY	6.7 µs	6.7 µs	2.06 µs	2.06 µs	2.9 µs	2.9 µs
OUTB	V:Register (Bit)	—	—	—	—	5.6 µs	??? µs
	P:Indirect (Bit)	—	—	—	—	16.4 µs	??? µs
OROUT	X, Y, C, GX, GY	8.3 µs	8.3 µs	6.1 µs	6.1 µs	3.4 µs	3.4 µs
NOT	None	—	—	3.2 µs	—	**???** µs	—

Булевы команды		**DL430**		**DL440**		**DL450**	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
PD	X, Y, C	15.1 µs	15.1 µs	8.5 µs	8.5 µs	10.0 µs	10.0 µs
STRPD	X, Y, C, T, CT, S, SP, GX, GY	—	—	—	—	3.7 µs	3.7 µs
STRND	X, Y, C, T, CT, S, SP, GX, GY	—	—	—	—	2.8 µs	2.8 µs
ANDPD	X, Y, C, T, CT, S, SP, GX, GY	—	—	—	—	3.6 µs	3.6 µs
ANDND	X, Y, C, T, CT, S, SP, GX, GY	—	—	—	—	2.7 µs	2.7 µs
ORPD	X, Y, C, T, CT, S, SP, GX, GY	—	—	—	—	3.6 µs	3.6 µs
ORND	X, Y, C, T, CT, S, SP, GX, GY	—	—	—	—	2.7 µs	2.7 µs
SET	1st #: X, Y, C	20.8 µs	5.2 µs	0.33 µs	0.33 µs	5.6 µs	1.0 µs
	S, GX, GY	20.8 µs	5.2 µs	14.6 µs	5.4 µs	5.6 µs	1.0 µs
	2nd #: X, Y, C, S, GX (N pt)	13.0 µs+ 7.8µsxN	5.2 µs	8.9 µs+ 5.7µsxN	5.4 µs	7.6 µs+ 0.6µsxN	1.2 µs
SETB	V:Register (Bit)	—	—	—	—	10.5 µs	2.3 µs
	P:Indirect (Bit)	—	—	—	—	22.2 µs	13.8 µs
RST	1st #: X, Y, C	19.5 µs	5.2 µs	0.33 µs	0.33 µs	5.6 µs	1.0 µs
	S, GX, GY	19.5 µs	5.2 µs	13.7 µs	4.5 µs	5.6 µs	5.6 µs
	2nd #: X, Y, C, S, GX, GY (N pt)	11.7 µs+ 7.8µsxN	5.2 µs	8.0 µs+ 5.7µsxN	4.5 µs	7.6 µs+ 0.6µsxN	1.2 µs
	1st #: T, CT	28.2 µs	5.2 µs	15.2 µs	3.8 µs	10.2 µs	0.9 µs
	2nd #: T, CT (N pt)	23.3 µs+ 5.8µsxN	5.2 µs	10.7 µs+ 4.6µsxN	3.8 µs	6.1 µs+ 1.9µsxN	1.2 µs
RSTB	V:Register (Bit)	—	—	—	—	10.5 µs	2.3 µs
	P:Indirect (Bit)	—	—	—	—	22.2 µs	13.8 µs
PAUSE	1wd: Y	24.0 µs	24.0 µs	14.7 µs	14.7 µs	7.4 µs	7.5 µs
	2wd: Y (N points)	18 µs+ 6µs x N	18µs+ 6µs x N	11µs+ 4µs x N	11µs+ 4µs x N	14.6µs+ 0.32µs xN	14.6µs+ 0.3µs x N

Логические команды сравнения

Логические команды сравнения			DL430		DL440		DL450	
Команда	**Разрешенные типы данных**		**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
STRE	1st	2nd						
	V: Data Reg.	V:Data Reg.	61 µs	13.5 µs	56 µs	12.2 µs	4.9 µs	4.9
		V:Bit Reg.	182 µs	13.5 µs	130 µs	12.2 µs	4.9 µs	4.9
		K:Constant	75 µs	13.5 µs	69 µs	12.2 µs	3.5 µs	3.5
		P:Indir. (Data)	—	—	178 µs	12.2 µs	9.9 µs	9.9
		P:Indir. (Bit)	—	—	270 µs	12.2 µs	9.9 µs	9.9
	V: Bit Reg.	V:Data Reg.	182 µs	13.5 µs	130 µs	12.2 µs	4.9 µs	4.9
		V:Bit Reg.	300 µs	13.5 µs	206 µs	12.2 µs	4.9 µs	4.9
		K:Constant	193 µs	13.5 µs	206 µs	12.2 µs	3.5 µs	3.5
		P:Indir. (Data)	—	—	252 µs	12.2 µs	9.9 µs	9.9
		P:Indir. (Bit)	—	—	347 µs	12.2 µs	9.9 µs	9.9
	P:Indir. (Data)	V:Data Reg.	—	—	174 µs	12.2 µs	3.4 µs	9.9
		V:Bit Reg.	—	—	248 µs	12.2 µs	3.4 µs	9.9
		K:Constant	—	—	182 µs	12.2 µs	8.3 µs	8.3
		P:Indir. (Data)	—	—	296 µs	12.2 µs	14.3 µs	14.3
		P:Indir. (Bit)	—	—	387 µs	12.2 µs	14.3 µs	14.3
	P:Indir. (Bit)	V:Data Reg.	—	—	265 µs	12.2 µs	3.4 µs	9.9
		V:Bit Reg.	—	—	341 µs	12.2 µs	3.4 µs	9.9
		K:Constant	—	—	276 µs	12.2 µs	8.3 µs	8.3
		P:Indir. (Data)	—	—	387 µs	12.2 µs	14.3 µs	14.3
		P:Indir. (Bit)	—	—	480 µs	12.2 µs	14.3 µs	14.3
STRNE	1st	2nd						
	V: Data Reg.	V:Data Reg.	63 µs	13.5 µs	56 µs	12.2 µs	4.9 µs	4.9
		V:Bit Reg.	180 µs	13.5 µs	130 µs	12.2 µs	4.9 µs	4.9
		K:Constant	77 µs	13.5 µs	69 µs	12.2 µs	3.5 µs	3.5
		P:Indir. (Data)	—	—	178 µs	12.2 µs	9.9 µs	9.9
		P:Indir. (Bit)	—	—	270 µs	12.2 µs	9.9 µs	9.9
	V: Bit Reg.	V:Data Reg.	180 µs	13.5 µs	130 µs	12.2 µs	4.9 µs	4.9
		V:Bit Reg.	298 µs	13.5 µs	206 µs	12.2 µs	4.9 µs	4.9
		K:Constant	195 µs	13.5 µs	206 µs	12.2 µs	3.5 µs	3.5
		P:Indir. (Data)	—	—	252 µs	12.2 µs	9.9 µs	9.9
		P:Indir. (Bit)	—	—	347 µs	12.2 µs	9.9 µs	9.9
	P:Indir. (Data)	V:Data Reg.	—	—	174 µs	12.2 µs	3.4 µs	9.9
		V:Bit Reg.	—	—	248 µs	12.2 µs	3.4 µs	9.9
		K:Constant	—	—	182 µs	12.2 µs	8.3 µs	8.3
		P:Indir. (Data)	—	—	296 µs	12.2 µs	14.3 µs	14.3
		P:Indir. (Bit)	—	—	387 µs	12.2 µs	14.3 µs	14.3
	P:Indir. (Bit)	V:Data Reg.	—	—	265 µs	12.2 µs	3.4 µs	9.9
		V:Bit Reg.	—	—	341 µs	12.2 µs	3.4 µs	9.9
		K:Constant	—	—	276 µs	12.2 µs	8.3 µs	8.3
		P:Indir. (Data)	—	—	387 µs	12.2 µs	14.3 µs	14.3
		P:Indir. (Bit)	—	—	480 µs	12.2 µs	14.3 µs	14.3

Логические команды сравнения			**DL430**		**DL440**		**DL450**	
Команда	**Разрешенные типы данных**		**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
ORE	1st	2nd						
	V: Data Reg.	V:Data Reg.	62 µs	11.0 µs	48 µs	10.8 µs	4.9 µs	4.9 µs
		V:Bit Reg.	180 µs	11.0 µs	122 µs	10.8 µs	4.9 µs	4.9 µs
		K:Constant	73 µs	11.0 µs	55 µs	10.8 µs	3.5 µs	3.5 µs
		P:Indir. (Data)	—	—	170 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	262 µs	10.8 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	180 µs	11.0 µs	122 µs	10.8 µs	4.9 µs	4.9 µs
		V:Bit Reg.	297 µs	11.0 µs	198 µs	10.8 µs	4.9 µs	4.9 µs
		K:Constant	191 µs	11.0 µs	131 µs	10.8 µs	3.5 µs	3.5 µs
		P:Indir. (Data)	—	—	244 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	340 µs	10.8 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	166 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	240 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	175 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	288 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	379 µs	10.8 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	257 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	333 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	268 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	380 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	473 µs	10.8 µs	14.3 µs	14.3 µs
ORNE	1st	2nd						
	V: Data Reg.	V:Data Reg.	62 µs	11.0 µs	48 µs	10.8 µs	4.9 µs	4.9 µs
		V:Bit Reg.	178 µs	11.0 µs	122 µs	10.8 µs	4.9 µs	4.9 µs
		K:Constant	75 µs	11.0 µs	55 µs	10.8 µs	3.5 µs	3.5 µs
		P:Indir. (Data)	—	—	170 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	262 µs	10.8 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	178 µs	11.0 µs	122 µs	10.8 µs	4.9 µs	4.9 µs
		V:Bit Reg.	296 µs	11.0 µs	198 µs	10.8 µs	4.9 µs	4.9 µs
		K:Constant	192 µs	11.0 µs	131 µs	10.8 µs	3.5 µs	3.5 µs
		P:Indir. (Data)	—	—	244 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	340 µs	10.8 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	166 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	240 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	175 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	288 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	379 µs	10.8 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	257 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	333 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	268 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	380 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	473 µs	10.8 µs	14.3 µs	14.3 µs

Логические команды сравнения			**DL430**		**DL440**		**DL450**	
Команда	**Разрешенные типы данных**		**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
ANDE	1st	2nd						
	V: Data Reg.	V:Data Reg.	60 µs	11.0 µs	48 µs	10.8 µs	4.9 µs	4.9 µs
		V:Bit Reg.	178 µs	11.0 µs	122 µs	10.8 µs	4.9 µs	4.9 µs
		K:Constant	74 µs	11.0 µs	55 µs	10.8 µs	3.5 µs	3.5 µs
		P:Indir. (Data)	—	—	170 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	262 µs	10.8 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	178 µs	11.0 µs	122 µs	10.8 µs	4.9 µs	4.9 µs
		V:Bit Reg.	296 µs	11.0 µs	198 µs	10.8 µs	4.9 µs	4.9 µs
		K:Constant	192 µs	11.0 µs	131 µs	10.8 µs	3.5 µs	3.5 µs
		P:Indir. (Data)	—	—	244 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	340 µs	10.8 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	166 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	240 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	175 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	288 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	379 µs	10.8 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	257 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	333 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	268 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	380 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	473 µs	10.8 µs	14.3 µs	14.3 µs
ANDNE	1st	2nd						
	V: Data Reg.	V:Data Reg.	62 µs	11.0 µs	48 µs	10.8 µs	4.9 µs	4.9 µs
		V:Bit Reg.	179 µs	11.0 µs	122 µs	10.8 µs	4.9 µs	4.9 µs
		K:Constant	73 µs	11.0 µs	55 µs	10.8 µs	3.5 µs	3.5 µs
		P:Indir. (Data)	—	—	170 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	262 µs	10.8 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	179 µs	11.0 µs	122 µs	10.8 µs	4.9 µs	4.9 µs
		V:Bit Reg.	297 µs	11.0 µs	198 µs	10.8 µs	4.9 µs	4.9 µs
		K:Constant	191 µs	11.0 µs	131 µs	10.8 µs	3.5 µs	3.5 µs
		P:Indir. (Data)	—	—	244 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	340 µs	10.8 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	166 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	240 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	175 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	288 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	379 µs	10.8 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	257 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	333 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	268 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	380 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	473 µs	10.8 µs	14.3 µs	14.3 µs

Логические команды сравнения			**DL430**		**DL440**		**DL450**	
Команда	**Разрешенные типы данных**		**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
STR	1st	2nd						
	T, CT	V:Data Reg.	65 µs	13.5 µs	56 µs	12.2 µs	4.8 µs	4.8 µs
		V:Bit Reg.	182 µs	13.5 µs	130 µs	12.2 µs	4.8 µs	4.8 µs
		K:Constant	75 µs	13.5 µs	63 µs	12.2 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	178 µs	12.2 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	270 µs	12.2 µs	9.9 µs	9.9 µs
	1st	2nd						
	V: Data Reg.	V:Data Reg.	66 µs	13.5 µs	56 µs	12.2 µs	4.8 µs	4.8 µs
		V:Bit Reg.	184 µs	13.5 µs	130 µs	12.2 µs	4.8 µs	4.8 µs
		K:Constant	79 µs	13.5 µs	63 µs	12.2 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	178 µs	12.2 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	270 µs	12.2 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	184 µs	13.5 µs	130 µs	12.2 µs	4.8 µs	4.8 µs
		V:Bit Reg.	800 µs	13.5 µs	206 µs	12.2 µs	4.8 µs	4.8 µs
		K:Constant	193 µs	13.5 µs	139 µs	12.2 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	252 µs	12.2 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	347 µs	12.2 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	174 µs	12.2 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	248 µs	12.2 µs	9.9 µs	9.9 µs
		K:Constant	—	—	182 µs	12.2 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	296 µs	12.2 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	387 µs	12.2 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	265 µs	12.2 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	341 µs	12.2 µs	9.9 µs	9.9 µs
		K:Constant	—	—	276 µs	12.2 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	387 µs	12.2 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	480 µs	12.2 µs	14.3 µs	14.3 µs
STRN	1st	2nd						
	T, CT	V:Data Reg.	65 µs	13.5 µs	56 µs	12.2 µs	4.8 µs	4.8 µs
		V:Bit Reg.	180 µs	13.5 µs	130 µs	12.2 µs	4.8 µs	4.8 µs
		K:Constant	75 µs	13.5 µs	63 µs	12.2 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	178 µs	12.2 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	270 µs	12.2 µs	9.9 µs	9.9 µs
	1st	2nd						
	V: Data Reg.	V:Data Reg.	65 µs	13.5 µs	56 µs	12.2 µs	4.8 µs	4.8 µs
		V:Bit Reg.	180 µs	13.5 µs	130 µs	12.2 µs	4.8 µs	4.8 µs
		K:Constant	77 µs	13.5 µs	63 µs	12.2 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	178 µs	12.2 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	270 µs	12.2 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	180 µs	13.5 µs	130 µs	12.2 µs	4.8 µs	4.8 µs
		V:Bit Reg.	298 µs	13.5 µs	206 µs	12.2 µs	4.8 µs	4.8 µs
		K:Constant	195 µs	13.5 µs	139 µs	12.2 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	252 µs	12.2 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	347 µs	12.2 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	174 µs	12.2 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	248 µs	12.2 µs	9.9 µs	9.9 µs
		K:Constant	—	—	182 µs	12.2 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	296 µs	12.2 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	387 µs	12.2 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	265 µs	12.2 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	341 µs	12.2 µs	9.9 µs	9.9 µs
		K:Constant	—	—	276 µs	12.2 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	387 µs	12.2 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	480 µs	12.2 µs	14.3 µs	14.3 µs

Логические команды сравнения			**DL430**		**DL440**		**DL450**	
Команда	**Разрешенные типы данных**		**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
OR	1st	2nd						
	T, CT	V:Data Reg.	64 µs	11.0 µs	46 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	180 µs	11.0 µs	124 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	74 µs	11.0 µs	57 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	168 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	264 µs	10.8 µs	9.9 µs	9.9 µs
	1st	2nd						
	V: Data Reg.	V:Data Reg.	63 µs	13.5 µs	48 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	180 µs	13.5 µs	122 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	77 µs	13.5 µs	55 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	170 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	262 µs	10.8 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	180 µs	13.5 µs	122 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	298 µs	13.5 µs	198 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	195 µs	13.5 µs	131 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	244 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	340 µs	10.8 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	165 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	240 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	175 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	288 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	379 µs	10.8 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	257 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	333 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	268 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	380 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	473 µs	10.8 µs	14.3 µs	14.3 µs
ORN	1st	2nd						
	T, CT	V:Data Reg.	64 µs	11.0 µs	46 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	178 µs	11.0 µs	124 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	74 µs	11.0 µs	57 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	168 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	264 µs	10.8 µs	9.9 µs	9.9 µs
	1st	2nd						
	V: Data Reg.	V:Data Reg.	63 µs	13.5 µs	48 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	180 µs	13.5 µs	122 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	77 µs	13.5 µs	55 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	170 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	262 µs	10.8 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	180 µs	13.5 µs	122 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	298 µs	13.5 µs	198 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	195 µs	13.5 µs	131 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	244 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	340 µs	10.8 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	165 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	240 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	175 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	288 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	379 µs	10.8 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	257 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	333 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	268 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	380 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	473 µs	10.8 µs	14.3 µs	14.3 µs

Логические команды сравнения			**DL430**		**DL440**		**DL450**	
Команда	**Разрешенные типы данных**		**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
AND	1st	2nd						
	T, CT	V:Data Reg.	62 µs	11.0 µs	46 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	178 µs	11.0 µs	124 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	73 µs	11.0 µs	57 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	168 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	264 µs	10.8 µs	9.9 µs	9.9 µs
	1st	2nd						
	V: Data Reg.	V:Data Reg.	62 µs	11.0 µs	48 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	178 µs	11.0 µs	122 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	75 µs	11.0 µs	55 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	170 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	262 µs	10.8 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	178 µs	11.0 µs	122 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	296 µs	11.0 µs	198 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	192 µs	11.0 µs	131 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	244 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	340 µs	10.8 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	165 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	240 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	175 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	288 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	379 µs	10.8 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	257 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	333 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	268 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	380 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	473 µs	10.8 µs	14.3 µs	14.3 µs
ANDN	1st	2nd						
	T, CT	V:Data Reg.	62 µs	11.0 µs	46 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	179 µs	11.0 µs	124 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	73 µs	11.0 µs	57 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	168 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	264 µs	10.8 µs	9.9 µs	9.9 µs
	1st	2nd						
	V: Data Reg.	V:Data Reg.	63 µs	13.5 µs	48 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	180 µs	13.5 µs	122 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	77 µs	13.5 µs	55 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	170 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	262 µs	10.8 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	180 µs	13.5 µs	122 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	298 µs	13.5 µs	198 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	195 µs	13.5 µs	131 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	244 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	340 µs	10.8 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	165 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	240 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	175 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	288 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	379 µs	10.8 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	257 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	333 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	268 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	380 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	473 µs	10.8 µs	14.3 µs	14.3 µs

Команды немедленного действия

Команды немедленного действия		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
STRI	X	22.0 µs	7.5 µs	15.6 µs	4.5 µs	128.0 µs	15.1 µs
STRNI	X	21.3 µs	7.5 µs	15.6 µs	4.5 µs	128.0 µs	15.3 µs
ORI	X	21.5 µs	5.2 µs	9.7 µs	4.8 µs	128.0 µs	14.8 µs
ORNI	X	20.8 µs	5.2 µs	9.7 µs	4.8 µs	128.0 µs	15.0 µs
ANDI	X	18.1 µs	5.2 µs	9.4 µs	2.1 µs	128.0 µs	14.8 µs
ANDNI	X	20.7 µs	5.2 µs	9.4 µs	2.1 µs	9.4 µs	2.1 µs
LDI	Y	—	—	—	—	406.3 µs	1.8 µs
OUTI	Y	27.3 µs	27.3 µs	18.0 µs	18.0 µs	135.0 µs	135.0 µs
OROUTI	Y	27.0 µs	27.0 µs	20.0 µs	20.0 µs	135.0 µs	23.5 µs
SETI	1st #: Y 2nd #: Y (N pt)	48.3 µs 24.6µs+ 23.7 xN	5.2 µs 5.2 µs	16.0 µs 18.0µs+ 15.9 xN	3.8 µs 3.8 µs	12.2 µs 139.7µs+ 0.6 xN	1.8 µs 2.2 µs
RSTI	1st #: Y 2nd #: Y (N pt)	47.1 µs 23.3µs+ 23.7 xN	5.2 µs 5.2 µs	15.1 µs 17.1µs+ 15.9 xN	3.8 µs 3.8 µs	12.2 µs 140.3µs+ 0.7 xN	1.8 µs 2.3 µs
LDIF	1st 2nd X K	—	—	252µs+ 16µs xN	3.8 µs	9.5 µs	2.3 µs
OUTIF	1st 2nd Y K	—	—	598µs+ 12µs xN	4.0 µs	12.5 µS	2.3 µs

Команды часов/календаря

Команды часов/календаря		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
DATE	V:Data Reg. V:Bit Reg.	—	—	135.0 µs 385.0 µs	4.0 µs 4.0 µs	21.3 µs 21.3 µs	1.9 µs 1.9 µs
TIME	V:Data Reg. V:Bit Reg.	—	—	121.0 µs 373.0 µs	4.0 µs 4.0 µs	13.2 µs 13.2 µs	1.9 µs 1.9 µs

Команды регистра сдвига, счетчика, таймера

Команды регистра сдвига, счетчика, таймера			DL430		DL440		DL450	
Команда	**Разрешенные типы данных**		**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
TMR	1st	2nd						
	T	V:Data Reg.	63.9 µs	29.1 µs	45.0 µs	21.0 µs	40.0 µs	23.0 µs
		V:Bit Reg.	179.5 µs	29.1 µs	119.0 µs	21.0 µs	40.0 µs	23.0 µs
		K:Constant	76.3 µs	29.1 µs	52.4 µs	21.0 µs	35.7 µs	19.5 µs
		P:Indir. (Data)	—	—	200.7 µs	21.0 µs	51.8 µs	33.9 µs
		P:Indir. (Bit)	—	—	273.4 µs	21.0 µs	51.8 µs	33.9 µs
TMRF	1st	2nd						
	T	V:Data Reg.	63.9 µs	29.1 µs	45.0 µs	21.0 µs	77.8 µs	22.6 µs
		V:Bit Reg.	179.5 µs	29.1 µs	119.0 µs	21.0 µs	77.8 µs	22.6 µs
		K:Constant	76.3 µs	29.1 µs	52.4 µs	21.0 µs	69.8 µs	19.2 µs
		P:Indir. (Data)	—	—	200.7 µs	21.0 µs	83.4 µs	37.5 µs
		P:Indir. (Bit)	—	—	273.4 µs	21.0 µs	83.4 µs	37.5 µs
TMRA	1st	2nd						
	T	V:Data Reg.	74.0 µs	50.3 µs	50.6 µs	33.8 µs	66.0 µs	26.1 µs
		V:Bit Reg.	298.7 µs	275.0 µs	199.5 µs	182.7 µs	66.0 µs	26.1 µs
		K:Constant	85.0 µs	61.2 µs	58.3 µs	42.1 µs	61.8 µs	21.7 µs
		P:Indir. (Data)	—	—	228.0 µs	205.4 µs	76.8 µs	37.3 µs
		P:Indir. (Bit)	—	—	376.7 µs	354.0 µs	76.8 µs	37.3 µs
TMRAF	1st	2nd						
	T	V:Data Reg.	74.0 µs	50.3 µs	50.6 µs	33.8 µs	74.8 µs	26.1 µs
		V:Bit Reg.	298.7 µs	275.0 µs	199.5 µs	182.7 µs	74.8 µs	26.1 µs
		K:Constant	74.0 µs	74.0 µs	58.3 µs	42.1 µs	71.0 µs	21.7 µs
		P:Indir. (Data)	—	—	228.0 µs	205.4 µs	85.7 µs	37.3 µs
		P:Indir. (Bit)	—	—	376.7 µs	354.0 µs	85.7 µs	37.3 µs
CNT	1st	2nd						
	CT	V:Data Reg.	46.2 µs	38.2 µs	33.6 µs	29.8 µs	37.9 µs	24.6 µs
		V:Bit Reg.	161.4 µs	159.4 µs	107.6 µs	103.8 µs	37.9 µs	24.6 µs
		K:Constant	58.6 µs	50.6 µs	41.0 µs	37.2 µs	35.7 µs	21.6 µs
		P:Indir. (Data)	—	—	191.4 µs	186.7 µs	40.8 µs	35.7 µs
		P:Indir. (Bit)	—	—	264.7 µs	260.0 µs	40.8 µs	35.7 µs
SGCNT	1st	2nd						
	CT	V:Data Reg.	57.3 µs	44.2 µs	41.3 µs	32.9 µs	39.3 µs	23.8 µs
		V:Bit Reg.	172.5 µs	159.4 µs	119.5 µs	105.7 µs	39.3 µs	23.8 µs
		K:Constant	69.3 µs	56.2 µs	46.5 µs	40.4 µs	33.7 µs	20.3 µs
		P:Indir. (Data)	—	—	164.7 µs	156.7 µs	47.1 µs	34.7 µs
		P:Indir. (Bit)	—	—	263.4 µs	247.4 µs	47.1 µs	34.7 µs
UDC	1st	2nd						
	CT	V:Data Reg.	90.0 µs	60.6 µs	60.0 µs	41.9 µs	45.1 µs	34.8 µs
		V:Bit Reg.	314.7 µs	285.3 µs	209.0 µs	190.8 µs	45.1 µs	34.8 µs
		K:Constant	102.2 µs	72.8 µs	58.6 µs	50.5 µs	41.4 µs	30.9 µs
		P:Indir. (Data)	—	—	210.7 µs	198.7 µs	56.4 µs	45.9 µs
		P:Indir. (Bit)	—	—	358.7 µs	340.0 µs	56.4 µs	45.9 µs
SR	C (N points to shift)		36.8µs+ 2.3µsxN	17.9 µs	25.6µs+ 1.6µsxN	17.7 µs	8.9 µs+ 0.5 µsxN	7.7 µs

Команды вывода, загрузки стека/данных.аккумулятора

Команды вывода, загрузки стека данных/ аккумулятора		**DL430**		**DL440**		**DL450**	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
LD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	102.0 µs 206.0 µs 112.0 µs 278.0 µs 391.0 µs	5.0 µs 5.0 µs 5.0 µs 5.0 µs 5.0 µs	97.0 µs 166.0 µs 110.0 µs 213.0 µs 272.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	6.4 µs 6.4 µs 12.7 µs 4.9 µs 4.9 µs	0.8 µs 0.8 µs 1.7 µs 0.8 µs 0.8 µs
LDD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	106.0 µs 488.0 µs 112.0 µs 282.0 µs 670.0 µs	5.0 µs 5.0 µs 5.0 µs 5.0 µs 5.0 µs	101.0 µs 354.0 µs 112.0 µs 216.0 µs 460.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	7.2 µs 7.2 µs 13.5 µs 5.1 µs 5.1 µs	0.8 µs 0.8 µs 1.7 µs 0.8 µs 0.8 µs
LDF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	87µs+ 16µs x N	4.0 µs	10.5µs+ 3.45µs xN	2.3 µs
LDA	O: (Octal constant for address)	95.0 µs	5.0 µs	90.0 µs	4.0 µs	4.9 µs	0.8 µs
LDX	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	517.0 µs 816.0 µs — —	5.0 µs 5.0 µs — —	433.0 µs 583.0 µs — —	4.0 µs 4.0 µs — —	10.0 µs 10.0 µs 19.9 µs 19.9 µs	1.7 µs 1.7 µs 1.7 µs 1.7 µs
LDSX	K: Constant	—	—	90.0µs	4.0 µs	19.0 µs	2.3 µs
LDR	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — —	— — — —	— — — —	— — — —	30.3 µs 30.3 µs 26.6 µs 39.9 µs 39.9 µs	1.8 µs 1.8 µs 1.8 µs 1.8 µs 1.8 µs
OUT	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	26.0 µs 181.0 µs 286.0 µs 538.0 µs	5.0 µs 5.0 µs 5.0 µs 5.0 µs	15.4 µs 96.7 µs 189.4 µs 334.6 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	4.7 µs 4.7 µs 11.1 µs 11.1 µs	0.8 µs 0.8 µs 1.7 µs 1.7 µs
OUTD	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	35.0 µs 419.0 µs 296.0 µs 777.0 µs	5.0 µs 5.0 µs 5.0 µs 5.0 µs	21.4 µs 232.5 µs 196.0 µs 471.4 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	5.4 µs 5.4 µs 11.7 µs 11.7 µs	0.8 µs 0.8 µs 1.8 µs 1.8 µs
OUTF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	52µs+ 12µs x N	4.0 µs	43.8 µs+ 6.2µs x N	2.3 µs
OUTX	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	551.0 µs 950.0 µs — —	5.0 µs 5.0 µs — —	356.0 µs 574.0 µs 471.0 µs 734.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	14.1 µs 14.1 µs 23.8 µs 23.8 µs	1.8 µs 1.8 µs 1.8 µs 1.8 µs
POP	None	82.0 µs	5.0 µs	88.0 µs	4.0 µs	3.7 µs	4.0 µs

Логические команды аккумулятора

Логические команды аккумулятора		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
AND	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	101.0 µs 206.0 µs — —	5.0 µs 5.0 µs — —	66.0 µs 136.0 µs 183.4 µs 272.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	4.9 µs 4.9 µs 11.1 µs 11.1 µs	0.8 µs 0.8 µs 1.7 µs 1.7 µs
ANDD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — — 115.0 µs	— — — — 5.0 µs	66.7 µs 323.4 µs 78.7 µs 186.0 µs 459.4 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	7.4 µs 7.4 µs 12.1 µs 3.6 µs 3.6 µs	1.7 µs 1.7 µs 1.7 µs 0.8 µs 0.8 µs
ANDF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	87µs+16µs x N	4.0 µs	3.9µs+ 3.4µs x N	2.3 µs
ANDS	None	—	—	77.4 µs	4.0 µs	5.0 µs	0.7 µs
OR	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	101.0 µs 206.0 µs — —	5.0 µs 5.0 µs — —	66.0 µs 136.0 µs 183.4 µs 272.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	4.9 µs 4.9 µs 11.1 µs 11.1 µs	0.8 µs 0.8 µs 1.8 µs 1.8 µs
ORD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — — 115.0 µs	— — — — 5.0 µs	69.4 µs 328.4 µs 78.7 µs 186.0 µs 459.4 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	7.5 µs 7.5 µs 7.5 µs 3.7 µs 3.7 µs	1.8 µs 1.8 µs 1.8 µs 0.8 µs 0.8 µs
ORF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	87µs+16µs x N	4.0 µs	8.8µs+ 3.5µs x N	2.3 µs
ORS	None	—	—	77.4 µs	4.0 µs	5.0 µs	0.7 µs
XOR	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	101.0 µs 206.0 µs — —	5.0 µs 5.0 µs — —	33.0 µs 136.0 µs 183.4 µs 272.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	5.0 µs 5.0 µs 11.2 µs 11.2 µs	0.8 µs 0.8 µs 1.8 µs 1.8 µs
XORD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — — 115.0 µs	— — — — 5.0 µs	69.4 µs 328.4 µs 78.7 µs 186.0 µs 459.4 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	7.5 µs 7.5 µs 12.1 µs 3.7 µs 3.7 µs	1.8 µs 1.8 µs 1.8 µs 0.8 µs 0.8 µs
XORF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	87µs+ 16µs x N	4.0 µs	8.8 µs+ 3.5µs x N	2.3 µs
XORS	None	—	—	77.3 µs	4.0 µs	5.0 µs	0.7 µs
CMP	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	56.0 µs 162.0 µs — —	5.0 µs 5.0 µs — —	36.0 µs 106.7 µs 158.7 µs 248.7 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	6.1 µs 6.1 µs 12.4 µs 12.4 µs	0.8 µs 0.8 µs 1.8 µs 1.8 µs

Логические команды аккумулятора		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
CMPD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — — —	— — — — —	48.7 µs 312.7 µs 67.4 µs 181.4 µs 455.4 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	8.4 µs 8.4 µs 4.6 µs 13.0 µs 13.0 µs	1.8 µs 1.8 µs 0.8 µs 1.8 µs 1.8 µs
CMPF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	248µs+ 16µs x N	4.0 µs	12.4µs+ 3.5µs x N	2.3 µs
CMPR	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — — —	— — — — —	— — — — —	— — — — —	39.6 µs 39.6 µs 29.7 µs 49.2 µs 49.2 µs	1.8 µs 1.8 µs 1.8 µs 1.8 µs 1.8 µs
CMPS	None	100.0 µs	5.0 µs	99.0 µs	4.0 µs	5.8 µs	0.7 µs

Команды работы с битами

Команды работы с битами		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
SUM	None	280.0 µs	5.0 µs	183.3 µs	4.0 µs	12.1 µs	1.6 µs
SHFL	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	52µs+ 21µs x N 157µs+ 21µs x N 63µs+ 21µs x N	5.0 µs 5.0 µs 5.0 µs	33µs+ 14µs x N 103µs+ 14µs x N 43µs+ 14µs x N	4.0 µs 4.0 µs 4.0 µs	9.8 µs+ .16µs x N 9.8 µs+ .16µs x N 7.9 µs+ .16µs x N	1.8 µs 1.8 µs 1.8 µs
SHFR	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	57µs+ 21µs x N 163µs+ 21µs x N 69µs+ 21µs x N	5.0 µs 5.0 µs 5.0 µs	36µs+ 14µs x N 107µs+ 14µs x N 43µs+ 14µs x N	4.0 µs 4.0 µs 4.0 µs	9.8 µs+ .16µs x N 9.8 µs+ .16µs x N 7.9 µs+ .16µs x N	1.8 µs 1.8 µs 1.8 µs
ROTL	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	66µs+ 25µs x N 171µs+ 25µs x N 78µs+ 25µs x N	5.0 µs 5.0 µs 5.0 µs	42µs+ 17µs x N 112µs+ 17µs x N 51µs+ 1 s x N	4.0 µs 4.0 µs 4.0 µs	5.3 µs 5.3 µs 7.1 µs	1.9 µs 1.9 µs 1.9 µs
ROTR	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	69µs+ 25µs x N 174µs+ 25µs x N 81µs+ 25µs x N	5.0 µs 5.0 µs 5.0 µs	44µs+ 17µs x N 114µs+ 17µs x N 54µs+ 1 s x N	4.0 µs 4.0 µs 4.0 µs	5.2 µs 5.2 µs 7.1 µs	1.9 µs 1.9 µs 1.9 µs
ENCO	None	107.0 µs	5.0 µs	69.4 µs	4.0 µs	23.5 µs	1.6 µs
DECO	None	61.0 µs	5.0 µs	39.4 µs	4.0 µs	6.5 µs	1.6 µs

Математические команды

Команды работы с битами		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
ADD	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	308.0 µs 413.0 µs — —	5.0 µs 5.0 µs — —	203.4 µs 273.4 µs 318.0 µs 406.7 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	43.8 µs 43.8 µs 50.2 µs 50.2 µs	0.8 µs 0.8 µs 1.8 µs 1.8 µs
ADDD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	313.0 µs 694.0 µs 347.0 µs — —	5.0 µs 5.0 µs 5.0 µs — —	206.0 µs 460.7 µs 250.0 µs 374.0 µs 594.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	48.9 µs 48.9 µs 37.7 µs 53.6 µs 53.6 µs	1.8 µs 1.8 µs 0.8 µs 1.8 µs 1.8 µs
SUB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	308.0 µs 413.0 µs — —	5.0 µs 5.0 µs — —	203.4 µs 273.4 µs 317.4 µs 405.4 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	43.0 µs 43.0 µs 49.4 µs 49.4 µs	0.8 µs 0.8 µs 1.8 µs 1.8 µs
SUBD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	313.0 µs 694.0 µs 347.0 µs — —	5.0 µs 5.0 µs 5.0 µs — —	206.7 µs 460.7 µs 250.7 µs 320.0 µs 592.7 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	48.2 µs 48.2 µs 36.7 µs 52.8 µs 52.8 µs	1.8 µs 1.8 µs 0.8 µs 1.8 µs 1.8 µs
MUL	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	458.0 µs 558.0 µs 469.0 µs — —	5.0 µs 5.0 µs 5.0 µs — —	300.0 µs 370.7 µs 342.7 µs 1020.7 µs 1108.7 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	159.0 µs 159.0 µs 153.0 µs 165.0 µs 165.0 µs	1.8 µs 1.8 µs 0.8 µs 1.8 µs 1.8 µs
MULD	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — — —	— — — —	— — — —	— — — —	480.1 µs 480.1 µs 484.0 µs 484.0 µs	1.9 µs 1.9 µs 1.9 µs 1.9 µs
DIV	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	6446.0 µs 6553.0 µs 6457.0 µs — —	5.0 µs 5.0 µs 5.0 µs — —	4358.0 µs 4446.7 µs 4361.4 µs 4490.7 µs 4578.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	217.0 µs 217.0 µs 211.0 µs 224.0 µs 224.0 µs	0.8 µs 0.8 µs 0.8 µs 1.8 µs 1.8 µs
DIVD	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — — —	— — — —	4428.0 µs 4682.0 µs 4543.0 µs 4806.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	222.0 µs 222.0 µs 224.0 µs 224.0 µs	1.9 µs 1.9 µs 1.9 µs 1.9 µs
ADDB	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	143.0 µs 248.0 µs 145.0 µs — —	5.0 µs 5.0 µs 5.0 µs — —	94.0 µs 164.0 µs 100.0 µs 210.7 µs 298.7 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	11.6 µs 11.6.0 µs 17.8 µs 10.4 µs 10.4 µs	0.8 µs 0.8 µs 0.8 µs 1.8 µs 1.8 µs

Команды работы с битами		**DL430**		**DL440**		**DL450**	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
ADDBD	V:Data Reg.	—	—	90.0 µs	4.0 µs	14.2 µs	0.8 µs
	V:Bit Reg.	—	—	344.7 µs	4.0 µs	14.2 µs	0.8 µs
	K:Constant	—	—	99.4 µs	4.0 µs	10.4 µs	0.8 µs
	P:Indir. (Data)	—	—	206.7 µs	4.0 µs	18.8 µs	1.8 µs
	P:Indir. (Bit)	—	—	479.4 µs	4.0 µs	18.8 µs	1.8 µs
SUBB	V:Data Reg.	143.0 µs	5.0 µs	94.0 µs	4.0 µs	11.8 µs	0.8 µs
	V:Bit Reg.	248.0 µs	5.0 µs	164.0 µs	4.0 µs	11.8 µs	0.8 µs
	K:Constant	145.0 µs	5.0 µs	100.0 µs	4.0 µs	10.3 µs	0.8 µs
	P:Indir. (Data)	—	—	210.0 µs	4.0 µs	18.1 µs	1.8 µs
	P:Indir. (Bit)	—	—	298.0 µs	4.0 µs	18.1 µs	1.8 µs
SUBBD	V:Data Reg.	—	—	90.0 µs	4.0 µs	14.1 µs	0.8 µs
	V:Bit Reg.	—	—	344.7 µs	4.0 µs	14.1 µs	0.8 µs
	K:Constant	—	—	99.4 µs	4.0 µs	10.2 µs	0.8 µs
	P:Indir. (Data)	—	—	205.4 µs	4.0 µs	18.8 µs	1.8 µs
	P:Indir. (Bit)	—	—	478.7 µs	4.0 µs	18.8 µs	1.8 µs
MULB	V:Data Reg.	123.0 µs	5.0 µs	80.7 µs	4.0 µs	5.2 µs	0.8 µs
	V:Bit Reg.	228.0 µs	5.0 µs	150.7 µs	4.0 µs	5.2 µs	0.8 µs
	K:Constant	134.0 µs	5.0 µs	92.7 µs	4.0 µs	3.8 µs	0.8 µs
	P:Indir. (Data)	—	—	198.0 µs	4.0 µs	11.4 µs	1.8 µs
	P:Indir. (Bit)	—	—	286.0 µs	4.0 µs	11.4 µs	1.8 µs
DIVB	V:Data Reg.	4889.0 µs	5.0 µs	3261.4 µs	4.0 µs	22.8 µs	0.8 µs
	V:Bit Reg.	4995.0 µs	5.0 µs	3331.4 µs	4.0 µs	22.8 µs	0.8 µs
	K:Constant	4902.0 µs	5.0 µs	3273.4 µs	4.0 µs	21.3 µs	0.8 µs
	P:Indir. (Data)	—	—	3380.0 µs	4.0 µs	29.1 µs	1.8 µs
	P:Indir. (Bit)	—	—	3468.0 µs	4.0 µs	29.1 µs	1.8 µs
ADDR	V:Data Reg.	—	—	—	—	46.8 µs	1.8 µs
	V:Bit Reg.	—	—	—	—	46.8 µs	1.8 µs
	K:Constant	—	—	—	—	33.8 µs	1.8 µs
	P:Indir. (Data)	—	—	—	—	56.5 µs	1.8 µs
	P:Indir. (Bit)	—	—	—	—	56.5 µs	1.8 µs
SUBR	V:Data Reg.	—	—	—	—	46.8 µs	1.8 µs
	V:Bit Reg.	—	—	—	—	46.8 µs	1.8 µs
	K:Constant	—	—	—	—	33.8 µs	1.8 µs
	P:Indir. (Data)	—	—	—	—	56.5 µs	1.8 µs
	P:Indir. (Bit)	—	—	—	—	56.5 µs	1.8 µs
MULR	V:Data Reg.	—	—	—	—	44.4 µs	1.8 µs
	V:Bit Reg.	—	—	—	—	44.4 µs	1.8 µs
	K:Constant	—	—	—	—	33.8 µs	1.8 µs
	P:Indir. (Data)	—	—	—	—	54.1 µs	1.8 µs
	P:Indir. (Bit)	—	—	—	—	54.1 µs	1.8 µs
DIVR	V:Data Reg.	—	—	—	—	49.0 µs	1.8 µs
	V:Bit Reg.	—	—	—	—	49.0 µs	1.8 µs
	K:Constant	—	—	—	—	38.4 µs	1.8 µs
	P:Indir. (Data)	—	—	—	—	58.3 µs	1.8 µs
	P:Indir. (Bit)	—	—	—	—	58.3 µs	1.8 µs

Команды работы с битами		**DL430**		**DL440**		**DL450**	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
ADDBS	None	—	—	97.4 µs	4.0 µs	11.4 µs	0.7 µs
SUBBS	None	—	—	96.7 µs	4.0 µs	11.7 µs	0.7 µs
MULBS	None	—	—	92.7 µs	4.0 µs	5.1 µs	0.7 µs
DIVBS	None	—	—	3072.0 µs	4.0 µs	12.2 µs	0.7 µs
ADDF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	224µs+ 16µs x N	4.0 µs	50.1 µs+ 3.5 µs x N	2.3 µs
SUBF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	224µs+ 16µs x N	4.0 µs	49.2 µs+ 3.5 µs x N	2.3 µs
MULF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	325µs+ 8µs x N	4.0 µs	163.1 µs+ 3.4 µs x N	2.3 µs
DIVF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	4472µs+ 8µs x N	4.0 µs	20.9 µs+ 3.4 µs x N	2.3 µs
ADDS	None	321.0 µs	5.0 µs	265.0 µs	4.0 µs	46.5 µs	0.6 µs
SUBS	None	324.0 µs	5.0 µs	265.0 µs	4.0 µs	45.6 µs	0.7 µs
MULS	None	475.0 µs	5.0 µs	392.0 µs	4.0 µs	362.5 µs	0.7 µs
DIVS	None	6463.0 µs	5.0 µs	4061.0 µs	4.0 µs	501.8 µs	0.7 µs
INC	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	119.0 µs 381.0 µs — —	5.0 µs 5.0 µs — —	78.0 µs 230.0 µs 196.0 µs 372.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	22.6 µs 22.6 µs 29.0 µs 29.0 µs	0.8 µs 0.8 µs 1.9 µs 1.9 µs
DEC	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	119.0 µs 381.0 µs — —	5.0 µs 5.0 µs — —	78.0 µs 230.0 µs 210.7 µs 298.7 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	22.3 µs 22.3 µs 28.6 µs 28.6 µs	0.8 µs 0.8 µs 1.9 µs 1.9 µs
INCB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	41.0 µs 303.0 µs — —	5.0 µs 5.0 µs — —	26.0 µs 178.0 µs 143.4 µs 329.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	7.3 µs 7.3 µs 13.5 µs 13.5 µs	0.8 µs 0.8 µs 1.9 µs 1.9 µs
DECB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	41.0 µs 303.0 µs — —	5.0 µs 5.0 µs — —	26.0 µs 178.0 µs 142.0 µs 330.7 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	7.3 µs 7.3 µs 13.5 µs 13.5 µs	0.8 µs 0.8 µs 1.9 µs 1.9 µs

Команды работы с битами		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
SQRT	None	—	—	—	—	197.6 µs	1.6 µs
SQRTR	None	—	—	—	—	61.96 µs	1.6 µs
SIN	None	—	—	—	—	274.7 µs	1.6 µs
SINR	None	—	—	—	—	110.96 µs	1.6 µs
COS	None	—	—	—	—	280.4 µs	1.6 µs
COSR	None	—	—	—	—	116.0 µs	1.6 µs
TAN	None	—	—	—	—	294.1 µs	1.6 µs
TANR	None	—	—	—	—	145.2 µs	1.6 µs
ASIN	None	—	—	—	—	349.6 µs	1.6 µs
ASINR	None	—	—	—	—	230.5 µs	1.6 µs
ACOS	None	—	—	—	—	355.5 µs	1.6 µs
ACOSR	None	—	—	—	—	237.8 µs	1.6 µs
ATAN	None	—	—	—	—	274.9 µs	1.6 µs
ATANR	None	—	—	—	—	155.5 µs	1.6 µs

Команды преобразования чисел

Команды преобразования чисел		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
BIN	None	409.0 µs	5.0 µs	107.4 µs	4.0 µs	127.3 µs	1.6 µs
BCD	None	409.0 µs	5.0 µs	152.0 µs	4.0 µs	125.3 µs	1.6 µs
INV	None	46.0 µs	5.0 µs	15.0 µs	4.0 µs	2.9 µs	1.6 µs
BCDCPL	None	—	—	232.7µs	4.0 µs	35.3µs	1.6 µs
ATH	V:Data Reg. (1 word) V:Bit Reg. (1 word)	— —	— —	1494.7µs 1910.0µs	4.0 µs 4.0 µs	14.3 µs 14.3 µs	1.9 µs 1.9 µs
HTA	V:Data Reg. (1 word) V:Bit Reg. (1 word)	— —	— —	1489.4µs 1960.7µs	4.0 µs 4.0 µs	14.3 µs 14.3 µs	1.9 µs 1.9 µs
SEG	None	101.0 µs	5.0 µs	64.0 µs	4.0 µs	6.9 µs	1.6 µs
GRAY	None	—	—	176.0 µs	4.0 µs	69.1 µs	1.6 µs
SFLDGT	None	—	—	224.0 µs	4.0 µs	21.6 µs	1.6 µs
RAD	None	—	—	—	—	175.0 µs	1.6 µs
RADR	None	—	—	—	—	42.5 µs	1.6 µs
DEG	None	—	—	—	—	176.2 µs	1.6 µs
DEGR	None	—	—	—	—	42.4 µs	1.6 µs
ITOR	None	—	—	—	—	11.1 µs	1.6 µs
RTOI	None	—	—	—	—	34.2 µs	1.6 µs

Команды работы с таблицами

Команды работы с таблицами		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
FILL	V:Data Reg. — V:Bit Reg. K:Constant	— — — 	— — — 	596µs+ 74µs x N 904µs+ 74µs x N 608µs+ 74µs x N	4.0µs 4.0µs 4.0µs	13.6 µs+ 4.4µs x N 13.6µs+ 1.4µs x N 17.3µs+ 4.2µs x N	1.9 µs 1.9 µs 1.9 µs
FIND	V:Data Reg. — V:Bit Reg. K:Constant	— — — 	— — — 	372µs+ 85µs x N 442µs+ 85µs x N 384µs+ 85µs x N	4.0µs 4.0µs 4.0µs	23.0µs+ 3.1µs x N 23.0µs+ 3.1µs x N 20.2µs+ 3.1µs x N	1.9 µs 1.9 µs 1.9 µs
FDGT	V:Data Reg. V:Bit Reg. K:Constant	— — —	— — —	1028.6µs 1098.7µs 1066.7µs	4.0µs 4.0µs 4.0µs	25.9 µs 25.9 µs 26.2 µs	1.9 µs 1.9 µs 1.9 µs
MOV	V:Data Reg. V:Bit Reg.	— —	— —	1767.0µs 3188.0µs	4.0µs 4.0µs	24.1 µs 24.1 µs	1.9 µs 1.9 µs
TTD	V	—	—	748.7µs	4.0µs	29.8 µs	1.9 µs
RFB	V	—	—	747.4µs	4.0µs	21.0 µs	1.9 µs
STT	V K	— —	— —	722.7 µs 784.0 µs	4.0µs 4.0µs	27.7 µs 24.9 µs	1.9 µs 1.9 µs
RFT	V	—	—	1548.0µs	4.0µs	22.1 µs	1.9 µs
ATT	V K	— —	— —	2725.4µs 2734.4µs	4.0µs 4.0µs	25.0 µs 22.1 µs	1.9 µs 1.9 µs
MOVMC	Move V:Data Reg. to MC Move V:Bit Reg. to MC Move from MC to V:Data Reg. Move from MC to V:Bit Reg.	— — — —	— — — —	1332.7µs 2215.4µs 818.0µs 1394.0µs	4.0µs 4.0µs 4.0µs 4.0µs	6.0 µs 6.0 µs 22.7 µs 22.7 µs	1.9 µs 1.9 µs 1.9 µs 1.9 µs
LDLBL	K	—	—	62.0 µs	4.0 µs	7.6 µs	1.9 µs

Команды управления процессором

Команды управления процессором		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
NOP	None	0	0	0	0	1 µs	1 µs
END	None	15.5 µs	15.5 µs	11.6 µs	11.6 µs	8.5 µs	8.5 µs
STOP	None	26.0 µs	5.0 µs	17.4 µs	4.0 µs	6.7 µs	1.6 µs
BREAK	None	—	—	717.0 µs	4.0 µs	15.3 µs	1.6 µs
RSTWT	None	22.0 µs	5.0 µs	14.7 µs	4.0 µs	4.0 µs	1.6 µs

Команды управления программой

Команды управления программой		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
GOTO	K	—	—	18.7 µs	4.0 µs	5.1 µs	4.6 µs
LBL	K	—	—	0 µs	0 µs	0 µs	0 µs
FOR	V, K	—	—	32.7 µs	4.0 µs	59.6 µs	7.1 µs
NEXT	None	—	—	57.4 µs	4.0 µs	80.0 µs	0 µs
GTS	K	—	—	38.7 µs	4.0 µs	20.3 µs	5.2 µs
SBR	K	—	—	0 µs	0 µs	0.8 µs	0 µs
RTC	None	—	—	37.4 µs	4.0 µs	6.1 µs	6.1 µs
RT	None	—	—	35.4 µs	4.0 µs	5.1 µs	0 µs
MLS	K (1-7)	16.3 µs	16.3 µs	16.6 µs	16.6 µs	2.1 µs	2.1 µs
MLR	K (0-7)	20.0 µs	20.0 µs	13.4 µs	13.4 µs	2.0 µs	2.0 µs

Команды прерывания

Команды прерывания		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
ENI	None	12.0 µs	5.0 µs	8.0 µs	4.0 µs	9.0 µs	1.6 µs
DISI	None	12.0 µs	5.0 µs	8.0 µs	4.0 µs	11.7 µs	1.6 µs
INT	0 (0-7)	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs
IRTC	None	180.0 µs	5.0 µs	121.4 µs	4.0 µs	0.7 µs	0.7 µs
IRT	None	180.0 µs	—	120.0 µs	—	1.4 µs	—

Команды RLL^{PLUS}

Команды RLL^{PLUS}		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
ISG	S	44.0 µs	42.0 µs	22.0 µs	20.0 µs	22.2 µs	21.0 µs
SG	S	44.0 µs	42.0 µs	22.0 µs	20.0 µs	22.2 µs	21.2 µs
JMP	S	14.0 µs	5.0 µs	10.7 µs	4.0 µs	20.7 µs	4.1 µs
NJMP	S	15.0 µs	5.0 µs	12.7 µs	4.0 µs	21.3 µs	4.5 µs
CV	S	—	—	30.0 µs	7.0 µs	13.8 µs	13.8 µs
CVJMP	S (N stages)	—	—	20 µs + 6 µs x N	7.0 µs	12.3 µs	12.3 µs
BCALL	C	—	—	12.0 µs	10.0 µs	19.4 µs	19.4 µs
BLK	C	—	—	21.0 µs	14.0 µs	18.3 µs	15.6 µs
BEND	None	—	—	6.0 µs	0 µs	7.8 µs	0 µs

Команды интеллектуального ввода/вывода

Команды интеллектуального ввода/вывода		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
RD	V:Data Reg.	1130+ 123xN µs	5.0 µs	754+ 82xN µs	4.0 µs	109.6 µs	1.9 µs
	V:Bit Reg.	1150+ 172xN µs	5.0 µs	766+ 105xN µs	4.0 µs	109.6 µs	1.9 µs
WT	V:Data Reg.	1840+ 150xN µs	5.0 µs	896+ 110xN µs	4.0 µs	109.8 µs	1.9 µs
	V:Bit Reg.	1875+ 180xN µs	5.0 µs	917+ 120xN µs	4.0 µs	109.8 µs	1.9 µs

Сетевые команды

Сетевые команды		DL430		DL440		DL440	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
RX	X, Y, C, T, CT, GX,, GY SP, S	760.0 µs	5.0 µs	488.0 µs	4.0 µs	111.8 µs	2.3 µs
	V:Data Reg.	760.0 µs	5.0 µs	488.0 µs	4.0 µs	111.8 µs	2.3 µs
	V:Bit Reg.	780.0 µs	5.0 µs	488.0 µs	4.0 µs	111.8 µs	2.3 µs
WX	X, Y, C, T, CT, GX,, GY SP, S	Source	5.0 µs	Source	4.0 µs	111.8 µs	2.3 µs
	V:Data Reg.	755+ 12xN µs	5.0 µs	503+ 8xN µs	4.0 µs	111.8 µs	2.3 µs
	V:Bit Reg.		5.0 µs		4.0 µs	111.8 µs	2.3 µs
PRINT	ASCII	—	—	—	—	104.0 µs	2.2 µs

Команды работы с сообщениями

Команды работы с сообщениями		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
FAULT	V:Data Reg.	164.0 µs	5.0 µs	107.0 µs	4.0 µs	107.3 µs	2.3 µs
	V:Bit Reg.	266.0 µs	5.0 µs	178.0 µs	4.0 µs	107.3 µs	2.3 µs
	K:Constant	158.0 µs	5.0 µs	158.7 µs	4.0 µs	98.1 µs	2.3 µs
DLBL	K	—	—	0 µs	0 µs	0 µs	0 µs
NCON	K	—	—	0 µs	0 µs	0 µs	0 µs
ACON	K	—	—	0 µs	0 µs	0 µs	0 µs

Команды барабанного командоаппарата

Команды барабанного командоаппарата		DL430		DL440		DL450	
Команда	**Разрешенные типы данных**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**	**Исполн.**	**Не испол.**
DRUM	CT	—	—	—	—	340.0 µs	62.6 µs
EDRUM	CT	—	—	—	—	243.0 µs	100.0 µs
MDRMD	CT	—	—	—	—	206.0 µs	142.00 µs
MDRMW	CT	—	—	—	—	150.0 µs	94.00 µs

Специальные реле

D

В этой главе...

— Специальные реле запуска и реального времени
— Реле состояния процессора
— Реле оперативного контроля системы
— Реле состояния аккумулятора
— Реле оперативного контроля аккумулятора

Реле запуска и реального времени

SP0	Первое сканирование	Включается только при первом сканировании после включении питания или при переходе из программного режима в рабочий. Реле отключается на втором сканировании. Полезно для функций, выполняемых только при запуске программы.
SP1	Всегда ВКЛЮЧЕНО	Обеспечивает контакт, гарантирующий выполнение команды при каждом сканировании.
SP3	Таймер на 1 мин	Включено в течение 30 сек и отключено в течение 30 сек.
SP4	Таймер на 1 сек	Включено в течение 0,5 сек и отключено в течение 0,5 сек.
SP5	Таймер на 100 мс	Включено в течение 50 мс и отключено в течение 50 мс.
SP6	Таймер на 50 мс	Включено в течение 25 мс и отключено в течение 25 мс.
SP7	Чередующееся сканирование	Включается через одно сканирование.

Реле состояния процессора

SP11	Форсированный рабочий режим	Включено всегда, когда переключатель процессора находится в положении RUN.
SP12	Терминальный рабочий режим	Включено, когда процессор находится в рабочем режиме, а переключатель в TERM.
SP13	Режим Test Run	Включено, когда процессор находится в режиме Test Run (тестирования в рабочем режиме).
SP14	Реле прерывания 1	Включено, когда выполняется команда BREAK. Оно выключено, когда процессор находится в любом другом режиме.
SP15	Режим испытания программ	Включено, когда процессор находится в режиме испытания программ.
SP16	Терминальный программный режим	Включено, когда переключатель процессора находится в положении TERM и процессор находится в программном режиме.
SP17	Реле форсированного останова	Включено всегда, когда переключатель процессора находится в положении STOP.
SP21	Реле прерывания 2	Включено, когда выполняются команды BREAK. Выключено, когда режим процессора изменяется на рабочий (RUN).
SP22	Разрешенное прерывание	Включено, когда прерывание разрешено с использованием команды ENI.
SP25	Блокирование батареи процессора	Включается, когда сигнализация падения напряжения батареи процессора блокируется DIP-переключателем 1 на задней панели процессора.
SP26	Блокирование обновления ввода/вывода	Включается (на DL440/DL450) прикладной программой, поэтому "замораживает" состояние ввода/вывода.
SP27	Селективное блокирование обновления ввода/вывода	Реле включается (на DL440/DL450) прикладной программой. Включение реле прекращает обновление ввода/вывода на последнем состоянии для модулей, для которых обнаружено отсутствие клеммного блока.
SP30	Состояние DIP-переключателя 1	(На DL440/DL450) отслеживает состояние Включено/Выключено DIP-переключателя 1 на задней панели процессора.
SP31	Состояние DIP-переключателя 2	(На DL440/DL450) отслеживает состояние Включено/Выключено DIP-переключателя 2 на задней панели процессора.
SP32	Состояние DIP-переключателя 3	(На DL440/DL450) отслеживает состояние Включено/Выключено DIP-переключателя 3 на задней панели процессора.
SP33	Состояние DIP-переключателя 4	(На DL440/DL450) отслеживает состояние Включено/Выключено DIP-переключателя 4 на задней панели процессора.
SP37	Ошибка при контроле сканирования	(На DL450) это реле будет включаться, если фактическое время сканирования будет больше установленного в режимах с фиксированным или ограниченным временем цикла сканирования. Контакты реле могут быть полезны при поиске программной ошибки.

Реле оперативного контроля системы

SP40	Критическая ошибка	Включено, когда возникает критическая ошибка, например, потеря связи с точками ввода/вывода.
SP41	Предупреждение	Включено, когда возникает некритическая ошибка, например, низкое напряжение батареи.
SP43	Низкое напряжение батареи (DL440/DL450)	Включено, когда напряжение либо на батареи картриджа памяти, либо процессора низкое. После выявления проверьте V7757 для определения точного кода ошибки.
SP44	Ошибка программной памяти	Включено, когда возникает ошибка памяти, например, ошибка четности памяти или отказ картриджа памяти.
SP45	Ошибка ввода/вывода	Включено, когда возникает ошибка ввода/вывода. Например, перегорел предохранитель или недостаточное питание 24 В.
SP46	Коммуникационная ошибка	Включено, когда обнаружена коммуникационная ошибка в любом порту процессора.
SP47	Ошибка в конфигурации ввода/вывода	Включено, если обнаружена ошибка в конфигурации ввода/вывода. В процессоре должна быть установлена проверка конфигурации ввода/вывода до того, как это реле начнет работать.
SP50	Команда FAULT	Включается при выполнении команды FAULT.
SP51	Реле превышения времени выполнения мат. операции	Включено, если процессор исчерпал время выполнения математической функции.
SP52	Грамматическая ошибка	Включено, когда обнаружена грамматическая ошибка либо при работе процессора, либо при проверке синтаксиса. В V7755 хранится точный код ошибки.
SP53	Указатель таблицы/ математика	Включается, если найдена ошибка при математической операции либо ошибка в указателе таблицы.
SP54	Ошибка связи	Включается, когда команды RX, WX, RD, WT выполняются с неправильным параметром.
SP56	Переполнение при выполнении табличной команды	Включается, если выполняется табличная команда с указателем, а значение указателя находится вне границ таблицы.

Реле состояние аккумулятора

SP53	Указатель таблицы/ математика	Включается, если найдена ошибка при математической операции либо ошибка в указателе таблицы.
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	Значение больше, чем	Включено, когда значение в аккумуляторе больше, чем значение в команде.
SP63	Нуль	Включено, когда в результате выполнения команды значение в аккумуляторе равно нулю.
SP64	Отрицательный перенос половины	Включено, когда команда 16-битового вычитания приводит к заимствованию.
SP65	Отрицательный перенос	Включено, когда команда 32-битового вычитания приводит к заимствованию.
SP66	Перенос половины	Включено, когда команда 16-битового сложения приводит к переносу.
SP67	Перенос	Включено, когда команда 32-битового сложения приводит к переносу.
SP70	Знак (отрицательный)	Включается всякий раз, когда значение в аккумуляторе становится отрицательным.
SP71	Ошибка в ссылке указателя	Включается, когда V-память, определенная указателем (P), не верна.
SP72	Плавающая точка	(На DL450) включается, когда численное значение в аккумуляторе является числом с плавающей точкой.
SP73	Переполнение	Включается, если происходит переполнение аккумулятора, когда сложение со знаком или вычитание приводит к неправильному биту знака.
SP74	Потеря значимости	(На DL450) включается, когда математическая операция с плавающей точкой приводит к ошибке потери значимости.
SP75	Ошибка в данных	Включается, когда выполняется команда в двоично-десятичном формате, но встретилось число не в двоично-десятичном формате.
SP76	Загрузка нуля	Включается, когда команда загружает в аккумулятор значение нуль.

Реле оперативного контроля коммуникаций

Реле оперативного контроля коммуникаций нумеруются попарно в соответствии с номерами портов процессора или положениями слотов в каркасе. Имеются два типа реле:

- Модуль/порт занят: Оно включается, когда порт или коммуникационный модуль в соответствующем слоте и каркасе занят на передачу или прием данных. Вы должны использовать это реле с командами RX и WX, чтобы не допустить попытки выполнить эти команды, когда модуль занят.
- Ошибка связи: Оно включается, когда обнаруживается коммуникационная ошибка. Эта ошибка автоматически стирается при выполнении другой команды RX и WX.

Все процессоры DL405		**Только процессор DL40**					
		SP112	CPU port busy Port 1	SP114	CPU port busy Port 2	SP116	CPU port busy Port 3
		SP113	Ошибка связи Port 1	SP115	Ошибка связи Port 2	SP117	Ошибка связи Port 3
Локальный каркас		**Каркас расширения 1**		**Каркас расширения 2**		**Каркас расширения 3**	
SP120	Модуль занят Слот 0	SP140	Модуль занят Слот 0	SP160	Модуль занят Слот 0	SP200	Модуль занят Слот 0
SP121	Ошибка связи Слот 0	SP141	Ошибка связи Слот 0	SP161	Ошибка связи Слот 0	SP201	Ошибка связи Слот 0
SP122	Модуль занят Слот 1	SP142	Модуль занят Слот 1	SP162	Модуль занят Слот 1	SP202	Модуль занят Слот 1
SP123	Ошибка связи Слот 1	SP143	Ошибка связи Слот 1	SP163	Ошибка связи Слот 1	SP203	Ошибка связи Слот 1
SP124	Модуль занят Слот 2	SP144	Модуль занят Слот 2	SP164	Модуль занят Слот 2	SP204	Модуль занят Слот 2
SP125	Ошибка связи Слот 2	SP145	Ошибка связи Слот 2	SP165	Ошибка связи Слот 2	SP205	Ошибка связи Слот 2
SP126	Модуль занят Слот 3	SP146	Модуль занят Слот 3	SP166	Модуль занят Слот 3	SP206	Модуль занят Слот 3
SP127	Ошибка связи Слот 3	SP147	Ошибка связи Слот 3	SP167	Ошибка связи Слот 3	SP207	Ошибка связи Слот 3
SP130	Модуль занят Слот 4	SP150	Модуль занят Слот 4	SP170	Модуль занят Слот 4	SP210	Модуль занят Слот 4
SP131	Ошибка связи Слот 4	SP151	Ошибка связи Слот 4	SP171	Ошибка связи Слот 4	SP211	Ошибка связи Слот 4
SP132	Модуль занят Слот 5	SP152	Модуль занят Слот 5	SP172	Модуль занят Слот 5	SP212	Модуль занят Слот 5
SP133	Ошибка связи Слот 5	SP153	Ошибка связи Слот 5	SP173	Ошибка связи Слот 5	SP213	Ошибка связи Слот 5
SP134	Модуль занят Слот 6	SP154	Модуль занят Слот 6	SP174	Модуль занят Слот 6	SP214	Модуль занят Слот 6
SP135	Ошибка связи Слот 6	SP155	Ошибка связи Слот 6	SP175	Ошибка связи Слот 6	SP215	Ошибка связи Слот 6
SP136	Модуль занят Слот 7	SP156	Модуль занят Слот 7	SP176	Модуль занят Слот 7	SP216	Модуль занят Слот 7
SP137	Ошибка связи Слот 7	SP157	Ошибка связи Слот 7	SP177	Ошибка связи Слот 7	SP217	Ошибка связи Слот 7

Веса изделий DL405

E

В этой главе...

— Таблица весов изделий

Таблица весов изделий

Процессоры	Вес
D4-430	800g
D4-440	878g
D4-450	844g
D4-440DC-1	890g
D4-440DC-2	890g
Картриджи памяти	
D4-RAM-1	42g
D4-RAM-2	138g
D4-RNB	38g
D4-UV-1	60g
D4-UV-2	60g
D4-EE-2	40g
Блоки расширени	
D4-EX	644g
D4-EXDC	660g
D4-EXDC-2	660g
Каркасы ввода/вывода	
D4-04B, -1, -04BNX	660g
D4-06B, -1 -06BNX	830g
D4-08B,-1, -08BNX	990g
Входные модули постоянного тока	
D4-08ND3S	250g
D4-16ND2	250g
D4-16ND2F	250g
D4-32ND3-1	190g
D4-32ND3-2	190g
D4-64ND2	220g
Входные модули переменного тока	
D4-08NA	240g
D4-16NA	270g
Входные модули переменного/ постоянного тока	
D4-16NE3	250g
F4-08NE3S	256g

Выходные модули постоянного тока	Вес
D4-08TD1	240g
F4-08TD1S	282g
D4-16TD1	270g
D4-16TD2	270g
D4-32TD1	190g
D4-32TD2	190g
D4-64TD1	210g
Выходные модули переменного тока	
D4-08TA	330g
D4-16TA	350g
Релейные выходные модули	
D4-08TR	260g
F4-08TRS-1	374g
F4-08TRS-2	390g
D4-16TR	310g
Аналоговые модули	
D4-04AD	270g
F4-04AD	300g
F4-04ADS	326g
F4-08AD	312g
D4-02DA	260g
F4-04DA	260g
F4-04DA-1	262g
F4-04DA-2	264g
Удаленный ввод/вывод	
D4-RM	228g
D4-RS	767g
D4-RSDC	760g
D4-SM	222g
D4-SS-88	324g
D4-SS-106	334g
D4-SS-16T	340g
D4-SS-16N	340g

Коммуникационные и сетевые модули	
D4-DCM	233g
F4-MAS-MB	252g
F4-SLV-MB	252g
F4-SLV-TW	264g
F4-SDN	258g
Со-процессоры™	
F4-CP128-1	252g
F4-CP128-2	252g
F4-CP512	258g
F4-CP128-T	281g
F4-CP128-R	278g
Специальные модули	
D4-INT	250g
D4-HSC	350g
F4-16PID	207g
F4-8MPI	1350g
D4-16SIM	250g
F4-SDS	211g
F4-4LTC	361g
D4-FILL	112g
Программирование	
D4-HPP	357g