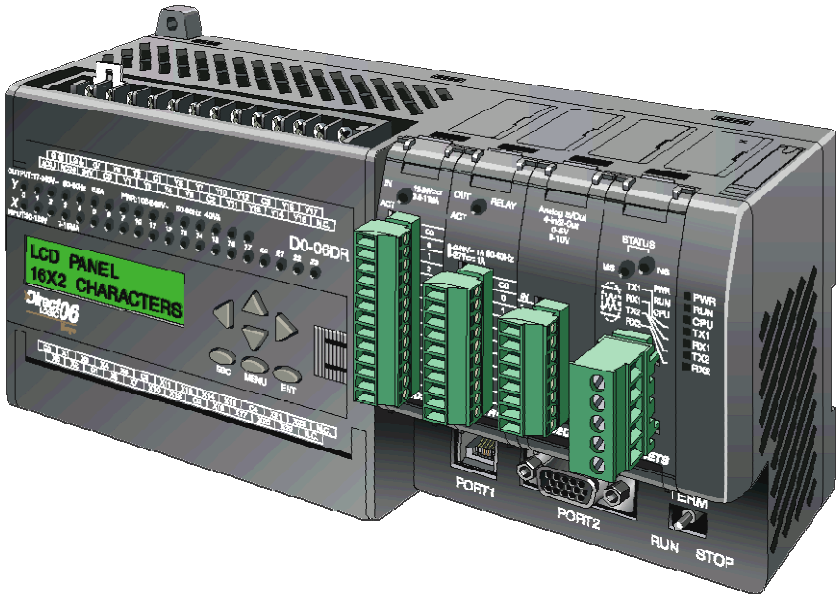
Аппаратные и программные средства АСУ ТП
тел.: (495) 105.77.98
E-mail: info@plcsystems.ru
Сайт: www.plcsystems.ru

Руководство пользователя контроллера DL06

Номер руководства D0-06USER-M-RUS
Часть 1



Книга 1.

Оглавление.

1.НАЧАЛЬНЫЕ СВЕДЕНИЯ

ВВЕДЕНИЕ. ... **1—2**
Цели данного руководства ... 1—2
Дополнительные руководства ... 1—2
Техническая поддержка ... 1—2
ИСПОЛЬЗУЕМЫЕ СОГЛАШЕНИЯ ... **1—3**
Ключевые темы каждой главы ... 1—3
ОБЗОР МИКРОКОНТРОЛЛЕРА DL06 ... **1—4**
Возможности микроконтроллеров DL06 ... 1—4
МЕТОДЫ ПРОГРАММИРОВАНИЯ ... **1—4**
Пакет программирования DirectSOFT32 под Windows ... 1—4
Ручной программатор ... 1—5
СХЕМА БЫСТРОГО ВЫБОРА ПОДСИСТЕМЫ ВВОДА/ВЫВОДА. ... **1—5**
БЫСТРЫЙ СТАРТ ... **1—6**
ШАГИ ПО УСПЕШНОМУ СОЗДАНИЮ СИСТЕМЫ АВТОМАТИКИ ... **1—10**
ВОПРОСЫ И ОТВЕТЫ ПО МИКРОКОНТРОЛЛЕРАМ DL06 ... **1—12**

2.УСТАНОВКА, ЭЛЕКТРОМОНТАЖ И СПЕЦИФИКАЦИИ

ИНСТРУКЦИЯ ПО ТЕХНИКЕ БЕЗОПАСНОСТИ ... **2—2**
Планирование безопасности ... 2—2
Три уровня защиты ... 2—2
Четкая последовательность выключения ... 2—3
Отключение питания системы ... 2—3
Аварийное отключение ... 2—3
ПЕРЕДНЯЯ ПАНЕЛЬ DL06 ... **2—4**
Съемный клеммник ... 2—5
МОНТАЖ ... **2—6**
Размеры устройства ... 2—6
Монтажный шкаф ... 2—6
План размещения в шкафу ... 2—7
Использование монтажных реек ... 2—8
Параметры окружающей среды ... 2—9
Соответствие инструкциям различных ведомств ... 2—9
ПОДКЛЮЧЕНИЕ ПИТАНИЯ ... **2—10**
Установка предохранителя в цепи подачи питания ... 2—10
Внешний источник питания ... 2—11
Размещение электропроводки ... 2—11
Защита цепей ввода и вывода с помощью предохранителей ... 2—12
Нумерация точек ввода/вывода ... 2—12
ОСНОВНЫЕ ПРИНЦИПЫ ЭЛЕКТРОМОНТАЖА СИСТЕМЫ ... **2—13**
Изоляционные барьеры ПЛК ... 2—13
Подключение панелей оператора ... 2—14
Подключение устройств программирования ... 2—14
Понятия приемник/ источник ... 2—15
Понятие «Общего полюса» входа/выхода ... 2—16
Подключение точек ввода/вывода постоянного тока к полупроводниковым полевым устройствам ... 2—17
Полупроводниковые входные датчики ... 2—17
Полупроводниковая выходная нагрузка ... 2—17
Способы подключения реле ... 2—19
Подавление бросков напряжения в цепи с индуктивной нагрузкой ... 2—20
Продление срока службы контактов реле ... 2—21
Подключение входов постоянного тока ... 2—22
Подключение выходов постоянного тока ... 2—23
Подключение высокоскоростных входов/выходов ... 2—24
СЛОВАРЬ ТЕРМИНОВ В СПЕЦИФИКАЦИИ ... **2—25**

СХЕМЫ ПОДКЛЮЧЕНИЯ И СПЕЦИФИКАЦИИ. ..... **2—26**
Схема соединений входов/выходов D0- 06AA ..... 2—26
Схема соединений входов/выходов D0- 06AR ..... 2—28
Схема соединений входов/выходов D0- 06DA ..... 2—30
Схема соединений входов/выходов D0- 06DD1 ..... 2—32
Схема соединений входов/выходов D0- 06DD2 ..... 2—34
Схема соединений входов/выходов D0- 06DR ..... 2—36
Схема соединений входов/выходов D0- 06DD1-D ..... 2—38
Схема соединений входов/выходов D0- 06DD2-D ..... 2—3840
Схема соединений входов/выходов D0- 06DR-D ..... 2—42

3.ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ВЫСОКОСКОРОСТНОГО ВВОДА И ИМПУЛЬСНОГО ВЫХОДА

ВВЕДЕНИЕ ..... **3—2**
Встроенное решение по управлению движением ..... 3—2
Возможность использования функций высокоскоростного ввода/вывода (HSIO) ..... 3—2
Специализированная схема высокоскоростного ввода/вывода ..... 3—3
Схемы подключения для каждого режима высокоскоростного ввода/вывода ..... 3—3
ВЫБОР РАБОЧЕГО РЕЖИМА ВЫСОКОСКОРОСТНОГО ВВОДА/ВЫВОДА ..... **3—4**
Шесть режимов высокоскоростного ввода/вывода ..... 3—4
Режим по умолчанию ..... 3—5
Конфигурирование режима высокоскоростного ввода/вывода ..... 3—6
Конфигурирование входов X0 – X3 ..... 3—6
РЕЖИМ 10: ВЫСОКОСКОРОСТНОЙ СЧЕТЧИК ..... **3—7**
Назначение ..... 3—7
Функциональная блок-схема ..... 3—7
Схема подключения ..... 3—8
Сопряжение с выходами счетчика ..... 3—8
Настройка на режим 10 ..... 3—9
Заданные значения и специальные реле ..... 3—9
Абсолютные и инкрементальные предустановленные значения ..... 3—10
Начальная ячейка заданных данных ..... 3—11
Использование менее 24 заданных значений ..... 3—11
Номера эквивалентных реле ..... 3—12
Расчет заданных значений ..... 3—13
Конфигурирование входа X ..... 3—14
Написание управляющей программы ..... 3—15
Пример программы 1: счетчик без заданных значений ..... 3—16
Пример программы 2: счетчик с заданными значениями ..... 3—18
Пример программы 3: счетчик с перезагрузкой ..... 3—21
Руководство по поиску неисправностей в режиме 10 ..... 3—23
РЕЖИМ 20: РЕВЕРСИВНЫЙ СЧЕТЧИК ..... **3—24**
Назначение ..... 3—24
Функциональная блок-схема ..... 3—24
Квадратурные сигналы энкодера ..... 3—25
Схема подключения ..... 3—25
Настройка на режим 20 ..... 3—27
Заданные значения и специальные реле ..... 3—27
Конфигурирование входа X ..... 3—28
Режим 20 реверсивного счетчика ..... 3—28
Написание управляющей программы ..... 3—29
Пример программы 1: Квадратурный счетчик с прерыванием. ..... 3—30
Пример программы 2: Реверсивный счетчик со стандартными входами. ..... 3—32
Пример программы 3: Квадратурный счетчик ..... 3—34
Руководство по поиску неисправностей в режиме 20 ..... 3—37
РЕЖИМ 30: ИМПУЛЬСНЫЙ ВЫХОД ..... **3—38**
Назначение ..... 3—38
Функциональная блок-схема ..... 3—39
Схема подключения ..... 3—40
Сопряжение с выходами привода ..... 3—40
Характеристики профилей движения ..... 3—41
Конфигурирование физического ввода/вывода ..... 3—41
Функции логического ввода/вывода ..... 3—41
Настройка для режима 30 ..... 3—42
Регистр выбора профиля/ скорости ..... 3—43
Таблица параметров профиля ..... 3—43
Автоматический трапецеидальный профиль ..... 3—43

Ступенчатый трапецеидальный профиль ... 3—44
Выбор типа профиля ... 3—45
Определение автоматического трапецеидального профиля ... 3—45
Определение ступенчатого трапецеидального профиля ... 3—46
Определение профиля управления скоростью ... 3—46
Работа с автоматическим трапецеидальным профилем ... 3—47
Пример 1: Автоматический трапецеидальный профиль без внешнего прерывания ... 3—48
Перезагрузка значения положения ... 3—49
Пример 2: Автоматический трапецеидальный профиль с внешним прерыванием ... 3—50
Пример 3: Автоматический трапецеидальный профиль с поиском исходного положения ... 3—53
Работа с ступенчатым трапецеидальным профилем ... 3—55
Пример 4: Ступенчатый трапецеидальный профиль ... 3—56
Работа с профилем скорости ... 3—59
Пример 5: Профиль скорости ... 3—60
Коды ошибок автоматического трапецеидального профиля ... 3—62
Руководство по поиску неисправностей в режиме 30 ... 3—62
РЕЖИМ 40: ВЫСОКОСКОРОСТНЫЕ ПРЕРЫВАНИЯ ... 3—64
Назначение ... 3—64
Функциональная блок-схема ... 3—64
Настройка на режим 40 ... 3—65
Прерывания и программа релейной логики ... 3—65
Временные параметры внешнего прерывания ... 3—66
Параметры прерывания по времени ... 3—66
Конфигурирование входа X/ прерывания по времени ... 3—66
Пример 1: Внешние прерывания ... 3—67
Пример 2: Прерывание по времени ... 3—68
РЕЖИМ 50: ВХОД С ЗАЩЕЛКОЙ ИМПУЛЬСОВ ... 3—69
Назначение ... 3—69
Функциональная блок-схема ... 3—69
Временные параметры защелки импульсов ... 3—69
Настройка на режим 50 ... 3—70
Конфигурирование входа X ... 3—72
Пример: импульсная защелка ... 3—73
РЕЖИМ 60: ДИСКРЕТНЫЕ ВХОДЫ С ФИЛЬТРОМ ... 3—74
Назначение ... 3—74
Функциональная блок-схема ... 3—74
Временные параметры входных фильтров ... 3—74
Настройка на режим 60 ... 3—75
Конфигурирование входа X ... 3—75
Пример: фильтрованные входы ... 3—76

4.ХАРАКТЕРИСТИКИ И РАБОТА ПРОЦЕССОРА

ВВЕДЕНИЕ ... 4—2
Особенности процессора DL06 ... 4—2
ОБЩИЕ ХАРАКТЕРИСТИКИ ПРОЦЕССОРА ... 4—3
ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ АППАРАТНЫХ СРЕДСТВ ПРОЦЕССОРА ... 4—4
Назначение контактов коммуникационных портов ... 4—4
Подсоединение устройств для программирования ... 4—5
Информация о настройках процессора ... 4—5
Индикаторы состояния ... 4—6
Функции переключателя режимов ... 4—6
Изменение режима работы ПЛК DL06 ... 4—7
Режим работы при включении питания ... 4—7
ИСПОЛЬЗОВАНИЕ БАТАРЕЙНОГО ПИТАНИЯ ... 4—8
Батарейное питание ... 4—8
Вспомогательные функции ... 4—9
Стирание существующих программ ... 4—9
Инициализация системной памяти ... 4—9
Установка областей сохранения памяти ... 4—10
Защита с помощью пароля ... 4—11
РАБОТА ПРОЦЕССОРА ... 4—12
Операционная система процессора DL06 ... 4—12
Работа в программном режиме ... 4—13
Работа в рабочем режиме ... 4—13
Чтение входов ... 4—14
Обслуживание периферийных устройств и форсирование ввода/вывода ... 4—14
Шина связи процессора ... 4—15

Обновление часов, специальных реле и специальных регистров ... 4—15
Выполнение прикладных программ ... 4—16
Решение уравнений контура ПИД-регулятора ... 4—16
Запись выходных данных ... 4—17
Запись выходов в специальные модули ввода/вывода ... 4—17
Диагностика ... 4—17
Время отклика ввода/вывода ... 4—17
Важно ли быстродействие для Вашего приложения? ... 4—17
Нормальное минимальное время отклика ввода/вывода ... 4—18
Нормальное максимальное время отклика ввода/вывода ... 4—18
Улучшенное время отклика ... 4—19
Анализ времени сканирования процессора ... 4—20
Чтение входов ... 4—20
Запись выходов ... 4—20
Обслуживание периферийных устройств ... 4—21
Связь по шине процессора ... 4—21
Обновление часов/календаря, специальных реле, специальных регистров ... 4—21
Выполнение прикладной программы ... 4—22
Системы счисления в ПЛК ... 4—23
Ресурсы ПЛК ... 4—23
V-память ... 4—24
Двоично-десятичные числа ... 4—24
Шестнадцатеричные числа ... 4—24
Карта памяти ... 4—25
Восьмиричная система исчисления. ... 4—25
Дискретные и двухбайтные ячейки памяти. ... 4—25
Ячейки V-памяти для дискретной области памяти ... 4—25
Входные точки (Тип данных X) ... 4—26
Выходные точки (Тип данных Y) ... 4—26
Управляющие реле (Тип данных C) ... 4—26
Таймеры и биты состояния таймеров (Тип данных T) ... 4—26
Текущее значение таймера (тип данных V) ... 4—27
Счетчики и биты состояния счетчика (тип данных CT) ... 4—27
Текущее значение счетчика (тип данных V) ... 4—27
Двухбайтная (пословная) память (тип данных V) ... 4—28
Стадии (тип данных S) ... 4—28
Специальные Реле (тип данных SP) ... 4—28
Системная V-память DL06 ... 4—29
Системные параметры и размещение данных по умолчанию (тип данных V) ... 4—29
Карта памяти DL06 ... 4—31
Карта битового отображения входов X/выходов Y ... 4—32
Карта отображения битов Управления/Состояния стадий ... 4—33
Карта памяти управляющих реле ... 4—35
Карта битов состояния таймеров ... 4—37
Карта битов состояния счетчика ... 4—37
Карта битов удаленного ввода/вывода ... 4—38
Размещение модулей. ... 4—42
Нумерация слотов. ... 4—42
Автоматическая конфигурация Ввода / вывода ... 4—43
Ручная конфигурация ввода/вывода ... 4—43
Расчет мощности ... 4—44
Электропитание ... 4—44
Мощность, потребляемая основным модулем ... 4—44
Мощность, потребляемая дополнительными модулями ... 4—44
Настройка портов DL06 ... 4—46
Описание разъемов ... 4—46
Выбор спецификации сети ... 4—47
Сеть RS-232 ... 4—47
Сеть RS-422 ... 4—47
Сеть RS-485 ... 4—47
Подключения к сетям MODBUS и DirectNET ... 4—48
Настройка порта 2 на MODBUS ... 4—48
Настройка порта 2 на DirectNET ... 4—49
Непроцедурный протокол (ASCII ввод/вывод и Печать) ... 4—50
Настройка порта 2 на непроцедурный протокол ... 4—50
Функционирование сети в режиме ведомого устройства ... 4—51
Поддерживаемые коды функций MODBUS. ... 4—51

Определение адресов MODBUS ... 4—51
Если программное обеспечение требует тип данных и адрес ... 4—52
Если программное обеспечение для MODBUS требует только адрес ... 4—54
ФУНКЦИОНИРОВАНИЕ В РЕЖИМЕ ВЕДУЩЕГО УСТРОЙСТВА СЕТИ ... 4—57
Шаг 1: указать номер порта ведущего устройства и номер ведомого устройства ... 4—58
Шаг 2: загрузить число байтов для передачи ... 4—58
Шаг 3: указать область памяти ведущего устройства ... 4—59
Шаг 4: указать область памяти ведомого устройства ... 4—59
Передача данных из программы релейной логики ... 4—60
Блокировки многократного чтения и записи ... 4—60
ФУНКЦИОНИРОВАНИЕ В РЕЖИМЕ ВЕДУЩЕГО УСТРОЙСТВА СЕТИ MODBUS (КОМАНДЫ MRX И MWX) ... 4—61
Поддерживаемые функциональные коды MODBUS ... 4—61
Настройка порта MODBUS ... 4—62
Чтение по сети MODBUS (MRX) ... 4—63
Запись в сети MODBUS (MWX) ... 4—65
Пример работы с командами MRX/MWX в DirectSOFT32 ... 4—67

5.СТАНДАРТНЫЕ КОМАНДЫ RLL

ВВЕДЕНИЕ ... 5—2
ИСПОЛЬЗОВАНИЕ БУЛЕВЫХ КОМАНД ... 5—5
Команда END ... 5—5
Простая логическая цепь с нормально-открытым контактом ... 5—5
Нормально закрытый контакт ... 5—5
Последовательно соединенные контакты ... 5—6
Промежуточные выходы ... 5—6
Параллельно соединенные элементы ... 5—6
Объединение последовательных цепей параллельно ... 5—7
Объединение параллельных цепей последовательно ... 5—7
Комбинации цепей ... 5—7
Булевое сравнение ... 5—7
Булевый стек ... 5—8
Немедленные булевые команды (Immediate Boolean) ... 5—9
БУЛЕВЫЕ КОМАНДЫ ... 5—10
Store (STR) ... 5—10
Store Not (STRN) ... 5—10
Store Bit-of-Word (STRB) ... 5—11
Store Not Bit-of-Word (STRNB) ... 5—11
Or (OR) ... 5—12
Or Not (ORN) ... 5—12
Or Bit-of-Word (ORB) ... 5—13
Or Not Bit-of-Word (ORNB) ... 5—13
And (AND) ... 5—14
And Not (ANDN) ... 5—14
And Bit-of-Word (ANDB) ... 5—15
And Not Bit-of-Word (ANDNB) ... 5—15
And Store (AND STR) ... 5—16
Or Store (OR STR) ... 5—16
Out (OUT) ... 5—17
Or Out (OR OUT) ... 5—17
Out Bit-of-Word (OUTB) ... 5—18
Not (NOT) ... 5—19
Positive Differential (PD) ... 5—19
Store Positive Differential (STRPD) ... 5—20
Store Negative Differential (STRND) ... 5—20
Or Positive Differential (ORPD) ... 5—21
Or Negative Differential (ORRND) ... 5—21
And Positive Differential (ANDPD) ... 5—22
And Negative Differential (ANDND) ... 5—22
Set (SET) ... 5—23
Reset (RST) ... 5—23
Set Bit-of-Word (SETB) ... 5—24
Reset Bit-of-Word (RSTB) ... 5—24
Pause (PAUSE) ... 5—25
БУЛЕВЫЕ КОМАНДЫ СРАВНЕНИЯ ... 5—26
Store If Equal (STRE) ... 5—26
Store If Not Equal (STRNE) ... 5—26
Or If Equal (ORE) ... 5—27

Or If Not Egual (ORNE)..... 5—27
And If Equal (ANDE)..... 5—28
And If Not Equal (ANDNE)..... 5—28
Store (STR)..... 5—29
Store Not (STRN)..... 5—29
Or (OR)..... 5—30
Or Not (ORN)..... 5—30
And (AND)..... 5—31
And Not (ANDN)..... 5—31
НЕМЕДЛЕННЫЕ КОМАНДЫ..... 5—32
Or Immediate (ORI)..... 5—32
Or Not Immediate (ORNI)..... 5—32
And Immediate (ANDI)..... 5—33
And Not Immediate (ANDNI)..... 5—33
Out Immediate (OUTI)..... 5—34
Or Out Immediate (OROUTI)..... 5—34
Out Immediate Formatted (OUTIF)..... 5—35
Set Immediate (SETI)..... 5—36
Reset Immediate (RSTI)..... 5—36
Load Immediate (LDI)..... 5—37
Load Immediate Formatted (LDIF)..... 5—38
КОМАНДЫ ТАЙМЕРА, СЧЕТЧИКА И РЕГИСТРА СДВИГА..... 5—39
Использование таймеров..... 5—39
Timer (TMR) и Timer Fast (TMRF)..... 5—40
Пример таймера, использующего биты состояния..... 5—41
Пример таймера, использующего контакты сравнения..... 5—41
Accumulating Timer (TMRA)..... 5—42
Accumulating Fast Time(TMRAF)..... 5—42
Пример аккумулирующего таймера, с использованием бита состояния..... 5—43
Пример аккумулирующего таймера, с использованием контактов сравнения..... 5—43
Использование счетчиков..... 5—44
Counter (CNT)..... 5—45
Пример счетчика, с использованием бит состояния..... 5—46
Пример счетчика, с использованием контактов сравнения..... 5—46
Stage Counter (SGCNT)..... 5—47
Пример Stage Counter, с использованием бит состояния..... 5—48
Пример Stage Counter, с использованием контактов сравнения..... 5—48
Up Down Counter (UDC)..... 5—49
Пример Up/Down счетчика, с использованием бита состояния..... 5—50
Пример Up/Down счетчика, с использованием контактов сравнения..... 5—50
Shift Register (SR)..... 5—51
КОМАНДЫ ЗАГРУЗКИ И ВЫВОДА ДАННЫХ ДЛЯ АККУМУЛЯТОРА/СТЕКА..... 5—52
Использование аккумулятора..... 5—52
Копирование данных в аккумулятор..... 5—52
Изменение данных аккумулятора..... 5—53
Использование стека аккумулятора..... 5—54
Использование указателей..... 5—55
Load (LD)..... 5—57
Load Double (LDD)..... 5—58
Load Formatted (LDF)..... 5—59
Load Address (LDA)..... 5—60
Load Accumulator Indexed (LDX)..... 5—61
Load Accumulator Indexed from Data Constants (LDSX)..... 5—62
Load Real Number (LDR)..... 5—63
Out (OUT)..... 5—64
Out Double (OUTD)..... 5—64
Out Formatted (OUTF)..... 5—65
Pop (POP)..... 5—65
Продолжение описание команды Pop..... 5—66
Out Indexed (OUTX)..... 5—67
Out Least (OUTL)..... 5—68
ЛОГИЧЕСКИЕ КОМАНДЫ АККУМУЛЯТОРА..... 5—69
And (AND)..... 5—69
And Double (ANDD)..... 5—70
And Formatted (ANDF)..... 5—71
And with Stack (ANDS)..... 5—72
Or (OR)..... 5—73

Or Double (ORD) ... 5—74
Or Formatted (ORF) ... 5—75
Or with Stack (ORS) ... 5—76
Exclusive Or (XOR) ... 5—77
Exclusive Or Double (XORD) ... 5—78
Exclusive Or Formatted (XORF) ... 5—79
Exclusive Or with Stack (XORS) ... 5—80
Compare (CMP) ... 5—81
Compare Double (CMPD) ... 5—82
Compare with Stack (CMPS) ... 5—84
Compare Real Number (CMPR) ... 5—85
МАТЕМАТИЧЕСКИЕ КОМАНДЫ ... 5—86
Add (ADD) ... 5—86
Add Double (ADDD) ... 5—87
Add Real (ADDR) ... 5—88
Subtract (SUB) ... 5—89
Subtract Double (SUBD) ... 5—90
Subtract Real (SUBR) ... 5—91
Multiply (MUL) ... 5—92
Multiply Double (MULD) ... 5—93
Multiply Real (MULR) ... 5—94
Divide (DIV) ... 5—95
Divide Double (DIVD) ... 5—96
Divide Real (DIVR) ... 5—97
Increment (INC) ... 5—98
Decrement (DEC) ... 5—98
Add Binary (ADDB) ... 5—99
Add Binary Double (ADDBD) ... 5—100
Subtract Binary (SUBB) ... 5—101
Subtract Binary Double (SUBBD) ... 5—102
Multiply Binary (MULB) ... 5—103
Divide Binary (DIVB) ... 5—104
Increment Binary (INCB) ... 5—105
Decrement Binary (DECB) ... 5—105
Add Formatted (ADDF) ... 5—106
Subtract Formatted (SUBF) ... 5—107
Multiply Formatted (MULF) ... 5—108
Divide Formatted (DIVF) ... 5—109
Add Top of Stack (ADDS) ... 5—110
Subtract Top of Stack (SUBS) ... 5—111
Multiply Top of Stack (MULS) ... 5—112
Divide by Top of Stack (DIVS) ... 5—113
Add Binary Top of Stack (ADDBS) ... 5—114
Subtract Binary Top of Stack (SUBBS) ... 5—115
Multiply Binary Top of Stack (MULBS) ... 5—116
Divide Binary by Top OF Stack (DIVBS) ... 5—117
ТРАНСЦЕНДЕНТНЫЕ ФУНКЦИИ ... 5—118
Sine Real (SINR) ... 5—118
Cosine Real (COSR) ... 5—118
Tangent Real (TANR) ... 5—118
Arc Sine Real (ASINR) ... 5—118
Arc Cosine Real (ACOSR) ... 5—119
Arc Tangent Real (ATANR) ... 5—119
Square Root Real (SQRTR) ... 5—119
КОМАНДЫ РАБОТЫ С БИТАМИ ... 5—120
Sum (SUM) ... 5—120
Shift Left (SHFL) ... 5—121
Shift Right (SHFR) ... 5—122
Rotate Left (ROTL) ... 5—123
Rotate Right (ROTR) ... 5—124
Encode (ENCO) ... 5—125
Decode (DECO) ... 5—126
КОМАНДЫ ПРЕОБРАЗОВАНИЯ ЧИСЕЛ В АККУМУЛЯТОРЕ ... 5—127
Binary (BIN) ... 5—127
Binary Coded Decimal (BCD) ... 5—128
Invert (INV) ... 5—129
Ten’s Complement (BCDCPL) ... 5—130

Binary to Real Conversion (BTOR) ... 5—131
Real to Binary Conversion (RTOB) ... 5—132
Radian Real Conversion (RADR) ... 5—133
Degree Real Conversion (DEGR) ... 5—133
ASCII-to-HEX (ATH) ... 5—134
HEX-to-ASCII (HTA) ... 5—135
Segment (SEG) ... 5—137
Gray Code (GRAY) ... 5—138
Shuffle Digits (SFLDGT) ... 5—139
Блок-схема Shuffle Digits ... 5—139
ТАБЛИЧНЫЕ КОМАНДЫ ... 5—141
Move (MOV) ... 5—141
Move Memory Cartridge(MOVMC) /Load Label (LDLBL) ... 5—142
Копирование данных из области меток данных в V-память ... 5—143
SETBIT ... 5—144
RSTBIT ... 5—144
Fill (FILL) ... 5—146
Find (FIND) ... 5—147
Find Greater Than (FDGT) ... 5—148
Table to Destination (TTD) ... 5—150
Remove from Bottom (RFB) ... 5—153
Source to Table (STT) ... 5—156
Remove from Table (RFT) ... 5—159
Add to Top (ATT) ... 5—162
Table Shift Left (TSHFL) ... 5—165
Table Shift Right (TSHFR) ... 5—165
AND Move (ANDMOV) ... 5—167
OR Move (ORMOV) ... 5—167
Exclusive OR Move (XORMOV) ... 5—167
Find Block (FINDB) ... 5—169
Swap (SWAP) ... 5—170
КОМАНДЫ ЧАСЫ / КАЛЕНДАРЬ ... 5—171
Date (DATE) ... 5—171
Time (TIME) ... 5—172
КОМАНДЫ УПРАВЛЕНИЯ ПРОЦЕССОРОМ ... 5—173
No Operation (NOP) ... 5—173
End (END) ... 5—173
Stop (STOP) ... 5—173
Reset Watch Dog Timer (RSTWT) ... 5—174
КОМАНДЫ УПРАВЛЕНИЯ ПРОГРАММОЙ ... 5—175
Goto Label (GOTO) (LBL) ... 5—175
For(FOR) / Next (NEXT) ... 5—176
Goto Subroutine (GTS) (SBR) ... 5—178
Subroutine Return (RT) ... 5—178
Subroutine Return Conditional (RTC) ... 5—178
Master Line Set (MLS) ... 5—181
Master Line Reset (MLR) ... 5—181
Объяснение работы Master Control Relays ... 5—181
Пример MLS/MLR ... 5—182
КОМАНДЫ ПРЕРЫВАНИЯ ... 5—183
Interrupt (INT) ... 5—183
Interrupt Return (IRT) ... 5—183
Interrupt Return Conditional (IRTC) ... 5—183
Enable Interrupts (ENI) ... 5—183
Disable Interrupts (DISI) ... 5—184
Пример программы внешнего прерывания ... 5—184
Пример программы прерывания по времени ... 5—185
КОМАНДЫ ВЫВОДА СООБЩЕНИЙ ... 5—186
Fault (FAULT) ... 5—186
Пример команды Fault ... 5—186
Data Label (DLBL) ... 5—187
ASCII Constant (ACON) ... 5—187
Numerical Constant (NCON) ... 5—187
Пример команды Data Label ... 5—188
Print Message (PRINT) ... 5—189
Write to Network (WX) ... 5—195
LCD ... 5—197

Непосредственный ввод текста ..... 5—197
Встраивание значений даты и/или времени ..... 5—198
Встраивание значений V-памяти ..... 5—198
Суффиксы форматов данных для встраивания значений из V- памяти. ..... 5—199
Ввод текста из V-памяти ..... 5—200
Команды MODBUS RTU ..... **5—201**
MODBUS Read from Network (MRX) ..... 5—201
Пример MRX ..... 5—203
MODBUS Write to Network (MWX) ..... 5—204
Пример MWX ..... 5—206
Команды ASCII ..... **5—207**
Чтение входной ASCII-строки. ..... 5—207
Запись выходной ASCII-строки ..... 5—207
Управление ASCII-строками ..... 5—208
ASCII Input (AIN) ..... 5—209
ASCII Find (AFIND) ..... 5—213
Пример поиска AFIND ..... 5—214
Пример объединения AFIND с командой AEX ..... 5—215
ASCII Extract (AEX) ..... 5—216
ASCII Compare (CMPV) ..... 5—217
ASCII Print to V–memory (VPRINT) ..... 5—218
ASCII Print from V–memory (PRINTV) ..... 5—223
ASCII Swap Bytes (SWAPB) ..... 5—224
ASCII Clear Buffer (ACRB) ..... 5—225

Глава 1

1. Начальные сведения.

В этой главе. . . .

Введение. ..... 1—2
Используемые соглашения ..... 1—3
Обзор микроконтроллера DL06 ..... 1—4
Методы программирования ..... 1—4
Схема быстрого выбора подсистемы ввода/вывода. ..... 1—5
Быстрый старт ..... 1—6
Шаги по успешному созданию системы автоматики ..... 1—10
Вопросы и ответы по микроконтроллерам DL06 ..... 1—12

Введение.

Цели данного руководства

Благодарим Вас за покупку микроконтроллера DL06. В этом руководстве представлены сведения о подключении, программировании и техническом обслуживании всех микроконтроллеров серии DL06, а также о взаимодействии микроконтроллеров с другими устройствами в единой системе управления. Данное руководство содержит важные сведения для программистов и лиц, подключающих контроллеры DL06. Это руководство содержит информацию, необходимую для мополучения микроконтроллера и его хранения.

Дополнительные руководства

Руководство D0-OPTIONS-M содержит техническую информацию о дополнительных модулях для контроллера DL06. В это руководство включены спецификации и схемы соединений, которые будут важны при использовании любого из дополнительных или коммуникационных модулей. Если Вы купили одну из операторских панелей или программное обеспечение DirectSOFT, то Вам необходимы руководства с описанием этих продуктов.

Техническая поддержка

Если Вы не можете найти решение Вашей конкретной задачи, или по любой другой причине Вы нуждаетесь в технической поддержке, пожалуйста обращайтесь к нам.

- Адрес нашего WEB-сайта: www.plcsystems.ru
- Электронная почта: support@plcsystems.ru
- Телефон в Москве: (495) 105-77-98

Наши специалисты будут рады ответить на все Ваши вопросы с понедельника по пятницу с 9.00 до 18.00 по московскому времени.

Если у Вас возникнут комментарии, вопросы или предложения по любому из наших продуктов, услугам или документации, то, пожалуйста, заполните и верните карточку «Предложения», которая включена в данное руководство.

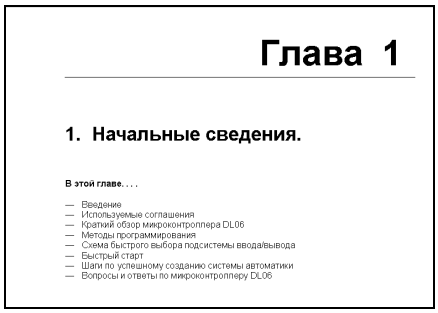
Используемые соглашения

Когда Вы видите этот значок «блокнот» в левой части страницы, то в абзаце расположенном справа, приводится специальное примечание. В примечании предоставляется информация, которая сделает Вашу работу более быстрой или эффективней.Слово **ПРИМЕЧАНИЕ**, выделенное полужирным шрифтом, отмечает начало текста.

Когда Вы видите этот значок «восклицательный знак» в левой части страницы, то в абзаце расположенном справа, приводится предупреждение. Данная информация поможет Вам предотвратить повреждения, потерю функциональности или даже гибель в экстремальных случаях. Любое предупреждение в этом руководстве должно быть расценено как важная информация, которая дожна быть прочитана полностью.
Слово **ПРЕДУПРЕЖДЕНИЕ**, выделенное полужирным шрифтом, отмечает начало текста.

Ключевые темы каждой главы

В начале каждой главы приводится список ключевых тем, которые можно найти в данной главе.



Глава 1

1. Начальные сведения.

В этой главе. . . .

— Введение
— Используемые соглашения
— Краткий обзор микроконтроллера DL06
— Методы программирования
— Схема быстрого выбора подсистемы ввода/вывода
— Быстрый старт
— Шаги по успешному созданию системы автоматики
— Вопросы и ответы по микроконтроллеру DL06

Обзор микроконтроллера DL06

Микроконтроллеры серии DL06 комбинируют в себе мощные возможности и компактные размеры. Серия DL06 предлагает расширяемый ввод/вывод, высокоскоростной счетчик, арифметику с плавающей точкой, ПИД-регулирование, ряд коммуникационных возможностей, LCD панель и многое другое.

Возможности микроконтроллеров DL06

Серия DL06 включает в себя восемь различных моделей. Все они имеют одинаковый внешний вид и одинаковые характеристики процессора. Процессор предлагает набор команд очень похожий на команды нового мощного процессора DL260, включая команды ASCII и MODBUS. Все микроконтроллеры DL06 имеют два коммуникационных порта, которые могут использоваться для программирования, подключения операторских панелей и работы с сетью.
Модели с входами постоянного тока предусматривают возможность переключения четырех входных каналов в режим высокоскоростного ввода. Модели с выходами постоянного тока предусматривают возможность переключения первого и второго канала вывода в режим импульсного выхода. Более подробная информация по этим и другим функциям приводится в главе 4 «Спецификации и работа процессора». Восемь моделей данной серии обеспечивают различные варианты ввода/вывода и питания , перечисленны в следующей таблице.

№ каталога	Тип дискретного входа	Тип дискретного выхода	Внешнее питание	Высоко-скоростной ввод	Импуль-сный выход
D0-06AA	Переменный ток	Переменный ток	~95-240В	Нет	Нет
D0-06AR	Переменный ток	Релейный	~95-240В	Нет	Нет
D0-06DA	Постоянный ток	Переменный ток	~95-240В	Да	Нет
D0-06DD1	Постоянный ток	Постоянный ток- потребитель	~95-240В	Да	Да
D0-06DD2	Постоянный ток	Постоянный ток-источник	~95-240В	Да	Да
D0-06DR	Постоянный ток	Релейный	~95-240В	Да	Нет
D0-06DD1-D	Постоянный ток	Постоянный ток- потребитель	-12-24В	Да	Да
D0-06DD2-D	Постоянный ток	Постоянный ток- источник	-12-24В	Да	Да
D0-06DR-D	Постоянный ток	Релейный	-12-24В	Да	Нет

Методы программирования

Для программирования микроконтроллеров доступны два языка: RLL (Relay Ladder Logic – язык релейной логики) и RLL^{PLUS}. RLL^{PLUS} объединяет стандартный язык релейной логики с возможностями стадийного программирования (Stage^{TM}). Оба языка одинаково хорошо поддерживаются как ручным программатором, так и пакетом программирования DirectSoft^{TM}.

Пакет программирования DirectSOFT32 под Windows

Микроконтроллеры DL06 можно программировать с помощью графического пакета DirectSOFT (версия 4 или выше), который поддерживает многие хорошо известные функции Windows, такие как «вырезать-вставить» между приложениями, редактирование по щелчку мыши, просмотр и редактирование нескольких прикладных программ одновременно, и т. д.

DirectSOFT32 (PC-PGMSW) поддерживает все серии контроллеров **Direct**LOGIC. Таким образом, Вы можете использовать один и тот же пакет для программирования DL05, DL06, DL105, DL205 и DL405. (При переходе на новый процессор может понадобиться обновление версии программного обеспечения). Пакет программирования **Direct**SOFT32 описывается в отдельном руководстве. Для программирования DL06 Вам потребуется версия 4.0 или выше.

Ручной программатор

Все микроконтроллеры DL06 имеют встроенный порт для программирования с использованием ручного программатора (D2-HPP), такой же программатор используется с сериями DL05, DL105 и DL205. Ручной программатор может быть использован для создания, изменения или отладки Вашей прикладной программы. Ручной программатор рассматривается в отдельном руководстве. Для программирования DL06 Вам понадобится D2-HPP со встроенным программным обеспечением версии 2.0 или выше.

Схема быстрого выбора подсистемы ввода/вывода.

Девять моделей DL06 имеют различные цепи ввода/вывода, которые могут взаимодействовать с разнообразными внешними устройствами. В некоторых случаях отдельные цепи ввода или вывода могут работать с постоянным или переменным током, как источник или потребитель тока. Внимательно просмотрите руководство ,чтобы подобрать контроллер DL06, соответствующий полевым устройствам Вашего приложения.

№ каталога	Входы			Выходы		
	Тип входа/ общие провода	**Потребитель /Источник**	**Диапазон напряже-ний**	**Тип выхода / число общих проводов**	**Потребитель / Источник**	**Номиналы напряжение/ ток***
D0-06AA	Переменный ток / 5	-	~90–120В	Переменный ток / 4	-	~17-240В, 50/60Гц, 0.5А
D0-06AR	Переменный ток / 5	-	~90–120В	Релейный/4	Потребитель или источник	6-27В, 2.0А ~6-240В, 2.0А
D0-06DA	Постоянный ток / 5	Потребитель или источник	=12 – 24В	Переменный ток / 4	-	~17-240В, 50/60Гц, 0.5А
D0-06DD1	Постоянный ток / 5	Потребитель или источник	=12 – 24В	Постоянный ток / 4	Потребитель	6-27В, 0.5А (Y0-Y1) 6-27В, 1.0А (Y2-Y17)
D0-06DD2	Постоянный ток / 5	Потребитель или источник	=12 – 24В	Постоянный ток / 4	Источник	12-24В, 0.5А (Y0-Y1) 12-24В, 1.0А(Y2-Y17)
D0-06DR	Постоянный ток / 5	Потребитель или источник	=12 – 24В	Релейный/4	Потребитель или источник	6-27В, 2.0А ~6-240В, 2.0А
D0-06DD1-D	Постоянный ток / 5	Потребитель или источник	=12 – 24В	Постоянный ток / 4	Потребитель	6-27В, 0.5А (Y0-Y1) 6-27В, 1.0А (Y2-Y17)
D0-06DD2-D	Постоянный ток / 5	Потребитель или источник	=12 – 24В	Постоянный ток / 4	Источник	12-24В, 0.5А (Y0-Y1) 12-24В, 1.0А(Y2-Y17)
D0-06DR-D	Постоянный ток / 5	Потребитель или источник	=12 – 24В	Релейный/4	Потребитель или источник	6-27В, 2.0А ~6-240В, 2.0А

* См. главу 2 «Спецификации» для Вашего конкретного варианта.

Быстрый старт

Этот пример не предназначен для объяснения всего, что Вы должны знать о программировании и запуске сложной системы управления. Он предназначен, только для демонстрации основных шагов, необходимых для включения контроллера и проверки его работоспособности. Если Вы не хотите пропустить важную информацию, пожалуйста, читайте предупреждения и примечания приведенные в руководстве.

Шаг 1: Распаковка оборудования DL06

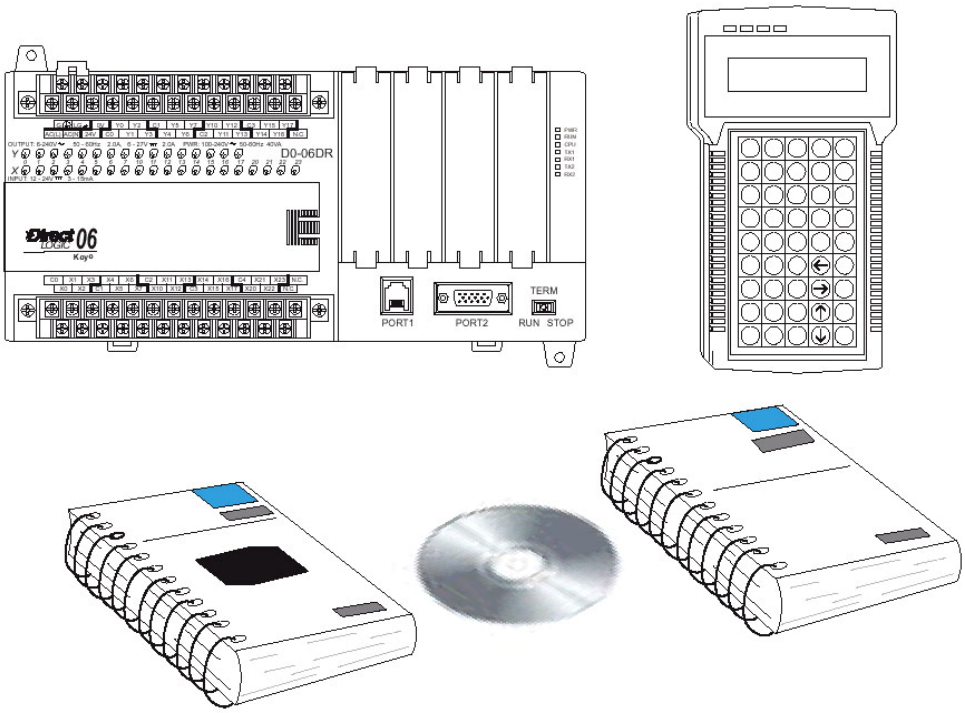
Распакуйте оборудование DL06 и соберите детали, необходимые для построения демонстрационной системы. Рекомендуемыми компонентами являются:

- Микроконтроллер DL06
- Шнур питания
- Тумблеры (см. Шаг 2 ниже)
- Провода для подключения, 0,33 - 1,3 мм^2
- Руководство пользователя DL06 (данное руководство)
- Небольшая крестовая отвертка с гранью 1,6мм или №1

Вам необходимо иметь, по крайней мере, одно из следующих средств для программирования:

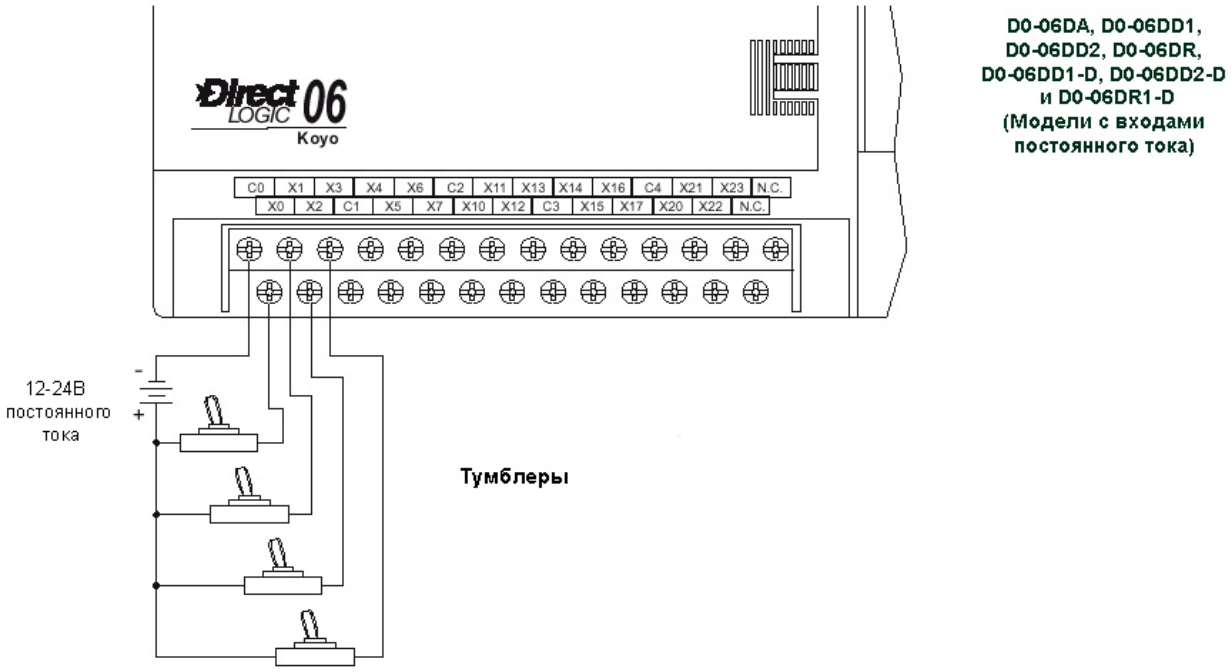
- Программное обеспечение DirectSOFT32, версия 4 или выше (PC-PGMSW или PC-PGM-BRICK), Руководство по DirectSOFT32 (поставляемое вместе с программным обеспечением) и кабель для программирования (D2-DSCBL, для подключения DL06 к персональному компьютеру),

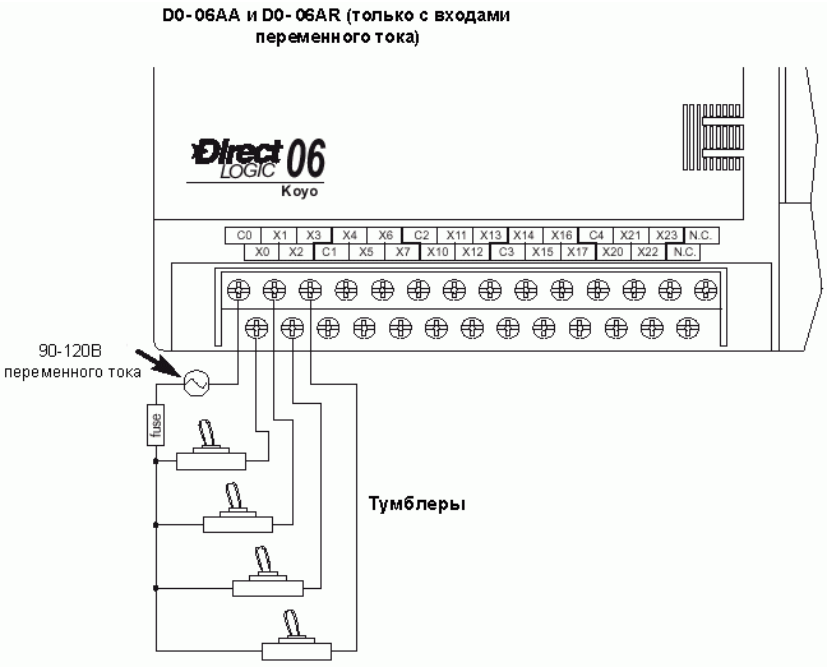
  или
- Ручной программатор D2-HPP, версия 2.0 или выше (поставляется вместе с кабелем для подключения). Руководство по Ручному программатору (D2-HPP-M) поставляется отдельно.



Шаг 2: подключение тумблеров к входным клеммам

Для продолжения примера быстрого подключения, а также для других примеров в данном руководстве Вам необходимо подсоединить один или больше входных тумблеров, как показано ниже. Если у Вас входы постоянного тока, а питание – переменный ток, то Вы можете использовать вспомогательный источник питания 24В на выходных клеммах или другой внешний источник питания на 12-24 В постоянного тока. Обеспечьте выполнение указаний в ПРЕДУПРЕЖДЕНИЯХ, сопровождающих текст.

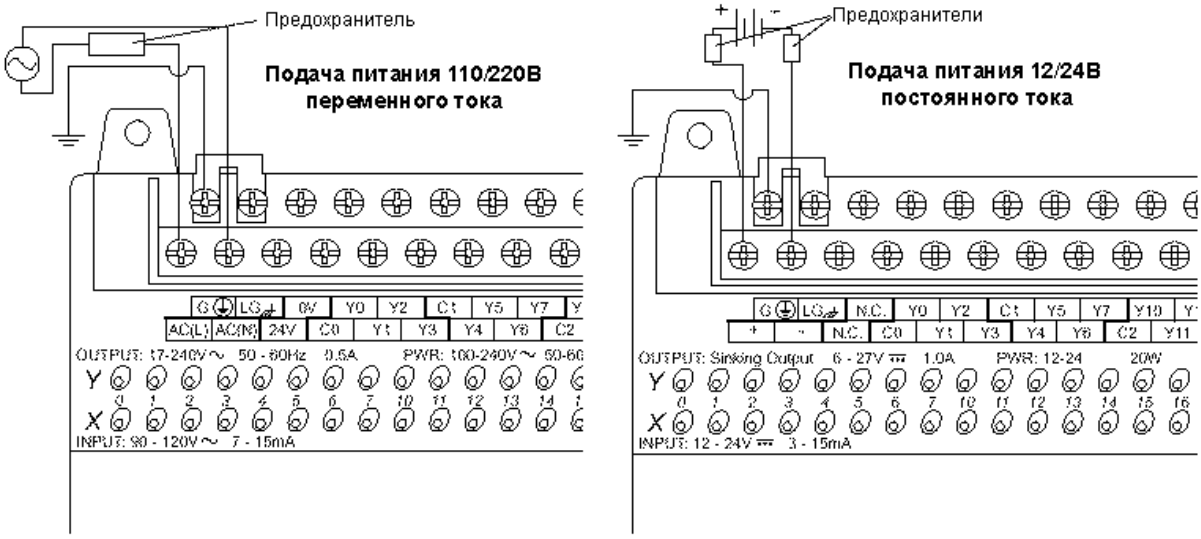




ПРЕДУПРЕЖДЕНИЕ:
При монтаже выключателей отключите питание и отсоедините DL06. Для входов переменного тока используйте только выключатели с номиналом, по крайней мере, 250 В, 1 А. Надежно закрепите выключатели перед их использованием.

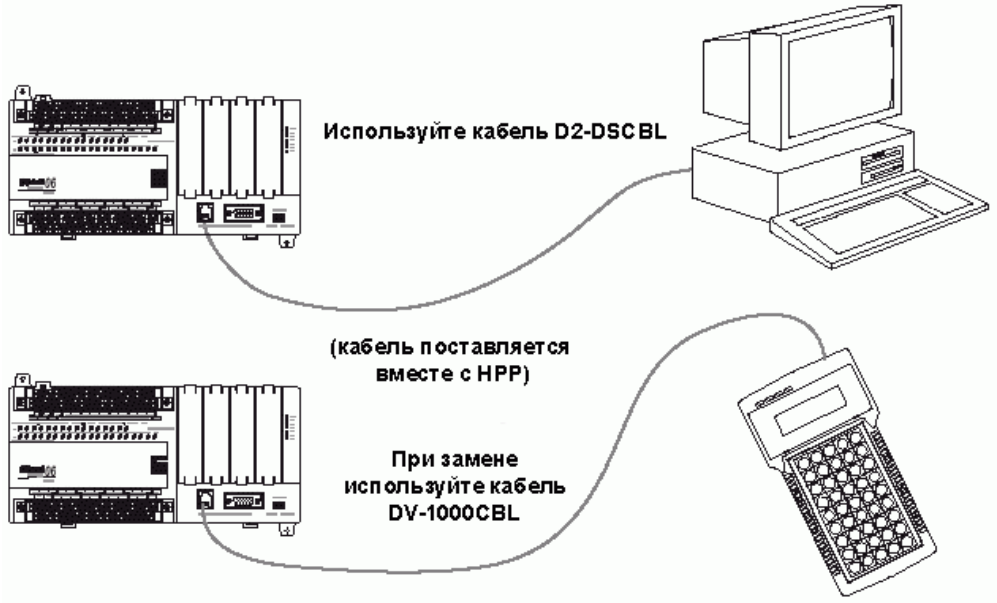
Шаг 3: Подключение питания.

Подсоедините электропитание 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 разъемов для подключения питания. Не подключайте питание до окончания монтажа.



Шаг 4: Подключение программирующих устройств.

Большинство программистов используют программное обеспечение DirectSOFT32, версии 4 или выше, установленное на персональном компьютере. В качестве альтернативы, если Вы нуждаетесь в компактном переносном устройстве программирования, можно использовать Ручной программатор (версия программного обеспечения 2.20 или выше). Оба устройства подсоединяются к COM порту 1 DL06 через соответствующий кабель.



Примечание: Ручной программатор не дает доступ к командам управления LCD дисплеем, ASCII и MODBUS

Шаг 5: Включение питания системы

Подайте питание в систему и проверьте, включается ли индикатор PWR при включении DL06. Если нет, то отключите питание системы и проверьте все соединения, обратитесь для помощи к разделу «Поиск неисправностей» в главе 9.

Шаг 6: Инициализация оперативной памяти.

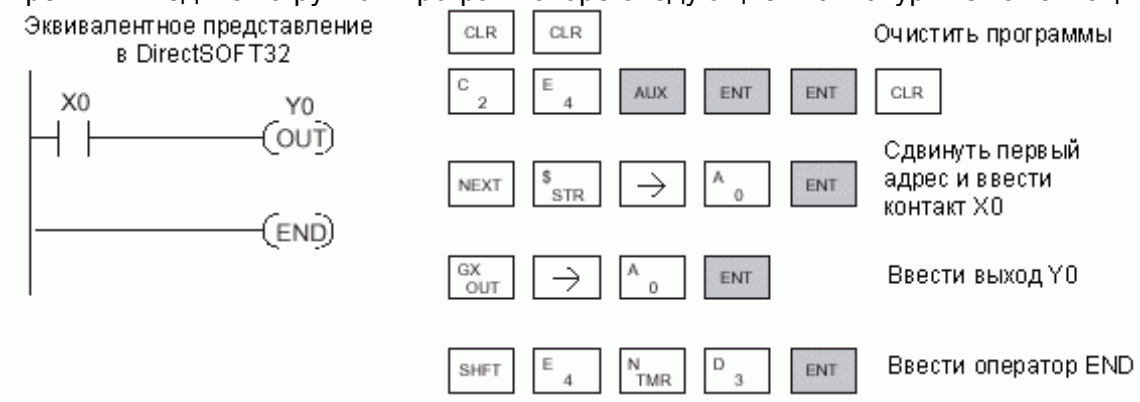
Хорошей мерой предосторожности является очистка системной оперативной памяти (Scratchpad) в новом микроконтроллере DL06. Существует два способа очистки системной памяти:

- В DirectSOFT32 выберите пункт меню PLC, затем Setup, далее Initialize Scratchpad (Инициализация оперативной системной памяти). Для получения дополнительной информации см. руководство по DirectSOFT32. Инициализация системной оперативной памяти установит параметры второго порта связи в их значения по умолчанию. Если Вы делали любые изменения этих параметров, то Вы должны будете обратить внимание на эти изменения и вернутся к ним после инициализации
- На ручном программаторе, используя клавишу AUX, запустите на выполнение команду AUX 54. Для получения дополнительной информации обратитесь к Руководству по ручному программатору

Шаг 7: Ввод программы.

В этом месте программистам в DirectSOFT следует обратиться к главе «Быстрый Запуск» руководства по DirectSOFT. Необходимо изучить, как устанавить связь с микроконтроллером DL06, как изменять режимы процессора на Run или Program, как вводить программу.

Если вы решили программировать с помощью ручного программатора, убедитесь, что процессор находится в Программном режиме (светодиод RUN на передней панели DL06 должен быть выключен). Если светодиод RUN включен, используйте клавишу MODE на ручном программаторе, чтобы перевести ПЛК в Программный режим. Введите на ручном программаторе следующие клавиатурные комбинации:



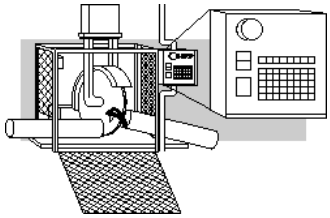
После ввода программы этого простого примера переведите ПЛК в рабочий режим, используя клавишу Mode на ручном программаторе.

Индикатор RUN на ПЛК начнет светиться, указывая, что процессор переведен в Рабочий режим. Если он не включится, то следует повторить этот шаг, обеспечивая надлежащий ввод программы, или обратиться к разделу «Поиск неисправностей» в главе 9.

После перевода процессора в рабочий режим индикатор состояния выхода для Y0 должен отражать положение переключателя на входном канале X0. Когда переключатель находится в положении «включен», выход также должен быть включен.

Шаги по успешному созданию системы автоматики

Шаг 1: Просмотреть Руководство по установке

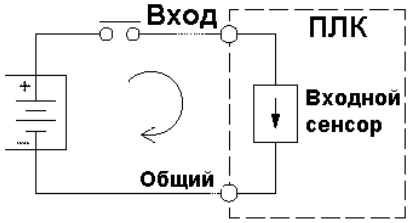


Всегда ставьте безопасность в качестве первого приоритета при любой системной разработке. В главе 2 дается несколько указаний, которые помогут создать более надежную и безопасную систему. Эта глава также включает указания по монтажу для различных вариантов микроконтроллера DL06.

Шаг 2: Понять методы настройки ПЛК (Setup Procedure)

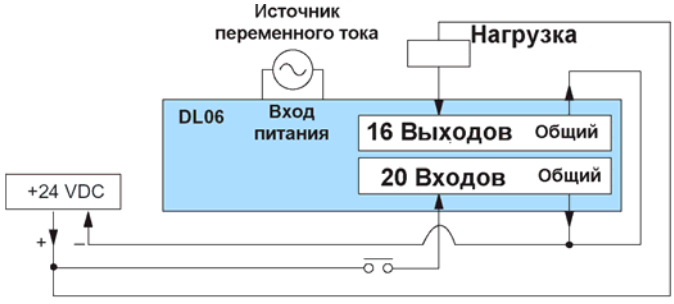
ПЛК является сердцем вашей автоматической системы. Выделите время для того, чтобы разобраться с разнообразными функциями DL06 и возможностями его настройки.

Шаг 3: Оценить критерии выбора подсистемы ввода / вывода



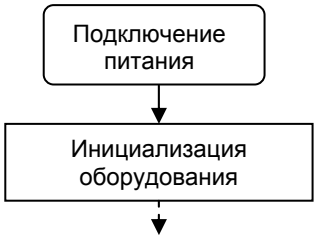
При выборе типа сигналов ввода/вывода, а также внешних устройств необходимо принимать во внимание многие соображения. Выделите время, чтобы разобраться с тем, как различные типы датчиков и нагрузок влияют на выбор типа сигналов ввода/вывода.

Шаг 4: Выбрать стратегию монтажа системы



Перед подключением к микроконтроллеру внешних устройств и источников питания внешних устройств важно понять возможные варианты конструирования системы.

Шаг 5: Понять работу системы



Прежде, чем Вы начнете ввод программы, необходимо понять, как DL06 обрабатывает информацию. Это относится не только к выполнению программы, но и к различным режимам работы и характеристикам расположения памяти.

Шаг 6: Просмотреть концепции программирования

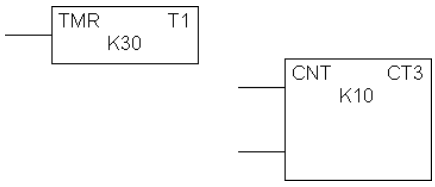
Набор команд DL06 обеспечивает три главных подхода к программированию прикладных задач, показанные на рисунке ниже.

- RLL – язык релейной логики —является наилучшим инструментом для решения задач булевой логики и при манипуляции с регистром/аккумулятором процессора. Этот язык включает множество команд, которые могут быть дополнены барабанными командоаппаратами и стадиями.
- Барабанный командоаппарат с управлением по таймеру/событиям включает до 16 шагов, предусматривает пошаговые переходы по времени и/или по событиям. Команды барабанного командоаппарата являются наилучшим инструментом для повторяющегося процесса с одной последовательностью шагов.
- RLL^{PLUS} – стадийное программирование – основано на диаграммах перехода состояний. Стадии разделяют программу на секции, соответствующие состояниям в блок-схеме вашего процесса.

После анализа приведенных выше концепций программирования вы будете иметь набор средств для написания вашей прикладной программы.

Шаг 7: Выбрать команды

Как только Вы установили ПЛК и выяснили основные концепции программирования, то Вы можете начать писать вашу прикладную программу. При этом Вы начнете использовать один из наиболее мощных наборов инструкций, доступных в маленьком ПЛК.



Шаг 8: Понять методы обслуживания и поиска неисправности

Иногда оборудование отказывает, когда мы меньше всего этого ожидаем. Переключатели могут выйти из строя, в нагрузке может возникнуть короткое замыкание и потребоваться замена какого-то оборудования и т. д. В большинстве случаев при поиске неисправности и при техническом обслуживании основное время тратится на локализацию проблемы. Микроконтроллер DL06 включает много встроенных функций, например, коды ошибок, которые помогут вам быстро идентифицировать неисправности.

Вопросы и ответы по микроконтроллерам DL06

В. Какие есть аналоги набору команд DL06?

О. Набор команд очень близок к набору команд процессора DL260. Команды DL06 включают команды барабанного командоаппарата, работу в сети, ASCII, Modbus, управление LCD панелью и возможности высокоскоростного ввода/вывода.

В. Должен ли я покупать полный программный пакет DirectSOFT32 для программирования DL06?

О. Нет. Мы предлагаем версию DirectSOFT32 - PC-PGM-BRICK только для микроконтроллеров , DL105 и DL06, по более низкой цене.

В. Является ли DL06 расширяемым ПЛК?

О. Да. DL06 моноблочный ПЛК, однако в контроллере есть 4 дополнительных слота, позволяющие Вам расширять систему без изменения занимаемой площади.

В. Можно ли с помощью DL06 управлять движением?

О. Да. DL06 имеет ограниченные возможности по управлению движением. Высокоскоростной ввод/вывод включает как входы с кодирующего устройства с высокоскоростными вычислениями и прерываниями, так и импульсный /направленный выход для управления шаговым электродвигателем. Доступны три профиля управления движением, они рассматриваются в главе 3

В. Хранятся ли пользовательские программы в съемном ЭППЗУ (Электронно-перепрограммируемом ПЗУ)?

О. Нет. Для хранения программ DL06 имеет стационарную ФЛЭШ-память, в которую можно записывать программы и стирать их тысячи раз. Вы можете пересылать программы в/из персональный компьютер. При помощи DirectSOFT

В. Имеет ли DL06 предохранители в выходных цепях?

О. Предохранители в выходных цепях отсутствуют. Поэтому мы рекомендуем устанавливать предохранители в каждом канале и в каждом общем проводе. См. главу 2 с указаниями по монтажу подсистемы Ввода/Вывода.

В. Соответствует ли микроконтроллер DL06 Директивам Европейского Союза ?

О. Микроконтроллер DL06 соответствует требованиям Директив Европейского Союза.

В. Какие устройства можно подсоединить к коммуникационным портам DL06 ?

О. **Порт 1.** Порт 1 имеет тип RS-232C с фиксированной скоростью передачи 9600 бод, имеет проверку на нечетность, адрес 1 и использует собственный протокол K-sequence. DL06 может быть также подсоединен через порт 1 к сетям MODBUS и DirectNET как ведомое устройство. К порту можно подключить следующие устройства:

- Панели оператора DV-1000, EZTouch, EZText, DirectTouch, DSData, Optimation.
- Программный пакет DirectSOFT32 (работающий на компьютере).
- Ручной программатор D2-HPP.
- Другие устройства, взаимодействующие по протоколам: K-sequence, DirectNet, MODBUS RTU.

О. **Порт 2.** Порт 2 – многофункциональный порт. Имеет тип RS-232C, RS422 или RS485 с настраиваемыми скоростями передачи в бодах (от 300 до 38400 бит/с), адресацией и настройкой контроля четности. Он поддерживает протоколы: K-sequence, DirectNET и MODBUS RTU, (master/slave), ASCII (ввод/вывод) и протокол печати (по последовательному порту).

В. Может ли DL06 принимать входные сигналы 5 В постоянного тока?

О. Нет, 5 В ниже пороговой границы подключения входов постоянного тока. Однако многие логические схемы ТТЛ (Транзисторно-транзисторная логика) могут передавать такие входы, если они подключены как входы с открытым коллектором (как потребители тока). См. главу 2 с указаниями по монтажу подсистемы Ввода/Вывода.

Глава 2

2. Установка, электромонтаж и спецификации

В этой главе…

Инструкция по технике безопасности ..... 2—2
Передняя панель DL06 ..... 2—4
Монтаж ..... 2—6
Подключение питания ..... 2—10
Основные принципы электромонтажа системы ..... 2—13
Словарь терминов в спецификации ..... 2—25
Схемы подключения и спецификации. ..... 2—26

Инструкция по технике безопасности

ПРИМЕЧАНИЕ: Оборудование с маркировкой CE надежно и безопасно выполняет требуемые функции в соответствии с основными стандартами, определенными директивами Европейского союза, при условии, что оно используется точно по назначению и согласно приведенным в данном руководстве инструкциям. Защита, предоставляемая оборудованием, может быть ослаблена, если это оборудование будет использоваться отличным от указанного в данном руководстве способом.

ПРЕДУПРЕЖДЕНИЕ: Ваша обязанность — предоставить персоналу безопасную рабочую среду и оборудование. Это должно стать главной задачей при разработке и установке системы. Системы автоматизации иногда выходят из строя, а это может привести к серьезным травмам обслуживающего персонала и поломке оборудования. При создании безопасной рабочей среды не стоит полагаться только на систему автоматизации. Для защиты любого узла системы, поломка которого может привести к тяжелым последствиям и травмам персонала, необходимо использовать внешние электромеханические устройства (например, реле или концевые выключатели), которые не зависят от программы ПЛК. Каждое приложение по-своему отличается от других, и поэтому в вашем случае могут потребоваться дополнительные средства. Для обеспечения правильного функционирования оборудования необходимо неукоснительно следовать всем правилам и нормативным требованиям местного и федерального характера.

Планирование безопасности

Чтобы рабочее место было действительно безопасным, надо, прежде всего, позаботиться о безопасности персонала и оборудования. При определении узлов оборудования, несущих наибольшую опасность персоналу и работе системы в целом, вам придется исследовать каждый компонент. Если вы не знакомы с принципами установки систем ПЛК или ваша компания не располагает необходимой документацией по этому вопросу, обратитесь за помощью в следующие источники.

- NEMA — Национальная Ассоциация Электротехнической Промышленности, расположенная в Вашингтоне (округ Колумбия), публикует множество документов, в которых обсуждаются стандарты и параметры промышленных систем управления оборудованием. Вы можете заказать эти публикации непосредственно в NEMA. Вот некоторые из них:
  ICS 1, Общие стандарты контроля и управления в промышленности
  ICS 3, Промышленные системы
  ICS 6, Компоненты защиты и кожухи для промышленных систем управления
- NEC — Национальный код электроустановок — предоставляет инструкции и правила, касающиеся установки и эксплуатации различной электротехники.
- Местные и федеральные управления — многие местные и федеральные агентства утверждают дополнительные требования к электрооборудованию, которые могут отличаться от информации в справочнике NEC. Подробную информацию вы можете получить у местного инспектора электротехнического оборудования или инспектора пожарной безопасности.

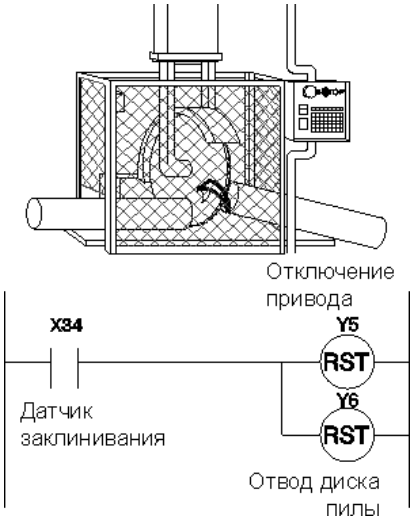
Три уровня защиты

Вышеупомянутые публикации содержат много ценных сведений по обеспечению безопасности оборудования. По крайней мере, вы должны следовать всем этим указаниям. Вдобавок, примите к сведению следующие технологии, обеспечивающие три уровня управления любой системой.

- Четкая последовательность выключения системы в программе ПЛК.
- Механическое отсоединение выходов от источника питания.
- Ключ аварийной остановки для отключения источника питания системы.

Четкая последовательность выключения

С задачами первого уровня обнаружения неисправностей прекрасно справляется программа управления микро ПЛК, которая способна идентифицировать неполадки в системе. Вам необходима последовательность операций для выключения приложения. Причиной подобных проблем обычно являются, например, заклинившие или застопоренные детали и другие узлы, которые не представляют собой угрозу жизни или поломки оборудования.



ПРЕДУПРЕЖДЕНИЕ. Программа управления не должна быть единственной формой защиты от любых проблем, которые могут привести к травмам персонала или поломке оборудования.

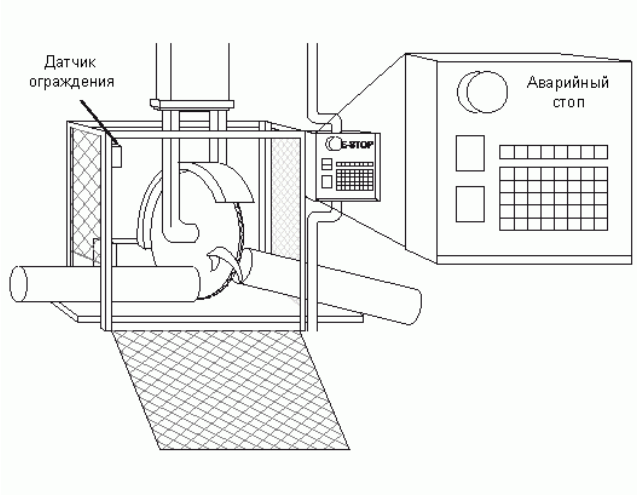
Отключение питания системы

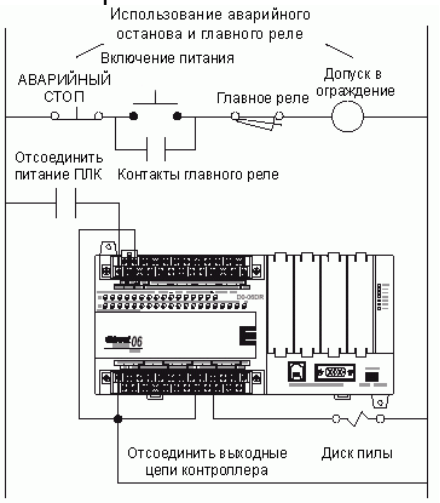
В целях предотвращения случайного (внезапного) запуска оборудования Вам также необходимо использовать электромеханические устройства, такие как, например управляющие реле и/или концевые выключатели. Нужно установить данные устройства так, чтобы они смогли предотвратить выполнение любой операции оборудования.

Например, если в машине заклинило какую-либо деталь, программа управления ПЛК сможет остановить диск пилы и отодвинуть вал. Однако, так как оператор должен (перед тем как вынуть заклинившую деталь) открыть защитный кожух, необходимо в дополнении установить концевой выключатель, который бы отсоединял питание системы при любом открытии кожуха.

Аварийное отключение

В используемом оборудовании должно быть предусмотрено быстрое отключение («вручную») всего энергопитания системы. Необходимо четко обозначить устройство отсоединения или выключатель надписями «Аварийная остановка».

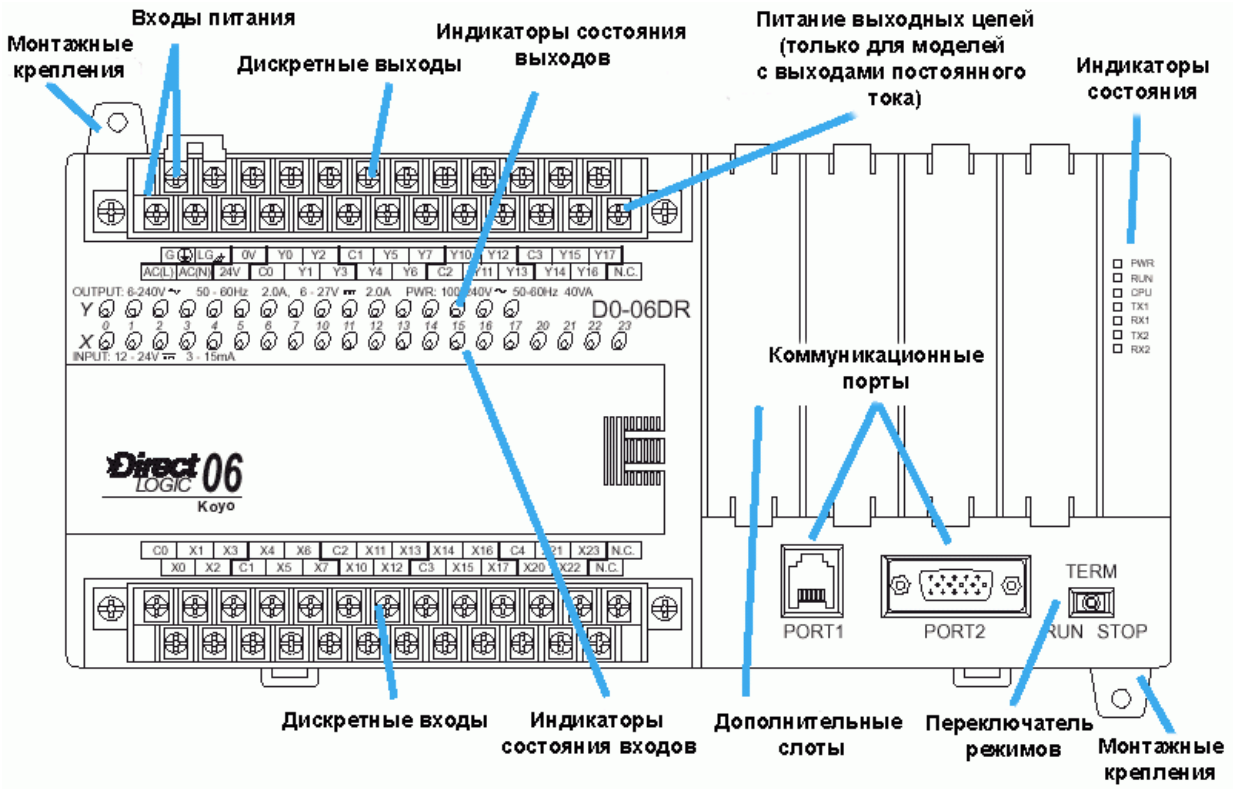




После аварийного выключения или любого выключения питания может потребоваться некоторая корректировка технических параметров, для того чтобы можно было приготовить программу управления ПЛК к последующему запуску. Например, перед тем как продолжить последовательность операций, может понадобиться установка специальных значений регистров (или восстановление исходных значений). В таком случае, вы можете использовать соответствующие ячейки памяти или включать предварительно определенные константы в программу управления.

Передняя панель DL06

Большая часть соединений, индикаторов и обозначений расположены на передней панели микро ПЛК DL06. Порты связи находятся на передней стороне контроллера, так же как слоты для дополнительных модулей и переключатель выбора режимов. См. нижеследующий рисунок.



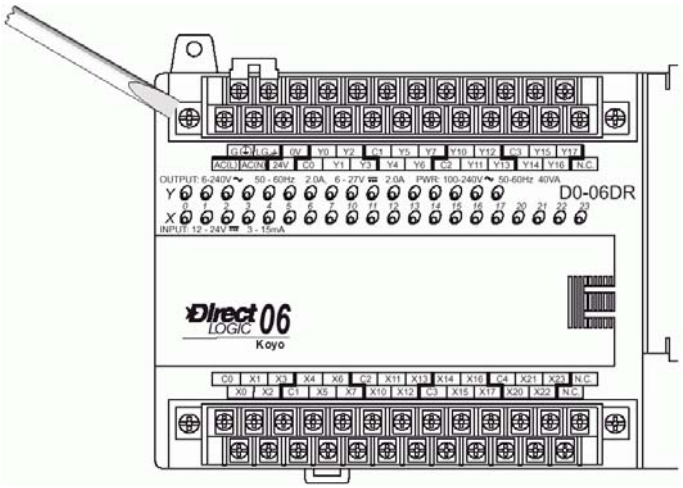
Выходные разъемы и разъемы питания принимают внешнее питание AC(L) - AC(N), логическую землю, заземление корпуса на обозначенных контактах. Оставшиеся клеммы для подключения общих проводов C0-C3 и выходов с Y0 до Y17. Клеммы 16 выходов пронумерованны в восьмеричной системе счисления, от Y0 до Y7 и от Y10 до Y17. В моделях с выходами постоянного тока крайняя правая клемма обеспечивает питание для выходных цепей.
Входные разъемы обеспечивают подключение входов X0 и X23 и общих проводов C0-C4.

ПРЕДУПРЕЖДЕНИЕ: В некоторых случаях электропитание периферийного устройства может оставаться на клеммной колодке даже после того, как будет выключен микро ПЛК. Для уменьшения риска получения электрошоковой травмы, не забывайте проверять наличие питание периферийного устройства перед тем, как будете осматривать или демонтировать клеммный блок.

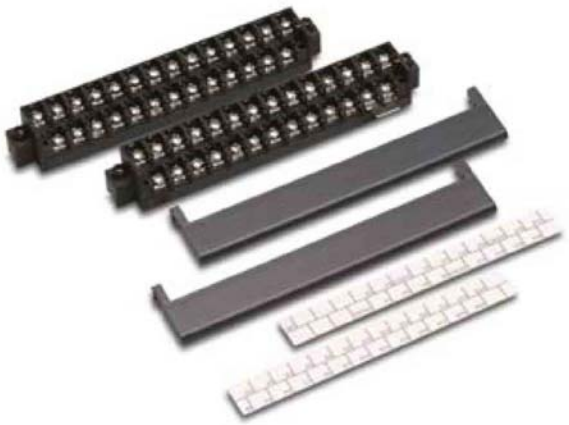
Съемный клеммник

Клеммы DL06 разделены на две группы. Каждая группа имеет свой собственный клеммный блок. На одном блоке монтаж выходов и питания, монтаж входов на другом.
Иногда, в целях облегчения электромонтажа, необходимо снять клеммный блок. Конструкция клеммного блока позволяет снять его с помощью небольшой отвертки. На следующем ниже рисунке показана данная процедура (демонтаж одного клеммного блока).

1. Отвинтите невыпадающие винты с каждого края клеммного блока.



2. Начиная от центра, отверткой приподнимите вверх весь узел до тех пор, пока клеммный блок полностью не освободится.
Клеммный блок DL06 снабжен обычными винтами (M3), для которых подойдет стандартная шлицевая или крестообразная отвертка. Используйте одножильный/многожильный провод размером от 16 AWG(1.5 мм кв) до 22 AWG (0.3 мм кв). Будьте осторожны — нет необходимости сильно закручивать винт; максимальное значение момента силы – от 0.882 до 1.020 Нм (7.806-9.028 фунт силы-дюйм).
Запасные клеммные блоки можно заказать по номеру каталога: D0-ACC-2 – набор аксессуаров для DL06, включающий в себя 2 клеммных блока, 2 крышки для клеммных блоков, 2 этикетки и коротких линейки и 1 короткую линейку.



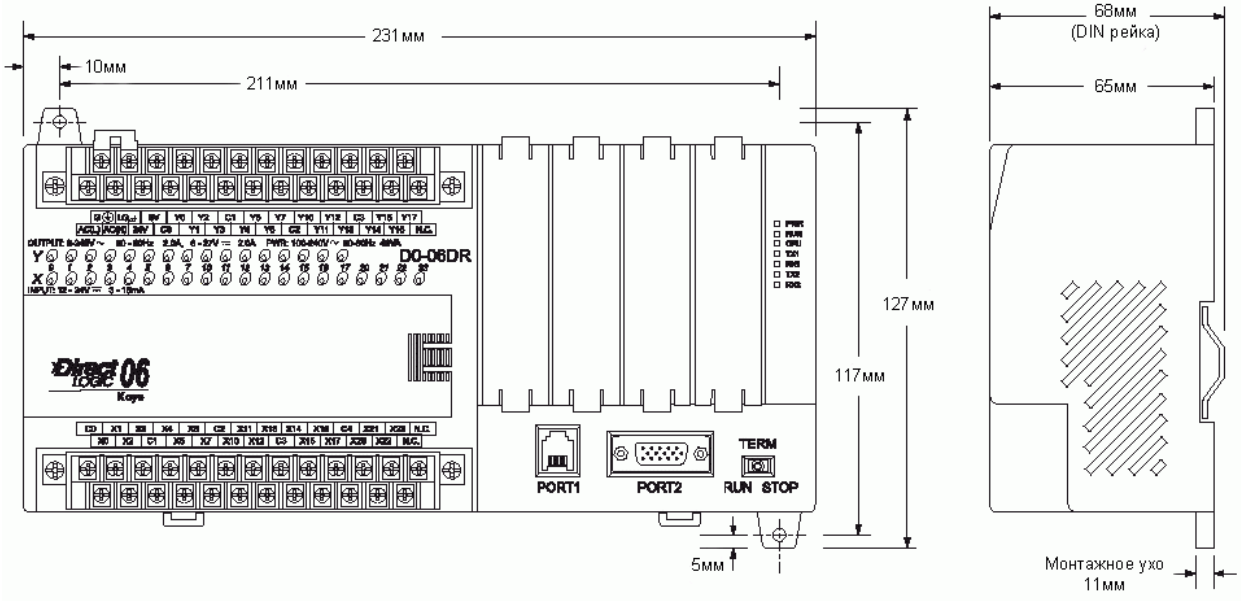
Монтаж

Для правильного монтажа системы ПЛК следует учитывать ряд параметров. Не забывайте о следующем:

- Параметрах окружающей среды
- Требованиях к электропитанию
- Соответствии инструкциям различных ведомств
- Выборе монтажного шкафа и размерах компонентов.

Размеры устройства

На следующем рисунке показаны наружные размеры и расположение крепежных элементов для всех моделей DL06. Вы должны придерживаться указаний инструкции по установке для соблюдения соответствующего расстояния между компонентами.



Монтажный шкаф

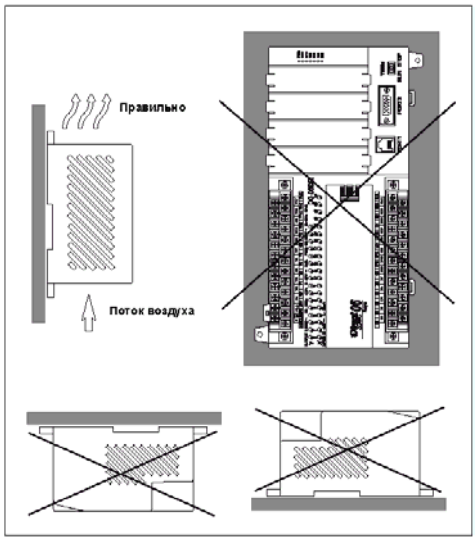
Для обеспечения безопасного и правильного функционирования системы DL06 очень важно правильно выбрать монтажный шкаф. Разнообразие приложений систем DL06 подразумевает и наличие совершенно разных параметров и технических характеристик. Но в число обязательных требований к защитным кожухам для микро ПЛК входят следующие:

- Соответствие стандартам электросоединений
- Безопасность при работе в промышленной среде
- Общее заземление
- Поддержание определенной температуры окружающей среды
- Доступ к оборудованию для монтажа
- Охрана или ограниченный доступ
- Наличие достаточного места для правильной установки и техобслуживания оборудования

План размещения в шкафу

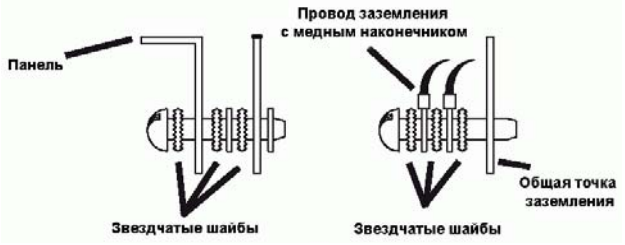
При размещении ПЛК в шкафу стоит подумать о многом. Вся информация схематично представлена на рисунке.

Примечание: существуют и другие дополнительные требования, зависящие от специфики вашего приложения и наличия других компонентов в монтажном шкафу.

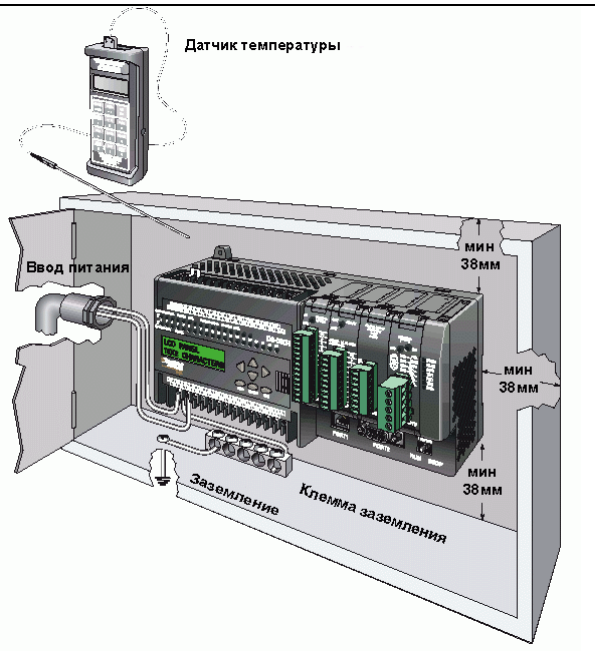


1. Чтобы обеспечить необходимое охлаждение оборудования, установите ПЛК горизонтально. Не следует монтировать блоки DL06 вертикально, вверх дном, а также на плоскую горизонтальную поверхность. Если вы размещаете два устройства в один корпус, необходимо сохранить зазор между ними не менее 183 мм.
2. Между самим ПЛК и стенками шкафа управления должен быть зазор не менее 39 мм. Не забывайте также о панелях оператора и других блоках, устанавливаемых на дверь.
3. Между ПЛК и любыми коммутационными линиями, проложенными параллельно клеммам, должен быть зазор не менее 78 мм.

ПРИМЕЧАНИЕ: *Минимальное расстояние между панелью двери или любого другого устройства,установленного на двери и ближайшим компонентом системы DL06 должно быть 38мм.*



4. Необходимо соединить зажим заземления на корпусе DL06 с точкой общего заземления. С целью уменьшения полного сопротивления лучше всего использовать медный многожильный провод. Для обеспечения хорошего контакта соединяемых поверхностей медные наконечники должны быть закручены и спаяны с концами многожильного провода.



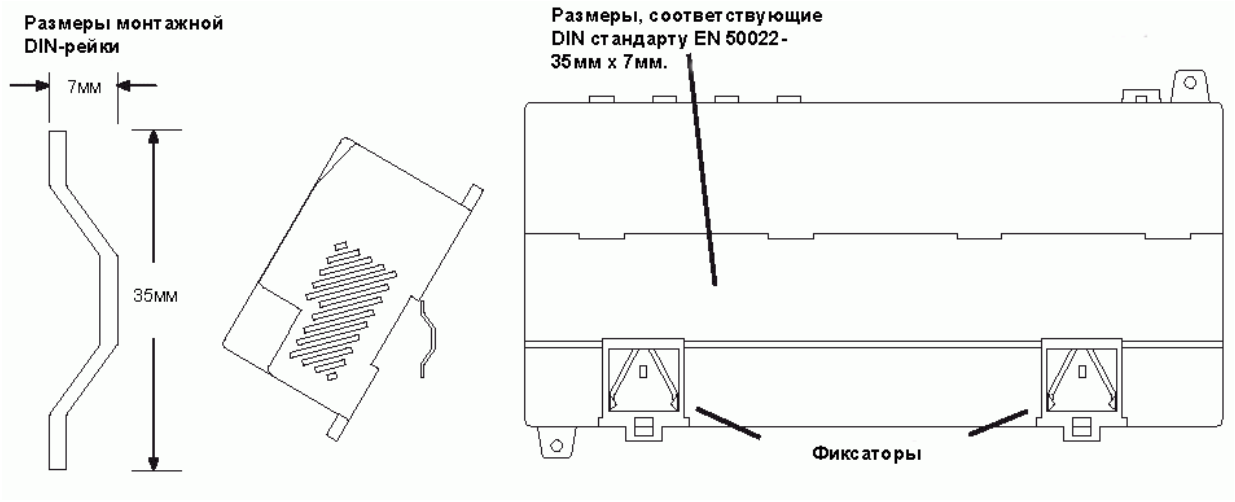
5. Для всех компонентов на панели должна быть одна общая точка заземления (например, медная шина), которой необходимо обеспечить путь возврата тока через землю. Общая точка заземления должна быть соединена с конечным заземлением панели. А заземление панели, в свою очередь, — с выходом на землю. Спецификация проводов, цвет изоляции и общие параметры системы безопасности должны соответствовать всем местным требованиям и стандартам по электротехнической безопасности.

6. Для корректного функционирования систем DL06 необходимо правильное общее заземление. Один полюс всех цепей управления и цепей подачи питания, а также экран гибкого экранированного кабеля, должны быть соответствующим образом соединены с общим заземлителем. Существует несколько способов обеспечения адекватного общего заземления:
а) Установка заземляющего электрода как можно ближе к панели.
б) Подключение к заземлению системы подачи питания.
7. Оцените варианты установки системы, при которых температура окружающей среды примет максимальное или минимальное значение, указанное в спецификации. Если Вы допускаете, что температура окружающей среды не будет соответствовать спецификации DL06, то для поддержания требуемых спецификацией температурных параметров потребуется установка нагревательного/охлаждающего элемента.
8. Системы DL06 предназначены для эксплуатации в сетях 95-240В переменного тока или 12-24В постоянного тока. Электроэнергия в некоторых случаях, где установлен микроконтроллер, не всегда стабильна и возмущения могут вызывать скачки напряжения. Из-за этого для защиты DL06 от скачков напряжения и электромагнитных помех рекомендуется использовать фильтры. Фильтры для использования с питанием ~120В и ~240В, 1-5А, можно приобрести в ООО «ПЛК-С», однако, Вы можете использовать фильтр на Ваш выбор. Эти блоки легко устанавливаются между источником питания и контроллером.

***ПРИМЕЧАНИЕ**: Если Вы используете в системе другие компоненты, не забудьте обратиться к соответствующему руководству по эксплуатации и выяснить, как данные компоненты повлияют на общие установочные размеры системы.*

Использование монтажных реек

Микроконтроллер серии DL06 можно закрепить на панели с помощью монтажных реек. Рекомендуем Вам использовать рейки, соответствующие стандарту DIN 50 022. Размеры реек: ширина - 35мм и высота - 7мм. При установке микро ПЛК на монтажные рейки обязательно используйте с каждой стороны ПЛК фиксирующие скобы. Фиксаторы предотвратят горизонтальное скольжение ПЛК по кронштейну, тем самым уменьшая вероятность случайного вытягивания слабо закрепленных проводов. В нижней части ПЛК находятся два небольших фиксатора. Для установки ПЛК на рейке необходимо после размещения ПЛК осторожно нажать на фиксаторы для полного закрепления прибора на рейке. Для снятия ПЛК потяните вниз фиксаторы, немного приподнимите ПЛК, а затем снимите его с рейки.



***ПРИМЕЧАНИЕ**: Посмотрите наш каталог или сайт чтобы узнать полную номенклатуру поставляемых нами различных аксессуаров для систем подключения на базе DIN-реек.*

Параметры окружающей среды

В данной таблице представлены параметры окружающей среды, соответствующие потребностям микроконтроллера DL06. Диапазоны значений, специфичные только для ручного программатора, приведены в конце таблицы. Некоторым типам выходных цепей будут соответствовать грвфики ухудшения параметров в зависимости от температуры окружающей среды и числа включенных выходов. Для рассмотрения параметров приобретенной модели микроконтроллера серии DL06 обратитесь к соответствующему разделу этой главы.

Характеристика	Диапазон
Температура хранения	- 20^0C - 70^0C
Рабочая температура*	0^0C - 55^0C
Рабочая влажность**	5% - 95% относительной влажности, без конденсата
Виброустойчивость	MIL STD 810C, Метод 514.2
Ударопрочность	MIL STD 810C, Метод 516.2
Помехоустойчивость	NEMA (ICS3--304)
Атмосферная среда	Без агрессивных газов

* Рабочая температура для ручного программатора и DV-1000: от 0° до 50°C
Температура хранения ручного программатора и DV--1000: от -20° до 70°C.
**Оборудование будет функционировать вплоть до 5% относительной влажности. Однако проблемы электростатики возникают намного чаще именно при низком уровне влажности (ниже 30%). Убедитесь в том, что Вы приняли надлежащие меры безопасности перед контактом с оборудованием. Для этих целей можно использовать заземляющие перемычки, антистатическое покрытие полов и т.д. (если Ваше оборудование установлено в среде с низкой влажностью)

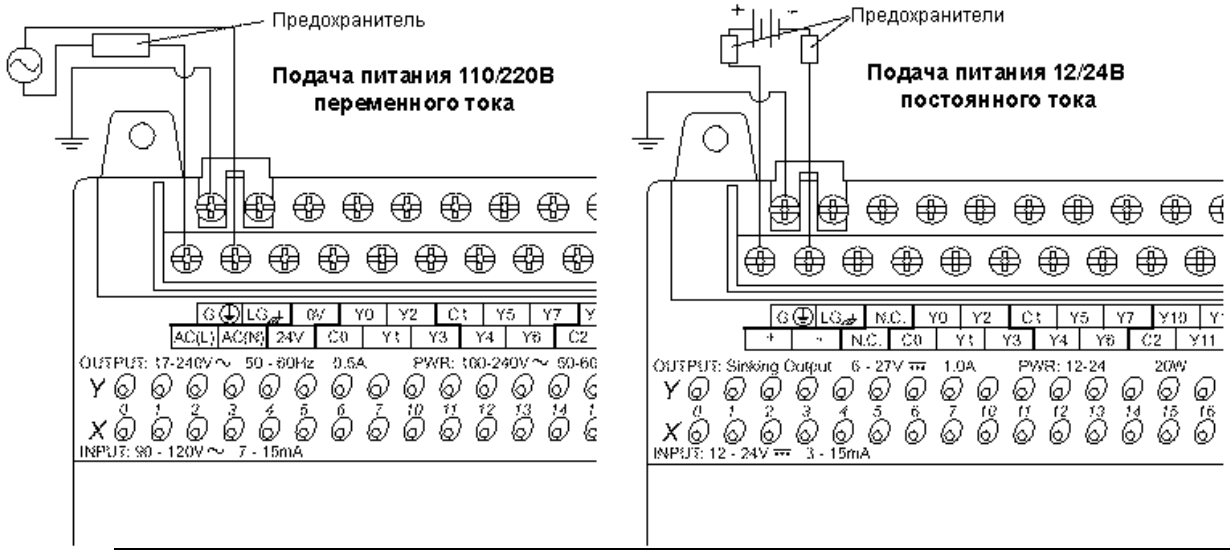
Соответствие инструкциям различных ведомств

В некоторых приложениях часть компонентов должна соответствовать определенным инструкциям различных ведомств. Оформление систем микро ПЛК серии DL06 находится в юрисдикции следующих учреждений:

- UL (Лаборатория по технике безопасности — организация UL США.),
- CUL (Лаборатория по технике безопасности — организация CUL Канада),
- CE Совет Европы(Европейский Экономический Союз)

Подключение питания

На следующем рисунке показано подключение внешнего источника питания к DL06. Соблюдайте все меры предосторожности, изложенные раннее в этом руководстве. Когда монтаж закончен, верните на место крышки разъемов до подключения питания.



ПРЕДУПРЕЖДЕНИЕ: Сразу после подключения питания необходимо закрыть защитную пластину клеммной колодки. Когда защитная пластина снята, существует опасность поражения электрическим током при случайном прикосновении к клеммам соединений или проводам подвода питания.

Установка предохранителя в цепи подачи питания

Во входных цепях питания внутренние предохранители не устанавлены, поэтому для обеспечения безопасности рабочей среды и оборудования необходимо установить внешнюю защиту электрической цепи. Для соответствия спецификациям UL/CUL необходимо установить предохранитель на цепь подвода питания. В зависимости от параметров используемой сети при установке предохранительных элементов Вам помогут следующие рекомендации:

Цепь переменного тока 208/240 Вольт

При работе оборудования в цепи 208/240В переменного тока (источником напряжения может быть как понижающий трансформатор, так и просто две фазы) необходимо установить предохранители на линию (L) и нейтральный провод (N). Рекомендуется предохранитель на 1.0А (быстродействующий).

Цепь переменного тока 110/125 Вольт

При питании от сети переменного тока 110/125В необходимо установить предохранитель только на подводящий провод (L). Нет необходимости защищать нулевой (N) провод. Рекомендуется предохранитель на 1.0А (быстродействующий).

Цепь постоянного тока 12/24 Вольт

При работе на более низких напряжениях, размер провода столь же важен как и установка предохранителя. Использование больших проводников минимизирует падение напряжения. Клеммы микроконтроллера DL06 предназначены для подключения одного провода 1.5мм^2 или двух проводов 0.75мм^2. Рекомендуемый размер плавкого предохранителя для цепи питания постоянного тока DL06 – 1.5A (например, Littlefuse 312.001 или эквивалентный).*

Внешний источник питания

Источник питания должен обеспечивать напряжение и силу тока в соответствии с индивидуальными спецификациями микроконтроллера, согласно следующим параметрам:

Параметр	DL06 с питанием переменным током	DL06 с питанием постоянным током
Входное напряжение	~110/220 В (~95-240В)	=12-24В (=10.8-26.4В)
Максимальный ток	13А, 1мс (~95-240 В) 15А, 1мс (~240-264В)	10А
Максимальная мощность	30 ВА	20 Вт
Выдерживаемое напряжение	1 минута при 1500В переменного тока между резервным, штатным и эксплуатационным заземлением	
Сопротивление изоляции	> 10 МОм при 500В постоянного тока	

ПРИМЕЧАНИЕ. *Параметры всех внутренних цепей указаны только с учетом ПЕРВИЧНОЙ ИЗОЛЯЦИИ.*

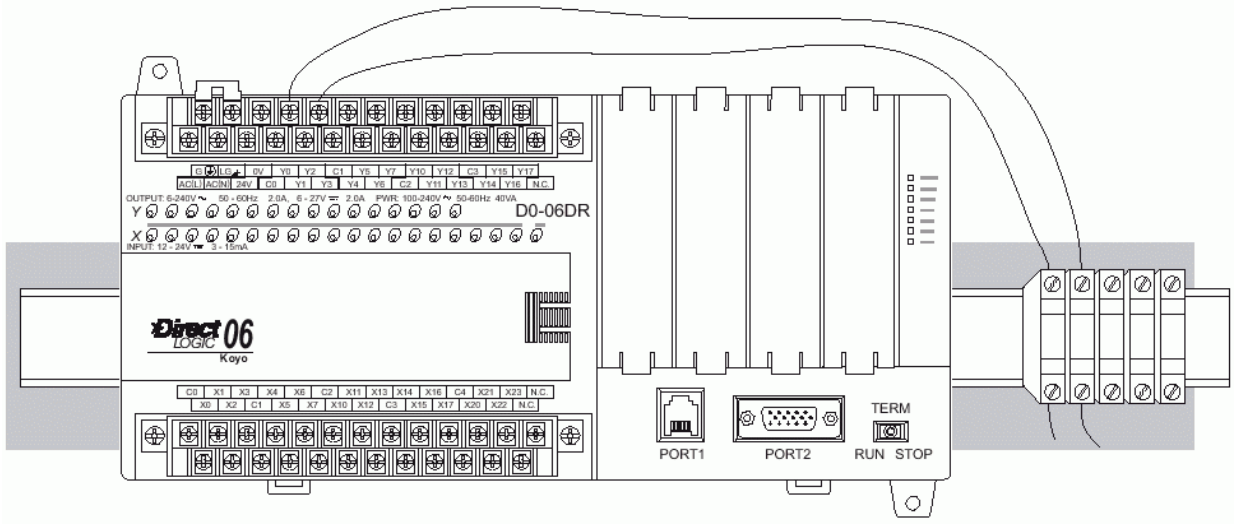
Размещение электропроводки

Следующие инструкции дают общие сведения о том, как выполнять соединения входов/выходов с микроконтроллера серии DL06. Более детальная информация по электромонтажу конкретной модели ПЛК приводится в соответствующих таблицах, которые приводятся позднее в этой главе.

1. К каждой клемме ПЛК DL06 можно подсоединить один провод 16 AWG(1.5 мм^2) или два провода 18 AWG. (0.75 мм^2). Не превышайте рекомендуемое значение.
2. Всегда используйте провода соответствующей длины. Не сращивайте провода для достижения нужной длины соединения.
3. Используйте наименьшую длину провода.
4. По возможности используйте для укладки проводов кабельные лотки.
5. Не размещайте провода вблизи кабелей высокого напряжения.
6. По возможности избегайте размещения входной проводки вблизи кабелей выхода.
7. Для уменьшения падения напряжения в случае, когда кабели прокладываются на большое расстояние, лучше использовать многожильный провод.
8. По возможности старайтесь не размещать провода постоянного тока вблизи проводки переменного тока.
9. В цепи должно быть как можно меньше остроугольных изгибов проводки.
10. Установите рекомендуемый фильтр в цепь питания для уменьшения бросков напряжения и электромагнитных помех.

Защита цепей ввода и вывода с помощью предохранителей

Входные и выходные цепи микроконтроллера DL06 не снабжены внутренними предохранителями. Для защиты их мы рекомендуем добавить в проводку ввода/вывода внешние предохранители. К каждому общему проводу можно подсоединить плавкий быстродействующий предохранитель с номинальным значением тока ниже, чем общее номинальное значение в блоке ввода/вывода. Можно установить на каждый выход предохранитель с номинальным значением, немного меньшим, чем максимальное значение тока на один выход в цепи. Далее в спецификациях на различные модели микроконтроллеров, можно найти значение максимальной силы тока на выходе (на одну точку или на общий провод). Использование внешнего предохранителя не гарантирует поломки микро ПЛК, но обеспечивает более полную защиту.



Нумерация точек ввода/вывода

Все модели серии DL06 имеют фиксированную конфигурацию ввода/вывода. Она базируется на восьмеричной системе счисления, применяемой и в других ПЛК семейства DirectLogic, начиная счет с точек X0 и Y0. Буква X всегда используется для обозначения точек ввода, а буква Y — точек вывода (выхода).

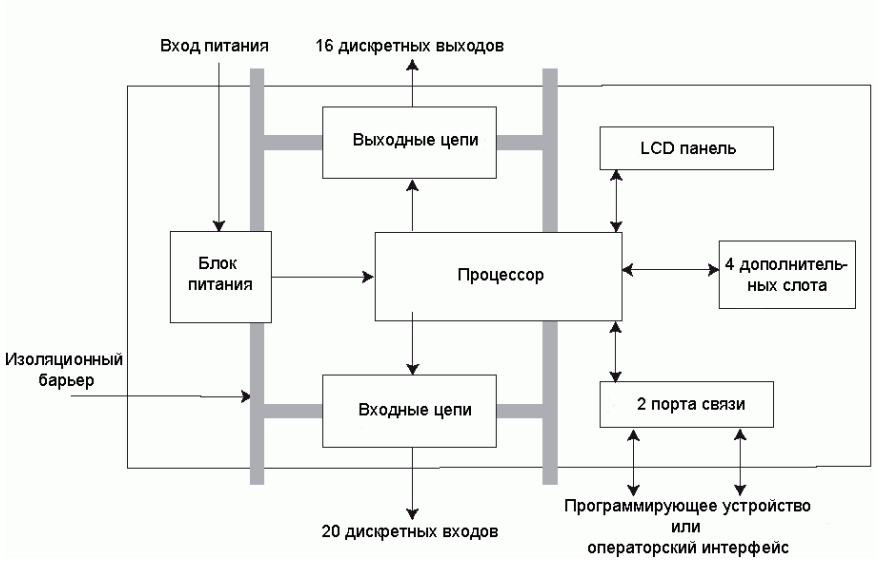
Нумерация ввода/вывода всегда начинается с ноля и не включает в себя цифры 8 и 9. Обычно адреса присваиваются группами по 8 или 16 элементов, в зависимости от количества точек в группе ввода/вывода. В семействе DL06 двадцати вводам соответствуют номера X0 – X23.Шестнадцать точек вывода проходят под номерами Y0 – Y17.

Основные принципы электромонтажа системы

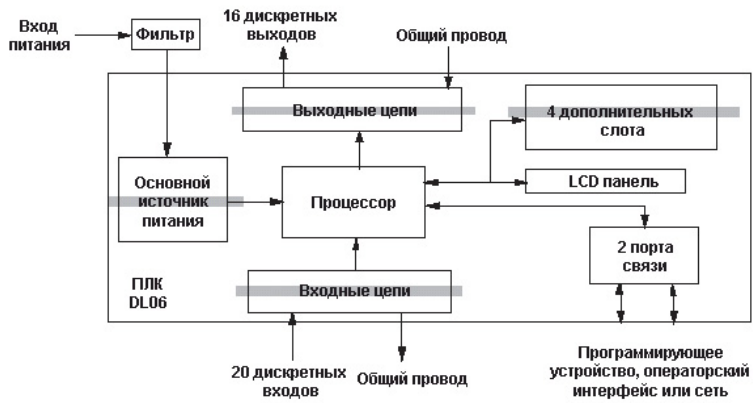
Семейство микроконтроллеров DL06 представляет собой очень гибкую систему, способную работать в различных конфигурациях соединений. Изучив этот раздел перед монтажом системы, Вы сможете подобрать наиболее подходящий вариант подключения для Вашего оборудования. Это поможет снизить себестоимость системы, предотвратит ошибки в электромонтаже и проблемы с безопасностью.

Изоляционные барьеры ПЛК

Схему ПЛК можно разделить на три основных области, разделяемых изоляционными барьерами (показаны на следующем рисунке). Электрическая изоляция обеспечивает безопасность таким образом, что пробой в одной области не приведет к поломкам в другой. Фильтр на цепь подвода питания обеспечит изоляцию между источником питания и электрической сети. Трансформатор в источнике питания обеспечивает электромагнитную изоляцию между основным и вторичным контуром. Оптопары обеспечивают оптическую изоляцию в цепях ввода и вывода. Это отделяет логические схемы от рабочей области, где подсоединено оборудование. Обратите внимание на то, что дискретные входы изолированы от дискретных выходов, так же как каждый элемент изолирован от логического узла системы. Изоляционные барьеры защищают операторские панели (и самого оператора) при авариях в цепи подвода питания или внешних устройств. Во время электромонтажа ПЛК очень важно не делать внешних соединений, подключающих логические схемы к остальным контурам.



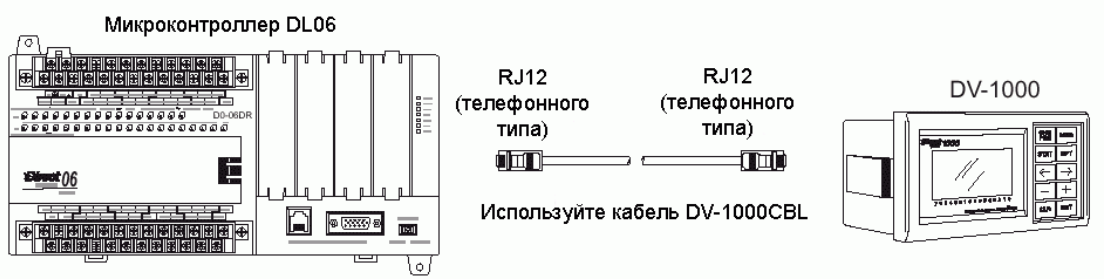
На следующем рисунке представлена внутренняя конфигурация ПЛК DL06 (со стороны передней панели).



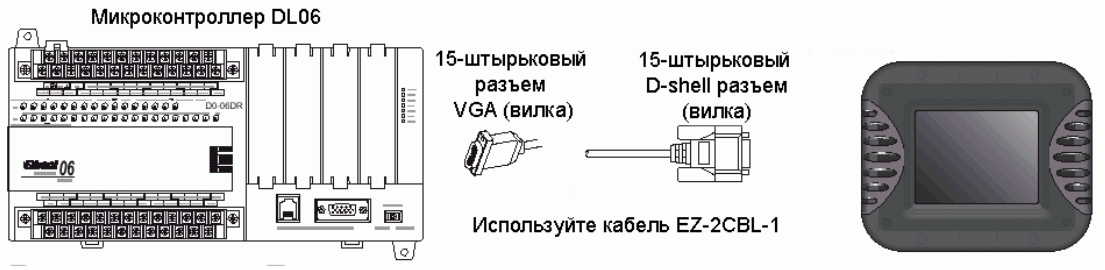
Подключение панелей оператора

Панели оператора соответствующим образом подсоединяются к источникам данных и питания системы. Интерфейсам с ЭЛТ обычно требуется отдельный источник питания переменного тока. Однако небольшие по размеру панели оператора, например, широко известные DV-1000 и OP-400 способны питаться непосредственно от микроконтроллера DL06.

Подсоедините DV-1000 к коммуникационному порту 1 Вашего микроконтроллера DL06 с помощью кабеля, показанного на рисунке ниже. Обычный кабель содержит провода передачи/приема информации и провод питания на +5 вольт.

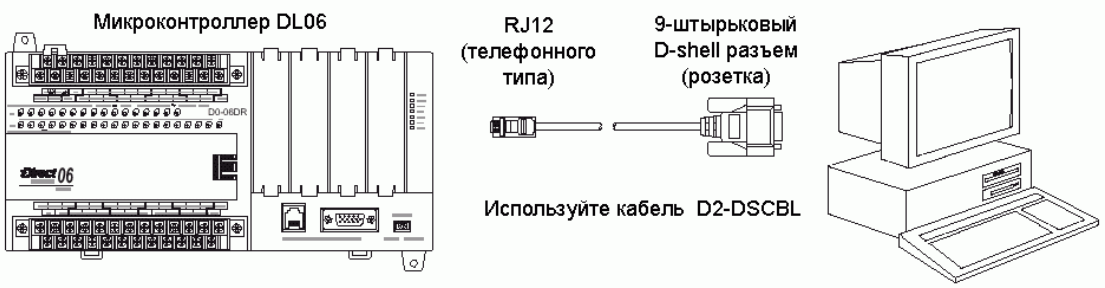


Для операторских панелей EZTouch и EZText потребуются отдельные соединения для связи и подключения питания. Подсоедините DL06 к соответствующему D-разъему сзади операторской панели помощью кабеля , показанного на рисунке ниже. Этим панелям потребуется питание =24В (рабочий диапазон =20-30В).

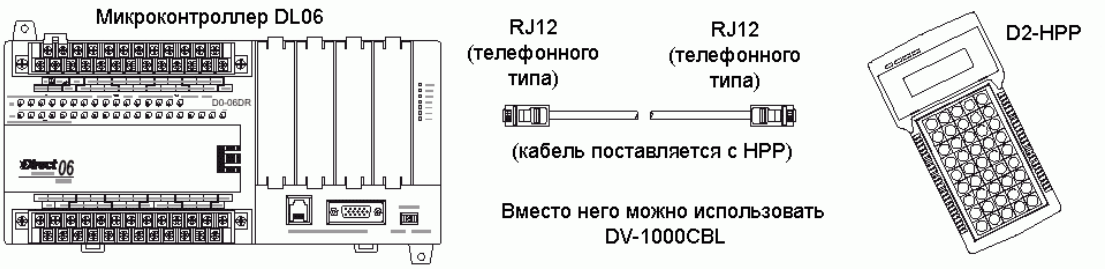


Подключение устройств программирования

Микроконтроллер DL06 можно запрограммировать с помощью ручного программатора или компьютерной программы DirectSOFT. Подсоедините DL06 к компьютеру с помощью кабеля, показанного на рисунке ниже.



Ручной программатор D2-HPP поставляется с кабелем связи. В качестве замены используйте кабель, изображенный на рисунке ниже.



Понятия приемник/ источник

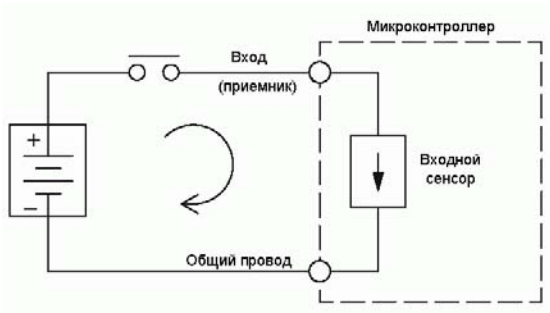
Перед дальнейшим изучением схем подключения Вы должны иметь полное понимание понятий «приемник» и «источник» тока. Эти термины часто используются при описании дискретных входных и выходных цепей постоянного тока. Цель данного раздела — облегчить понимание этих понятий. Сначала приводится краткое определение, за которым следуют практические приложения.

Приемник = предусматривает замыкание цепи на землю (-)

Источник = предусматривает замыкание цепи на источник питания (+)

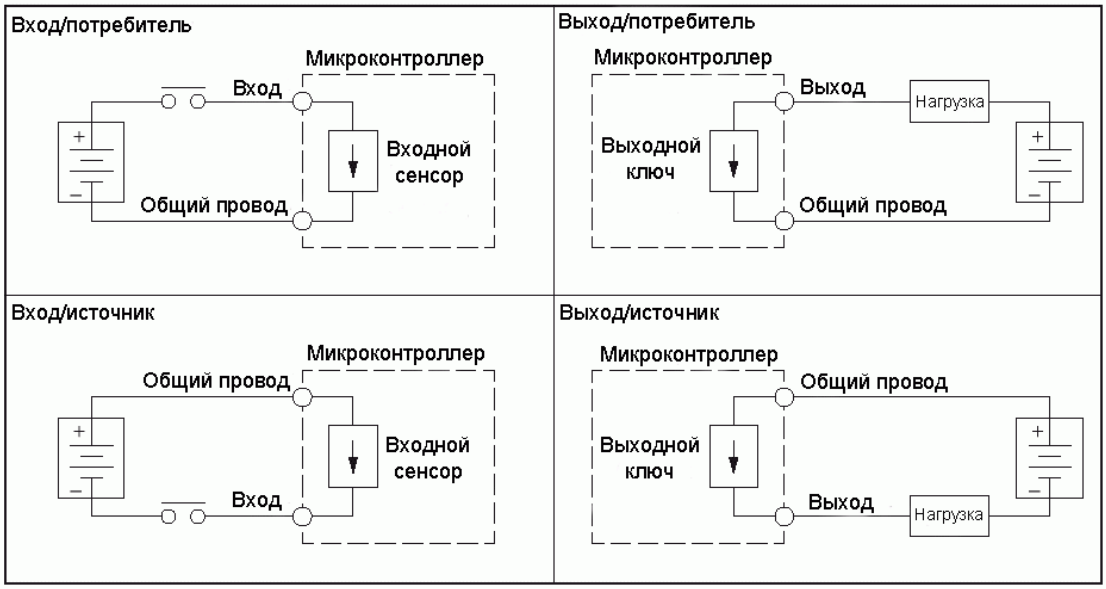
Обратите внимание на полярность. *Понятия «приемник» и «источник» относятся только к цепям постоянного тока.* Входные и выходные точки, которые являются приемниками либо источниками, могут проводить ток только в одном направлении. Это означает, что можно подсоединить внешний источник питания и полевые 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(-).



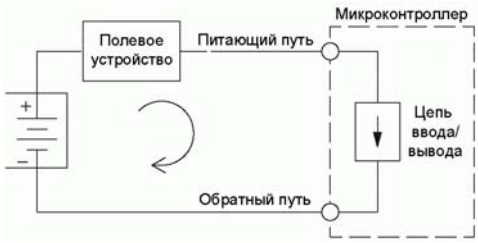
Начните с входной клеммы ПЛК, продолжайте через цепи считывания входа ПЛК и закончите на общем полюсе, соедините источник питания (-) с общим полюсом. После добавления выключателя между источником питания (+) и входом вы замкнете цепь. Если замкнуть выключатель, ток потечет в направлении, указанном стрелками.

Применяя указанный принцип построения цепи, получим показанные ниже четыре возможные схемы входного/выходного приемника/источника. У микроконтроллеров DL06 входы могут быть приемником или источником и выходы или приемником или источником. Любая пара схем ввода/вывода, показанных ниже, возможна с одной из моделей DL06.

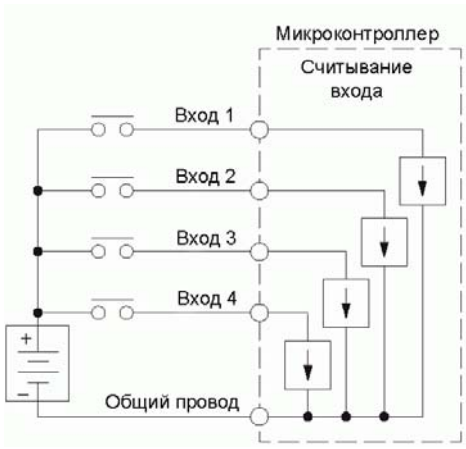


Понятие «Общего полюса» входа/выхода

Для того чтобы цепи Ввода/Вывода работали, необходимо, чтобы ток входил в одну клемму и выходил через другую. Поэтому с каждым дискретным Вводом/Выводом 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).

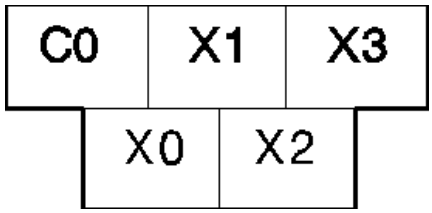


Практически все дискретные входы или выходы на ПЛК совместно используют путь возврата с другими точками входа/выхода. На рисунке справа показана группа (или блок) из 4 входа, которые совместно используют обратный общий провод. В этом случае четыре входа требуют только пять клемм вместо восьми.



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 общий провод. Большего сечения.*

Большинство входных и выходных цепей DL06 объединены в блоки, которые совместно используют обратный провод. Хорошим указателем такой группы Ввода/Вывода являются монтажные метки. Общие блоки отделены жирной линией. Более тонкая линия отделяет входы связанные с тем общим проводом. На рисунке справа X0, X1, X2 и X3 совместно используют общий провод C0, расположенный слева от X1.



Приведенная ниже маркировка показывает наличие пяти групп по 4 входа и четырех групп по 4 выхода. Для каждой группы существует один общий канал.

G ⏚	LG ⏦	0V	Y0	Y2	C1	Y5	Y7	Y10	Y12	C3	Y15	Y17
AC(L)	AC(N)	24V	C0	Y1	Y3	Y4	Y6	C2	Y11	Y13	Y14	Y16	N.C.

C0	X1	X3	X4	X6	C2	X11	X13	X14	X16	C4	X21	X23	N.C.
X0	X2	C1	X5	X7	X10	X12	C3	X15	X17	X20	X22	N.C.	

Следующая маркировка предназначена для моделей с выходом постоянного тока (приемник), таких как D0-06DD1 и D0-06DD1-D. Один общий контакт используется для каждой группы из четырех выходов, и один обозначенный контакт предназначен для питания выходных цепей.

G ⏚	LG ⏦	0V	Y0	Y2	C1	Y5	Y7	Y10	Y12	C3	Y15	Y17
AC(L)	AC(N)	24V	C0	Y1	Y3	Y4	Y6	C2	Y11	Y13	Y14	Y16	+V

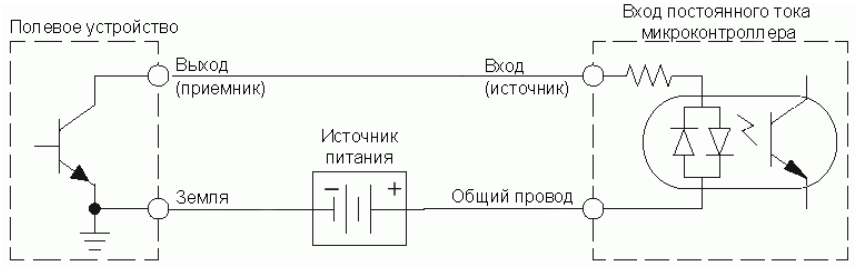
C0	X1	X3	X4	X6	C2	X11	X13	X14	X16	C4	X21	X23	N.C.
X0	X2	C1	X5	X7	X10	X12	C3	X15	X17	X20	X22	N.C.	

Подключение точек ввода/вывода постоянного тока к полупроводниковым полевым устройствам

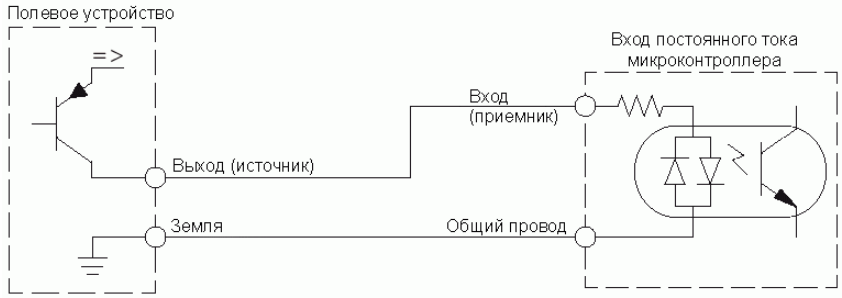
В предыдущем разделе понятия Приемник и Источник для цепей ввода/вывода постоянного тока объяснялись возможностью тока течь только в одном направлении. Это справедливо также для многих полевых устройств, которые имеют полупроводниковую (транзисторную) аппаратуру сопряжения. Другими словами, полевые устройства также могут быть «приемниками» и «источниками». *Когда два устройства соединяются последовательно в цепи постоянного тока (как например в случае, когда подключается полевое устройство к входам или выходам постоянного тока микроконтроллера), то один из них должен соединяться как «источник», а другой — как «приемник» тока.*

Полупроводниковые входные датчики

Некоторые входные цепи постоянного тока DL06 могут принимать ток в любом направлении, поэтому они могут подключаться как «источник», либо как «приемник». В следующей цепи полевое устройство имеет транзисторный выход с открытым коллектором NPN. К нему подается ток от входной точки ПЛК, которая является «источником». Питание может подаваться от любого источника (+12 или +24В постоянного тока), который удовлетворяет техническим условиям.



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

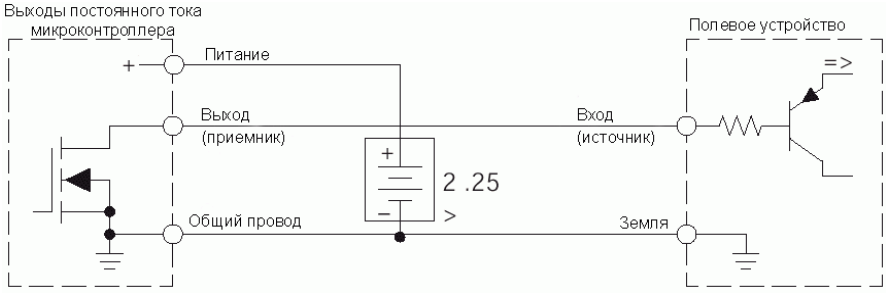


Полупроводниковая выходная нагрузка

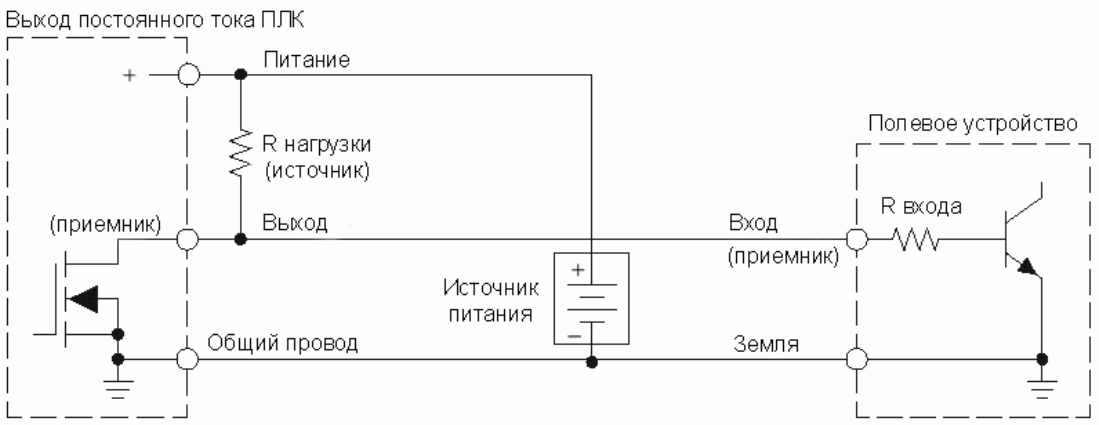
Иногда в приложении требуется соединить выходную точку ПЛК с полупроводниковым входом полевого устройства. Этот тип соединения обычно используется для управления сигналами низкого уровня мощными исполнительными механизмами.

У семейства микроконтроллеров DL06 выходы постоянного тока или только приемники или источники. Все шестнадцать выходов имеют один общий провод, даже притом, что имеются четыре отдельных общих клеммы.

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 выход ПЛК – приемник постоянного тока соединен 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 выход со входом цепи питания выходного модуля постоянного тока.



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

***ПРИМЕЧАНИЕ 2**. Применение нагрузочного резистора для реализации выхода-источника имеет своим результатом инвертирование логики выхода . Другими словами, с точки зрения релейной логики, на вход рабочего устройства подается питание, когда выход ПЛК отключен. Ваша программа должна учитывать это и вырабатывать инвертированный выход. Или вы можете делать инверсию в другом месте, 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 (типовое значение 15мА). Далее используйте (Iвхода) и напряжение внешнего источника питания для вычисления Rнагрузки. Затем вычислите мощность Pнагрузки (в ваттах) для оценки правильного значения Rнагрузки.

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

$$R_{нагрузки} = \frac{V_{источника} - 0.7}{I_{входа}} - R_{входа}$$

$$P_{нагрузки} = \frac{V_{источника}^2}{R_{нагрузки}}$$

Способы подключения реле

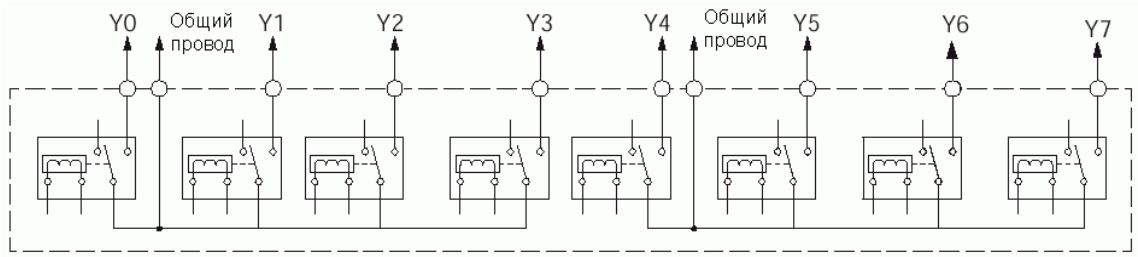
Модели D0-06AR и D0-06DR имеют релейные выходы. Реле являются наилучшим решением для следующих приложений:

- 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).

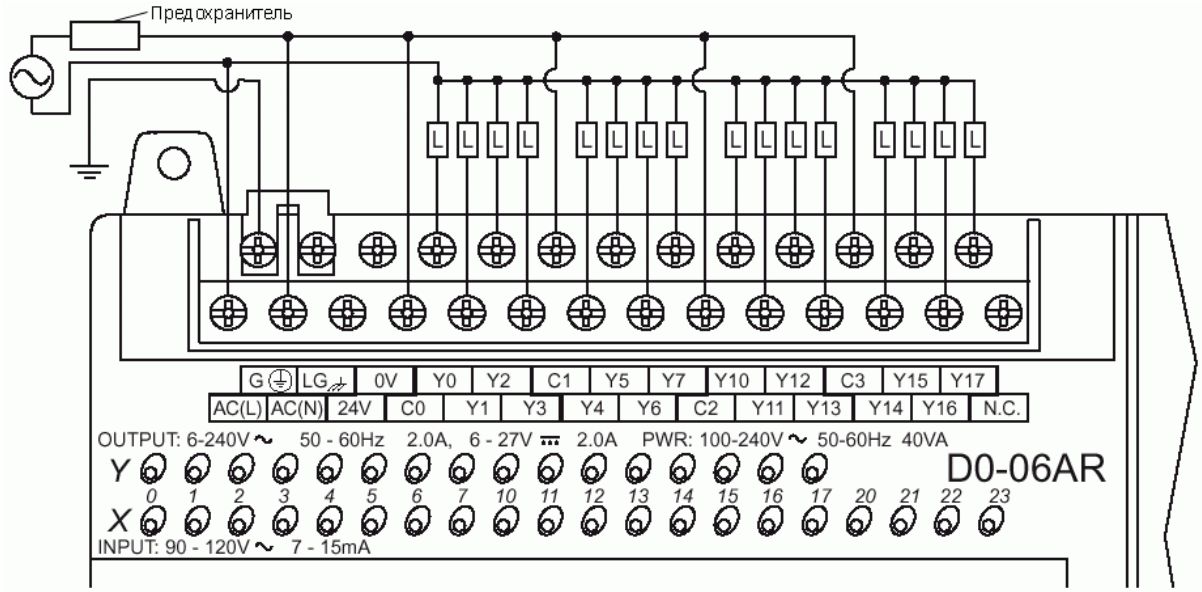
Некоторые приложения, в которых НЕЛЬЗЯ использовать реле:

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

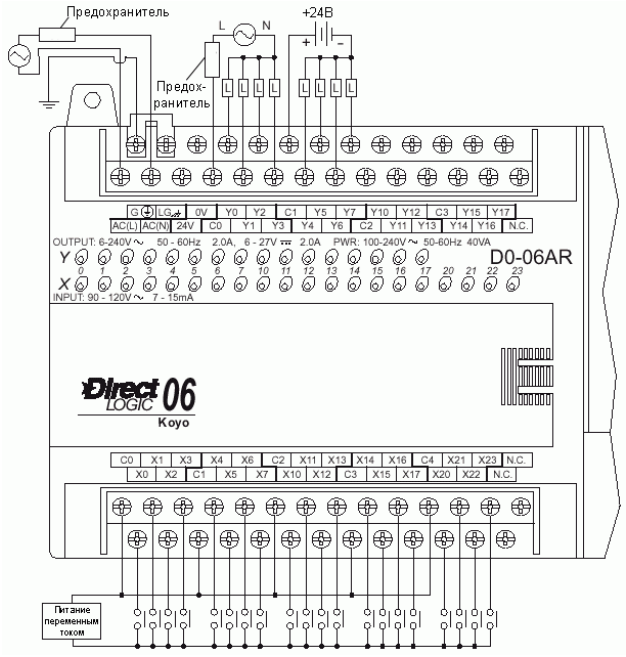
В этом разделе рассказано о различных вариантах подключения выходов реле к нагрузке. Релейные выходы DL06 - шестнадцать нормально-разомкнутых однополюсных реле. Они сгруппированы по четыре реле на один общий канал. На схеме показаны реле и внутренняя проводка ПЛК. Обратите внимание на то, что каждая группа изолирована от другой группы выходов.



На следующей схеме все элементы нагрузки используют один и тот же источник питания переменного тока, который питает DL06. В этом примере все общие каналы соединены в один.



В схеме на следующей странице нагрузки для точек Y0-Y3 используют один и тот же источник питания переменного тока, который обслуживает DL06. Нагрузки для точек Y4-Y7 используют отдельный источник питания постоянного тока. В этом примере общие каналы разделены в соответствии с тем, какой источник питает определенный элемент нагрузки.



Подавление бросков напряжения в цепи с индуктивной нагрузкой

При обесточивании приборов с индуктивной нагрузкой через контакт реле, эти приборы генерируют импульсное напряжение. При замыкании контакт реле «дрожит», что в свою очередь активизирует и обесточивает катушку до тех пор, пока «дребезг» не прекратится. Генерируемые переходные напряжения значительно превышают по амплитуде напряжение питания, особенно напряжение постоянного тока.

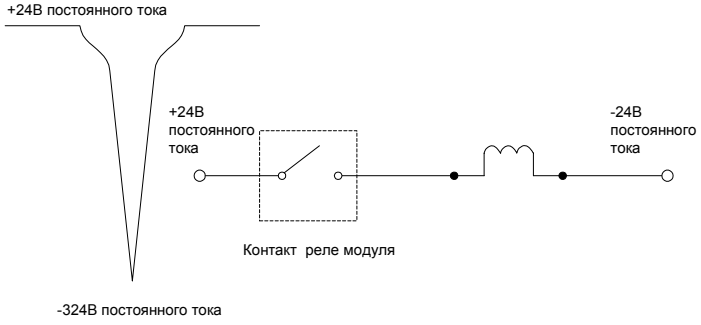
При включении индуктивной нагрузки с питанием от источника постоянного тока, в момент открытия контактов реле (или при «дребезге») в цепи появляется полное напряжение питания. При включении индуктивной нагрузки с питанием от источника переменного тока существует один шанс из 60 (60 Гц) или 50 (50 Гц), что когда синусоида переменного тока пересечет нулевой уровень, контакт реле будет открыт (или произойдет «дребезг»). Если при открытии контакта реле, напряжение не равно нулю, в индукторе сохраняется энергия, которая высвобождается в тот момент, когда неожиданно снимается напряжение. Этот выброс энергии и является причиной возникновения импульсов напряжения .

Когда приборы с индуктивной нагрузкой (двигатели, стартеры, промежуточные реле, соленоиды, клапаны, и др.) управляются с помощью релейных контактов, рекомендуется параллельно катушке периферийного устройства подключить устройство подавления выбросов напряжения. Если индуктивный прибор снабжен разъемным соединителем, то устройство подавления выбросов напряжения можно установить в клеммной колодке на выходе реле.

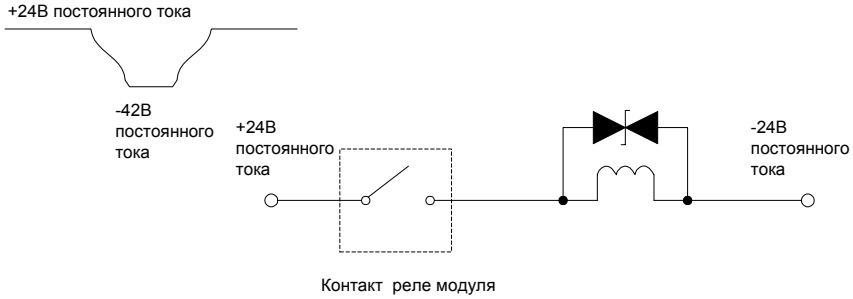
Подавители импульсного напряжения (Transient Voltage Suppressor, TVS,) обеспечивают лучшую защиту от импульсного напряжения в катушках с источниками питания постоянного или переменного тока, обеспечивая самое быстрое реагирование с наименьшим выбросом.

Далее в списке средств подавления выбросов напряжения в катушках с питанием от источников постоянного или переменного тока стоят **варисторы** на окисе металла (MOV).

Например, на следующей далее диаграмме сигнала отображена энергия, высвобожденная в момент открытия контакта, который предназначен для включения 24-вольтового соленоида (постоянного тока). Обратите внимание на большой всплеск напряжения.



На этом рисунке представлена та же самая цепь с параллельно подсоединенным фильтром - подавителем (TVS). Заметьте, что всплеск напряжения заметно уменьшился.



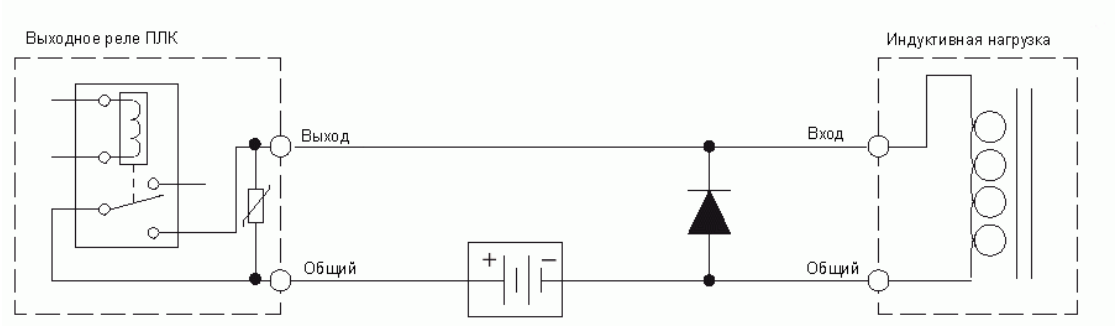
ПЛК-Системы поставляет восьмикананый модуль с TVS на 24В: ZL-TD-24

Продление срока службы контактов реле.

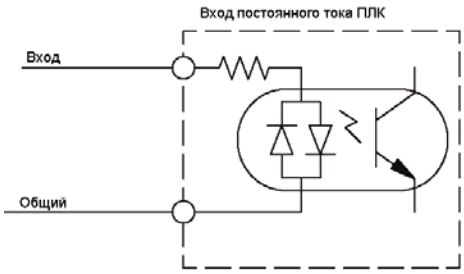
На степени износа контактов реле сказывается число совершенных включений, искрение в моменты включения и выключения, а также наличие загрязнений. Существует ряд рекомендаций, с помощью которых Вы сможете продлить срок службы контактов реле. Например, включение и выключение реле только при необходимости, и (по возможности) включение и выключение цепи нагрузки в тот момент, когда в ней присутствует ток наименьшей силы. Вдобавок, позаботьтесь о подавлении всплесков напряжения, образуемых в индуктивных элементах нагрузки (например, контакторах или соленоидах).

При наличии индуктивной нагрузки в цепи постоянного тока мы рекомендуем использовать подавляющий диод, как показано на следующей схеме (НЕ используйте эту цепь с питанием от источника переменного тока). При включении нагрузки диод имеет обратную проводимость (высокое сопротивление). После отключения нагрузки сохраненная энергия в катушке высвобождается в форме отрицательного всплеска напряжения. В этот момент диод становится в прямой проводимости (низкое сопротивление) и отводит энергию на землю. Это защищает релейные контакты от высокого напряжения, которое может возникнуть при открытии контактов.

Установите диод как можно ближе к периферийному индуктивному устройству. Используйте диод с амплитудным инверсивным напряжением не менее 100 В, 3А или более прямого тока. Выберите тип диода с быстрым восстановлением (например, тип Шотки). НЕ используйте диоды для малых сигналов, например 1N914, 1N941 и др. Перед началом работы убедитесь в том, что диод правильно установлен в цепь. Если диод установлен в обратном направлении, то при включении реле он вызовет короткое замыкание в цепи питания.



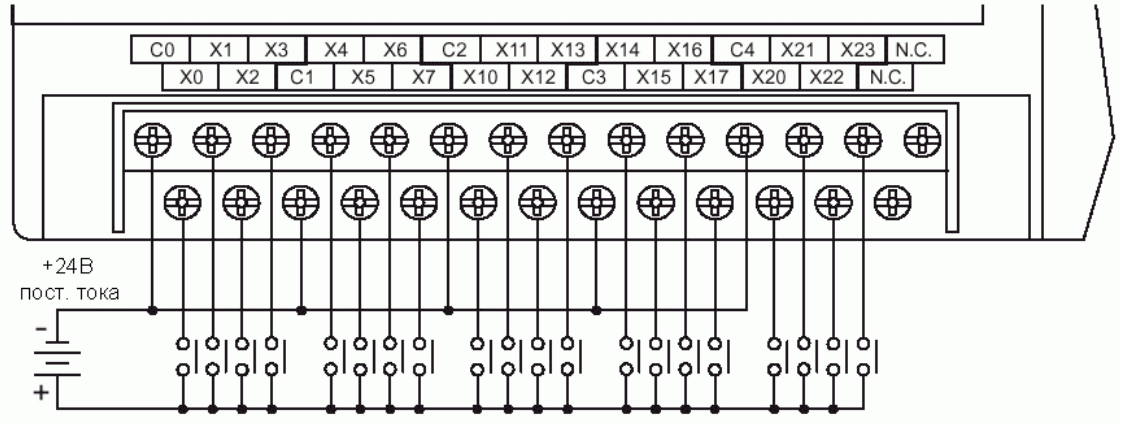
Подключение входов постоянного тока



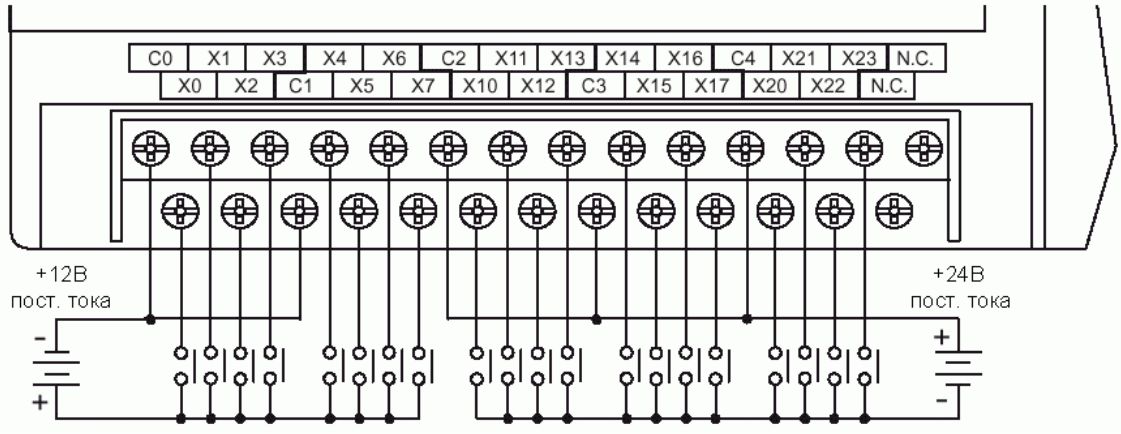
Микроконтроллер серии DL06 со входами постоянного тока представляют собой очень гибкую систему, способную работать в режиме приемника или источника. Двойные диоды (на рисунке справа) позволяют току проходить в любом направлении. Входы принимают 10.8 - 26.4 В постоянного тока. Используемые приборы работают при +12 В постоянного тока и +24 В постоянного тока. На самом деле Вы можете подсоединить каждую группу входов, связанную с одним общим и как потребитель и как источник тока.

Все каналы группы должны быть одного типа: либо приемники, либо источники.

В первом примере все входы работают приемниками, все общие объединены.



В следующем примере первые восемь входов работают приемниками, а последние двенадцать — источниками.

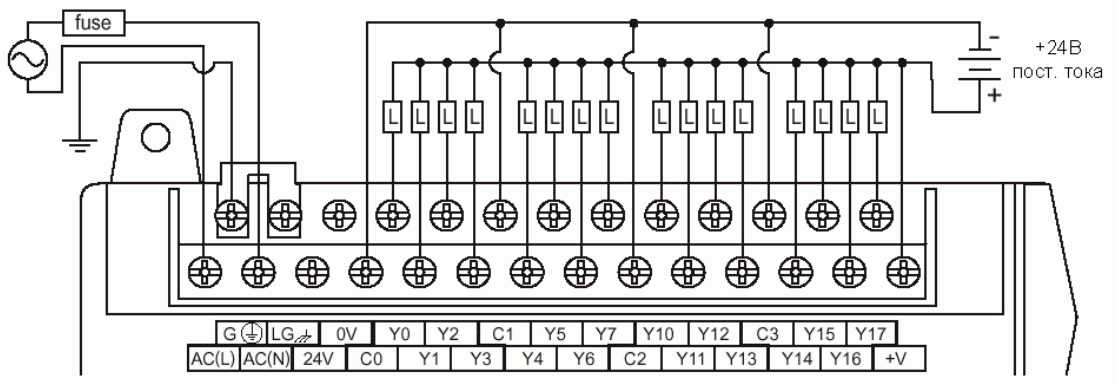


Подключение выходов постоянного тока

Выходные цепи постоянного тока DL06 представляют собой высокоэффективные транзисторные переключатели с низким сопротивлением в открытом состоянии и быстрым временем переключения. Примите к сведению следующие характеристики, присущие только выходам постоянного тока:

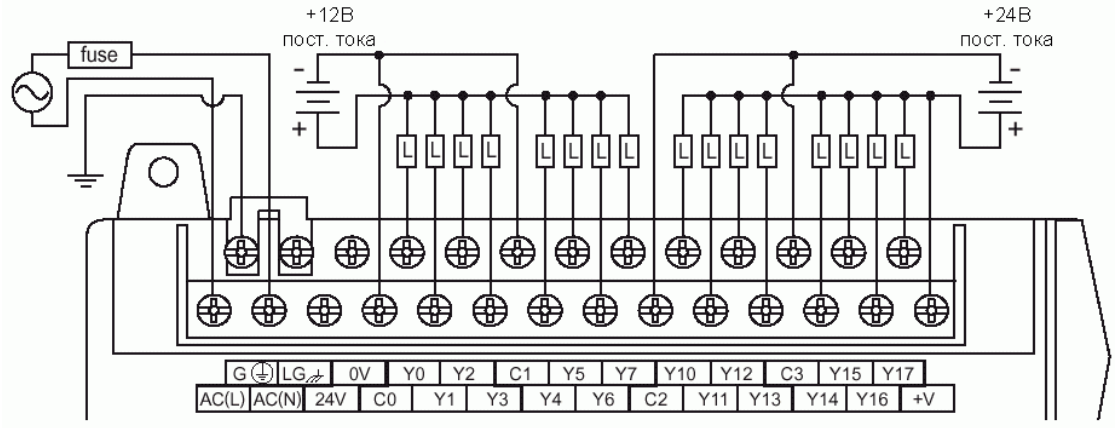
- Для всех шестнадцати выходов существует только одна общая цепь. Все шестнадцать выходов находятся в одной группе.
- Выходные ключи — только типа приемник (потребитель) или только источник. ***Для определения типа выхода у конкретной модели посмотрите подробную спецификацию в этом руководстве.***
- Для питания выходной цепи внутри ПЛК необходим внешний источник питания. Один полюс источника питания (-) нужно подсоединить к общей клемме, а другой (+) — к крайней правой клемме верхнего коннектора (+V).

В следующем примере все шестнадцать выходов получают питание от одного источника питания.



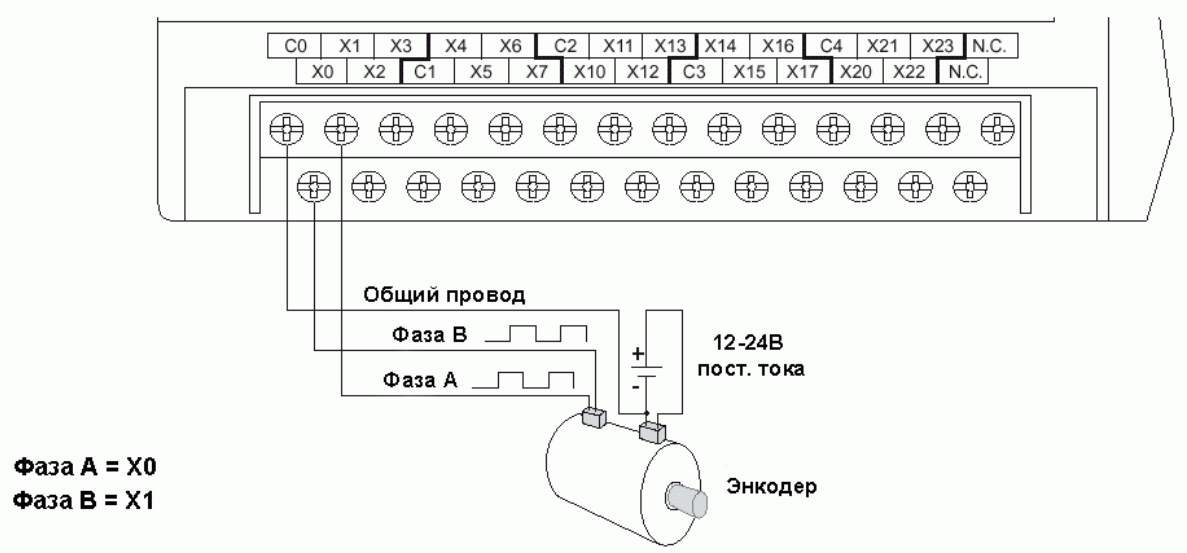
В следующем примере выходы имеют «раздельные» источники питания. Первые восемь выходов используют +12-вольтовый источник постоянного тока, а последние восемь — +24-вольтовый источник. В любом случае Вы можете «разбить» выходы на любое количество источников, пока:

- все напряжения питания находятся в заданном диапазоне
- все выходы подсоединены как цепи приемники
- все питающие контакты (-) соединены между собой

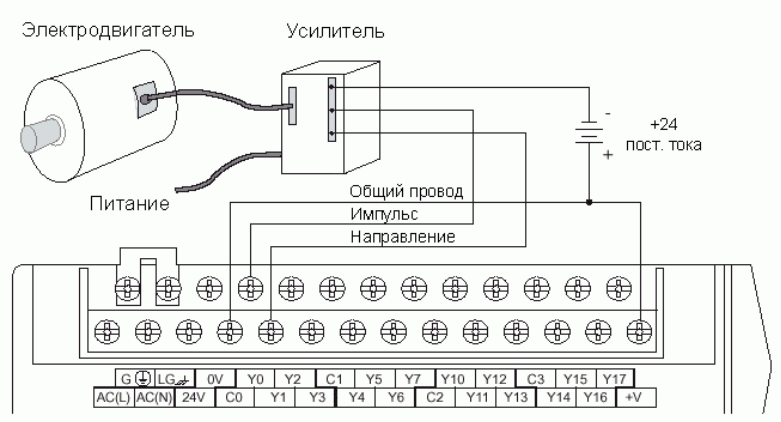


Подключение высокоскоростных входов/выходов

Модели серии DL06 с входами и выходами постоянного тока снабжены специальной схемой высокоскоростных входов/выходов. Конфигурацию цепи можно программировать, обработка отдельных точек входа/выхода идет независимо от сканирования ЦПУ. В главе 3 обсуждаются варианты программирования высокоскоростного ввода/вывода. В то время как цепь высокоскоростного ввода/вывода может работать в шести режимах, в этой главе представлены структурные схемы лишь двух наиболее популярных режимов работы. Высокоскоростной ввод возможен с контактов X0-X3. При правильной настройке (как показано на рисунке) DL06 может считать квадратурные импульсы с частотой до 7 кГц, идущие от датчика угла поворота (энкодера).



Модели DL06 с выходами постоянного тока могут использовать функцию импульсного вывода. DL06 в состоянии генерировать высокоскоростные импульсы с частотой до 10КГц для управления, например, шаговыми двигателями системами с микропроцессорным приводом. Выходы Y0 и Y1 могут генерировать импульсы и направляющие сигналы, или импульсные сигналы CCW(против часовой стрелки) и CW(по часовой стрелке) соответственно. Технические характеристики высокоскоростных входов и импульсных выходов смотрите в главе 3.



Словарь терминов в спецификации

Дискретный вход

Одно из двадцати входных соединений с ПЛК, которые преобразуют электрический сигнал, идущий от внешнего устройства, в двоичный сигнал (вкл или выкл), считываемый ЦПУ при каждом цикле ПЛК.

Дискретный выход

Одно из шестнадцати выходных соединений от ПЛК, которые преобразуют выход внутренней релейной программы (0 или 1) в операцию включения или выключения выходного переключателя. Это позволяет программе включать и выключать внешние нагрузки.

Общий контакт входа/выхода

Соединение на клеммах входа или выхода, совместно используемое несколькими цепями входа/выхода. Обычно это и есть цепь возврата к источнику питания цепи входа/выхода.

Диапазон входного напряжения

Рабочий диапазон напряжения входной цепи

Максимальное напряжение

Максимально допустимое напряжение во входной цепи.

Уровень напряжения включения

Минимальный уровень напряжения, при котором происходит включение точки входа.

Уровень напряжения выключения

Максимальный уровень напряжения, при котором происходит выключение точки входа.

Полное входное сопротивление (импеданс)

Входное полное сопротивление используется для вычисления силы тока на входе при определенном уровне рабочего напряжения.

Сила тока на входе

Характерная сила тока для активного (ВКЛ) входа.

Минимальный рабочий ток

Минимальная сила тока во входной цепи, необходимая для поддержания работы во включенном состоянии.

Максимальный потребляемый ток в отключенном состоянии

Максимальная сила тока во входной цепи, необходимая для поддержания работы в выключенном состоянии.

Время срабатывания ВЫКЛ-ВКЛ

Время, которое требуется модулю для перехода из выключенного состояния во включенное.

Время срабатывания ВКЛ-ВЫКЛ

Время, которое требуется модулю для перехода из включенного состояния в выключенное.

Индикаторы состояния

Светодиоды, отображающие состояние (ВКЛ/ВЫКЛ) входа или выхода. Все светодиоды DL06 подключены к логическим входным или выходным цепям (а не к физическим входам или выходам).

Схемы подключения и спецификации.

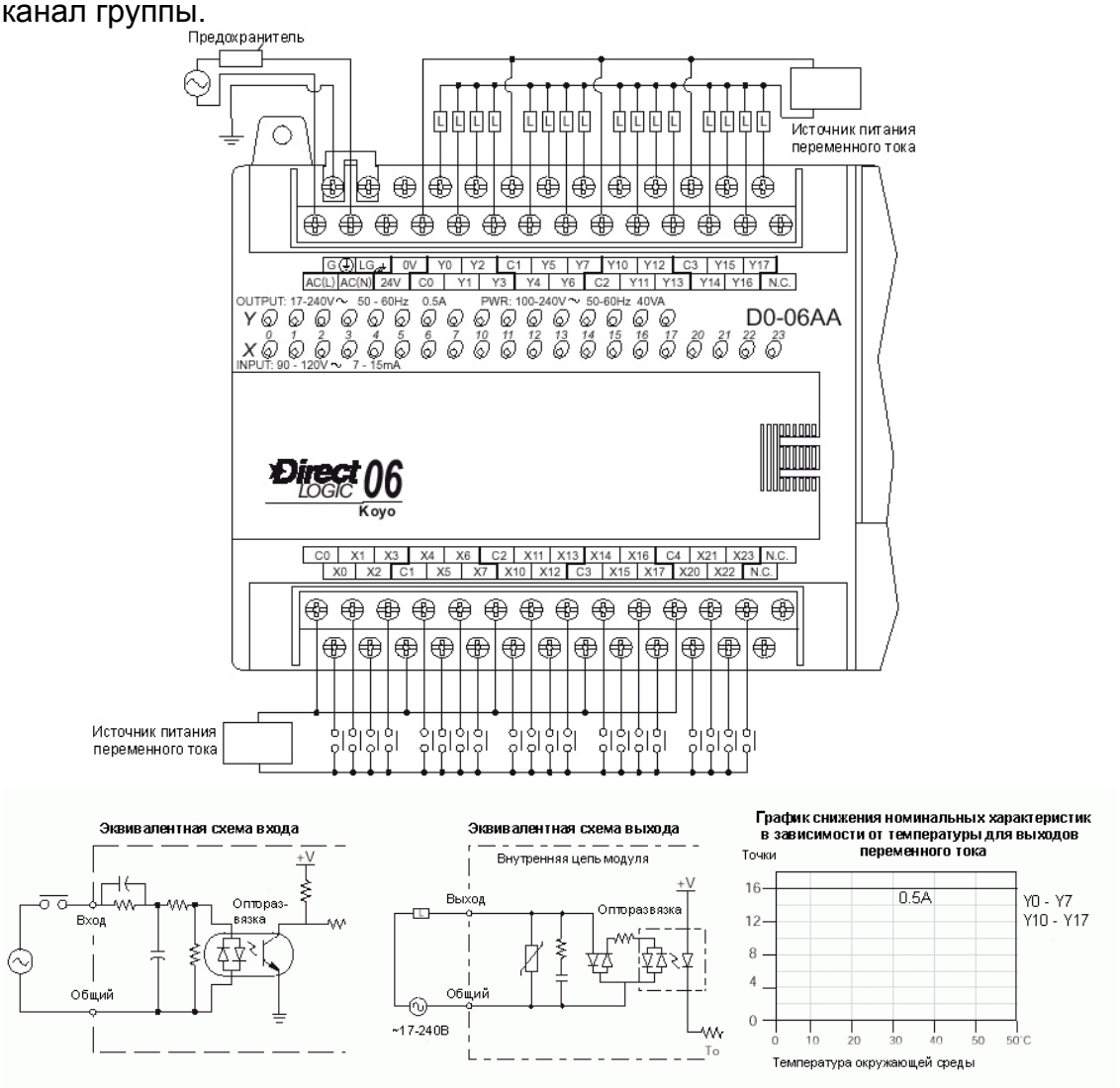
В оставшейся части главы представлена подробная техническая информация о микроконтроллерах DL06. Для каждого типа микроконтроллера приводится основная монтажная схема, эквивалентные схемы цепей входа/выхода и спецификации.

Схема соединений входов/выходов D0- 06AA

Микроконтроллер D0-06AA имеет двадцать входов переменного тока и шестнадцать выходов переменного тока. На схеме приведен пример типичного подключения внешних устройств. Как видно на схеме, для подвода внешнего источника питания используются четыре клеммы.

Входы объединены в пять групп по четыре канала. Каждой группе отводится отдельный общий провод. В приведенном ниже примере все общие цепи соединены в одну, но можно использовать отдельные источники питания и общие цепи. На эквивалентной схеме входной цепи представлен один канал группы.

Выходы объединены в четыре группы по четыре переключателя. Каждой группе отводится одна общая цепь. В приведенном ниже примере все общие цепи соединены в одну, но можно использовать отдельные источники питания и общие цепи. На эквивалентной выходной цепи представлен один канал группы.



Основные характеристики D0-06AA	
Потребление электроэнергии	~100 - 240В, макс. 40ВА
Порт связи 1: 9600 бод (фиксированно), 8 бит данных, 1 стоповый бит, проверка на нечетность	K-Sequence (Slave), DirectNET (Slave), MODBUS (Slave)
Порт связи 2: 9600 бод (по умолчанию), 8 бит данных, 1 стоповый бит, проверка на нечетность	K-Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence / на принтер, ASCII ввод/вывод
Тип программного кабеля, для порта1	D2--DSCBL
Рабочая температура	от 0 до 55°C
Температура хранения	от -20 до 70°C
Относительная влажность	от 5 до 95% (без конденсации)
Воздушная среда	Без агрессивных газов
Вибрация	MIL STD 810C 514.2
Ударная нагрузка	MIL STD 810C 516.2
Помехоустойчивость	NEMA ICS3-304
Тип клеммной колодки	Съемный
Сечение провода	Один провод 1.3мм^2 или 2 провода 0.78мм^2, мин. 0.2 мм^2

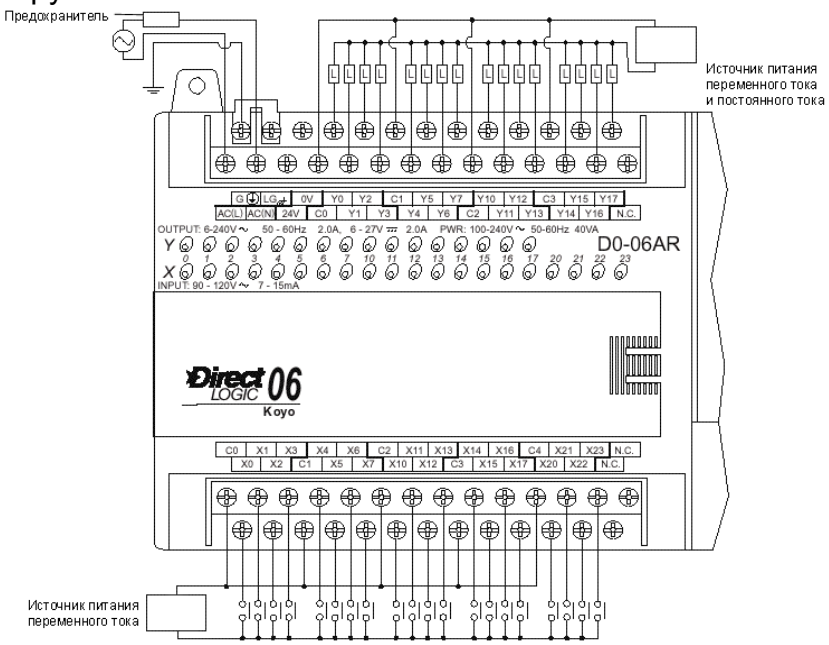
Характеристики входов переменного тока	
Диапазон входного напряжения (мин.- макс.)	~80 - 132В, 47 - 63 Гц
Рабочий диапазон напряжения	~90 - 120В, 47 - 63 Гц
Потребляемый ток	8 мА при ~100В, 50 Гц 10 мА при ~100В, 60 Гц
Макс. потребляемый ток	12 мА при ~132В, 50 Гц 15 мА при ~132В, 60 Гц
Входное сопротивление	14КОм при 50 Гц, 12КОм при 60 Гц
Сила тока/напряжение ВКЛ	>6 мА при ~75В
Сила тока/напряжение ВЫКЛ	<2 мА при ~20В
Время срабатывания ВЫКЛ-ВКЛ	< 40 мс
Время срабатывания ВКЛ-ВЫКЛ	**<** 40 мс
Срабатывание индикаторов состояния	От логических цепей
Общие	На 4 канала 1 общий, 5 групп (изолированые)

Характеристики выходов переменного тока	
Диапазон выходного напряжения (мин. - макс.)	~15 - 264В, 47 -63 Гц
Рабочий диапазон напряжения	~17 - 240В, 47 -63 Гц
Падение напряжения во вкл. состоянии	~1.5В при >50мА; ~4В при <50мА
Максимальный ток	0.5 А/точка, 1.5А / общий
Максимальный ток утечки	<4 мА при ~264 В
Максимальный пусковой ток	10 А за 10 мс
Минимальная нагрузка	10 мА
Время срабатывания ВЫКЛ-ВКЛ	1 мс
Время срабатывания ВКЛ-ВЫКЛ	1 мс+1/2 периода
Срабатывание индикаторов состояния	От логических цепей
Общие	На 4 канала 1 общий, 4 группы (изолированые)
Предохранители	Нет (рекомендуются внешние)

Схема соединений входов/выходов D0- 06AR

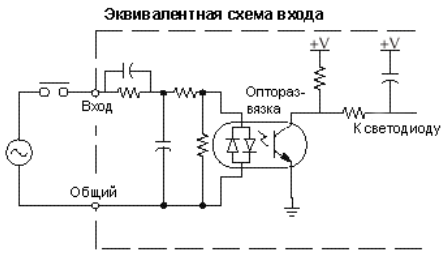
Микроконтроллер D0-06AR имеет двадцать входов переменного тока и шестнадцать релейных выходов. На схеме приведен пример типичного подключения внешних устройств. Как видно на схеме, для подвода внешнего источника питания используются четыре клеммы.

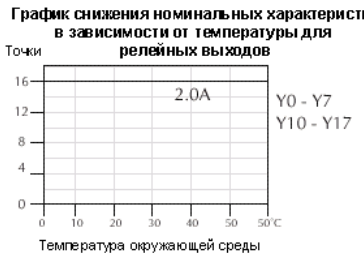
Двадцать входов переменного тока используют клеммы, расположенные в нижней части клеммника. Входы объединены в пять групп по четыре канала. Каждой группе отводится отдельный общий провод. В приведенном ниже примере все общие цепи соединены в одну, но можно использовать отдельные источники питания и общие цепи. На эквивалентной входной цепи представлен один канал группы.

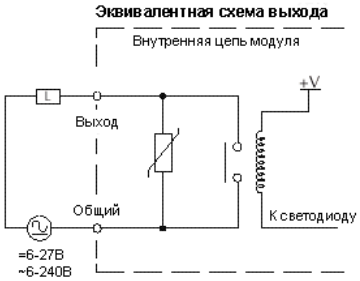


Стандартный ресурс реле (1000 операций)

Напряжение и вид нагрузки	Ток нагрузки	
	1А	2А
24В Активная	500К	250К
24В Активная	100К	50К
~110В Активная	500К	250К
~110В Активная	200К	100К
~220В Активная	350К	200К
~220В Активная	100К	50К







Шестнадцать релейных выходов используют клеммы, расположенные в верхней части клеммника. Выходы сгруппированы в четыре группы по четыре реле с нормально разомкнутыми контактами. Каждой группе отводится одна общая клемма. В приведенном выше примере все общие провода соединены в одной точке, но можно использовать отдельные источники питания и общие цепи. На эквивалентной схеме выходной цепи представлен один канал группы. Релейные контакты могут использоваться в цепях постоянного или переменного тока.

Основные характеристики D0-06AR	
Потребление электроэнергии	~100 - 240В, макс. 40ВА
Порт связи 1: 9600 бод (фиксировано): 8 бит данных, 1 стоповый бит, проверка на нечетность	K-Sequence (Slave), DirectNET (Slave), MODBUS (Slave)
Порт связи 2: 9600 бод (по умолчанию): 8 бит данных, 1 стоповый бит, проверка на нечетность	K-Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence / на принтер, ASCII ввод/вывод
Тип программного кабеля	D2--DSCBL
Рабочая температура	от 0 до 55°C
Температура хранения	от -20 до 70°C
Относительная влажность	от 5 до 95% (без конденсации)
Воздушная среда	Без агрессивных газов
Вибрация	MIL STD 810C 514.2
Ударная нагрузка	MIL STD 810C 516.2
Помехоустойчивость	NEMA ICS3-304
Тип клеммной колодки	Съемный
Сечение провода	Один провод 1.3мм^2 или 2 провода 0.78мм^2, мин. 0.2 мм^2

Характеристики входов переменного тока X0-X23	
Диапазон входного напряжения (мин.- макс.)	~80 - 132В, 47 - 63 Гц
Рабочий диапазон напряжения	~90 - 120В, 47 - 63 Гц
Потребляемый ток	8 мА при ~100В, 50 Гц 10 мА при ~100В, 60 Гц
Макс. потребляемый ток	12 мА при ~132В, 50 Гц 15 мА при ~132В, 60 Гц
Входное сопротивление	14КОм при 50 Гц, 12КОм при 60 Гц
Сила тока/напряжение ВКЛ	>6 мА при ~75В
Сила тока/напряжение ВЫКЛ	<2 мА при ~20В
Время срабатывания ВЫКЛ-ВКЛ	< 40 мс
Время срабатывания ВКЛ-ВЫКЛ	**<** 40 мс
Срабатывание индикаторов состояния	От логических цепей
Общие	На 4 канала 1 общий, 5 групп (изолирован ые)

Характеристики релейных выходов Y0-Y17	
Диапазон выходного напряжения	(мин. - макс.) ~5 - 264В (47 -63 Гц), =5-30В
Рабочий диапазон напряжения	~6 - 240В (47 -63 Гц), =6-27В
Выходной ток	2А/точка, 6А / общий
Максимальный ток утечки	0.1 мА при ~264 В
Минимальная рекомендуемая нагрузка	5 мА при =5 В
Время срабатывания ВЫКЛ-ВКЛ	<15 мс
Время срабатывания ВКЛ-ВЫКЛ	<10 мс
Срабатывание индикаторов состояния	От логических цепей
Общие	На 4 канала 1 общий, 4 группы (изолированые)
Предохранители	Нет (рекомендуются внешние)

Схема соединений входов/выходов D0- 06DA

Микроконтроллер D0-06DA имеет двадцать входов постоянного тока и шестнадцать выходов переменного тока. На схеме приведен пример типичного подключения внешних устройств. Как видно на схеме, для подвода внешнего источника питания используются четыре клеммы.

Входы объединены в пять групп по четыре канала. Каждая группа имеет отдельный общий провод и может быть подключена как приемник или источник. В приведенном ниже примере все общие цепи соединены в одну, но можно использовать отдельные источники питания и общие цепи. Эквивалентная схема для стандартных входов представлена ниже и быстродействующие входы показаны слева.

Выходы объединены в четыре группы по четыре переключателя триак. Каждой группе отводится одна общая цепь. В приведенном ниже примере все общие цепи соединены в одну, но можно использовать отдельные источники питания и общие цепи. На эквивалентной выходной цепи представлен один канал группы.

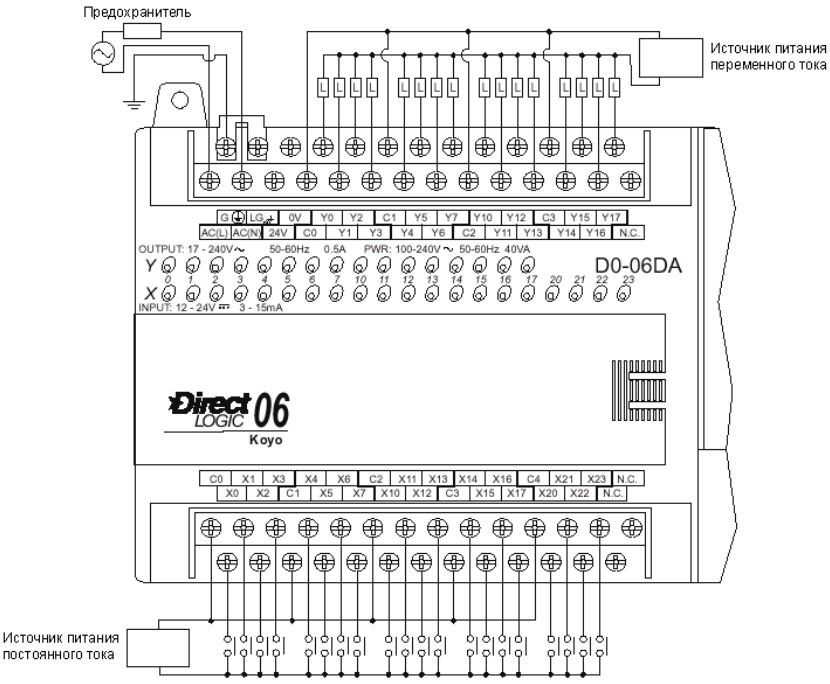
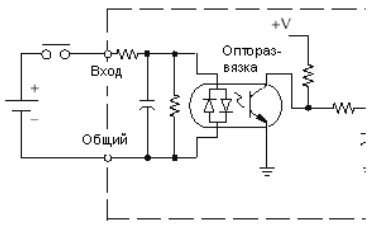
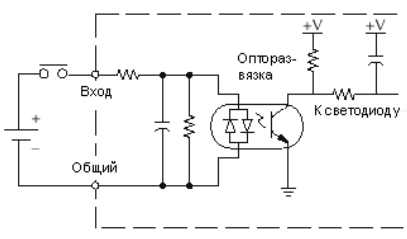


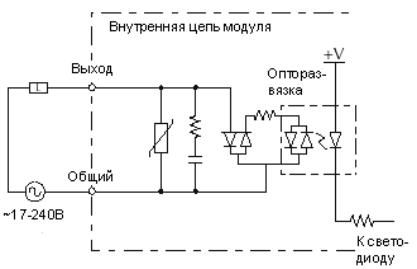
График снижения номинальных характеристик в зависимости от температуры для выходов переменного тока



Быстродействующие входы (X0-X3)



Стандартные входы (X4-X23)



Эквивалентная схема выхода

Основные характеристики D0-06DA	
Потребление электроэнергии	~100 - 240В, макс. 40ВА
Порт связи 1: 9600 бод (фиксировано), 8 бит данных, 1 стоповый бит, проверка на нечетность	K-Sequence (Slave), DirectNET (Slave), MODBUS (Slave)
Порт связи 2: 9600 бод (по умолчанию), 8 бит данных, 1 стоповый бит, проверка на нечетность	K-Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence / на принтер, ASCII ввод/вывод
Тип программного кабеля	D2--DSCBL
Рабочая температура	от 0 до 55°C
Температура хранения	от -20 до 70°C
Относительная влажность	от 5 до 95% (без конденсации)
Воздушная среда	Без агрессивных газов
Вибрация	MIL STD 810C 514.2
Ударная нагрузка	MIL STD 810C 516.2
Помехоустойчивость	NEMA ICS3-304
Тип клеммной колодки	Съемный
Сечение провода	Один провод 1.3мм^2 или 2 провода 0.78мм^2, мин. 0.2 мм^2

Характеристики входов постоянного тока		
Параметр	**Быстродействующие входы, X0 – X3**	**Стандартные входы постоянного тока X4 –X23**
Диапазон напряжения	=10.8 - 26.4 В	=10.8 - 26.4 В
Рабочий диапазон напряжения	=12 -24 В	=12 -24 В
Максимальное напряжение	=30В (макс. частота - 7 кГц)	=30В
Мин. длительность импульса	70 мкс	Нет
Напряжение Включения	>10В постоянного тока	>10В постоянного тока
Напряжение Выключения	<2.0 В постоянного тока	<2.0 В постоянного тока
Входное сопротивление	1.8 кОм при =12-24В	2.8кОм при 12-24В пост. тока
Мин. ток Включения	>5 мА	>4 мА
Макс. ток Выключения	< 0.5 мА	< 0.5 мА
Время срабатывания ВЫКЛ-ВКЛ	<70 мкс	2 - 8 мс, обычно 4 мс
Время срабатывания ВКЛ-ВЫКЛ	<70 мкс	2 - 8 мс, обычно 4 мс
Срабатывание индикаторов	От логических цепей	От логических цепей
Общие	На 4 канала 1 общий, 5 групп (изолированны)	

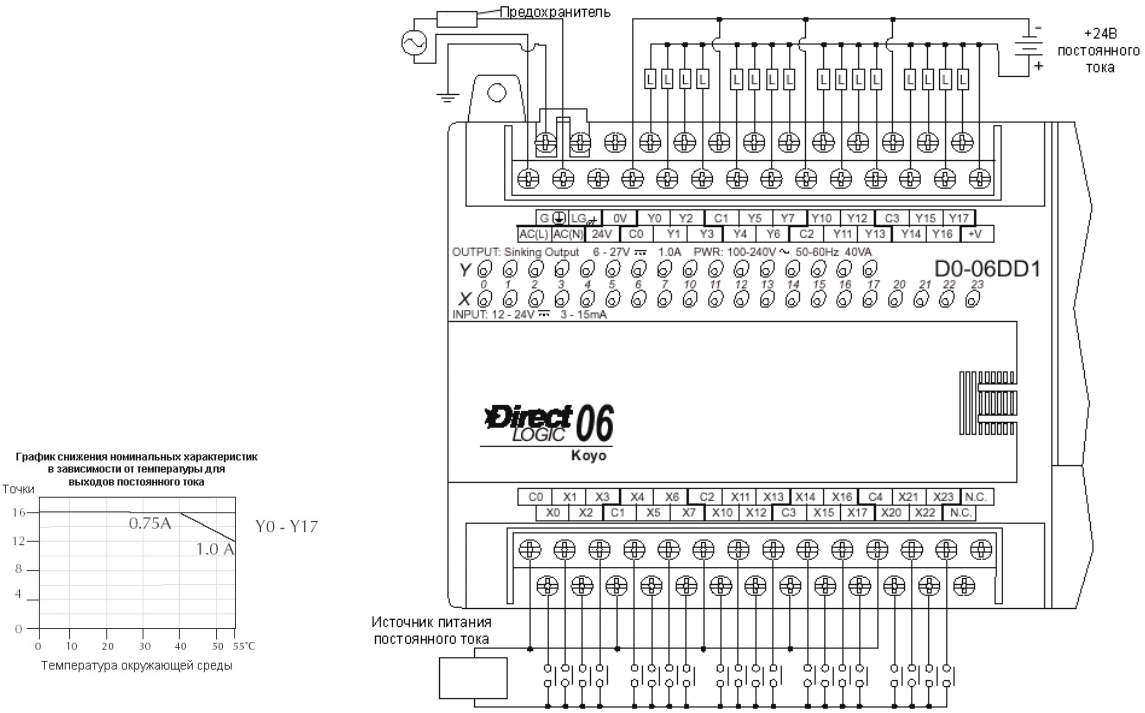
Характеристики выходов переменного тока	
Диапазон выходного напряжения (мин.- макс.)	~15 - 264В, 47 - 63 Гц
Рабочее напряжение	~17 - 240В, 47 - 63 Гц
Падение напряжения во вкл. состоянии	~1.5В при >50мА; ~4В при <50мА
Максимальный ток	0.5 А/канал, 1.5А / общий
Максимальный ток утечки	<4 мА при ~264В, 60Гц
Максимальный пусковой ток	10 А за 10 мс
Минимальная нагрузка	10 мА
Время срабатывания ВЫКЛ-ВКЛ	1 мс
Время срабатывания ВКЛ-ВЫКЛ	1 мс+1/2 периода
Срабатывание индикаторов состояния	От логических цепей
Общие	На 4 канала 1 общий, 4 группы (изолированный)
Предохранители	Нет (рекомендуются внешние)

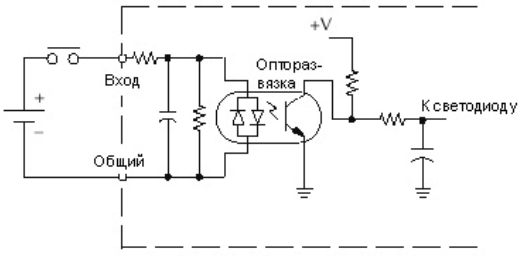
Схема соединений входов/выходов D0- 06DD1

Микроконтроллер D0-06DD1 имеет двадцать входов (потребитель/источник) и шестнадцать выходов (потребитель) постоянного тока. На схеме приведен пример типичного подключения внешних устройств. Как видно на схеме, для подвода внешнего источника питания используются четыре клеммы.

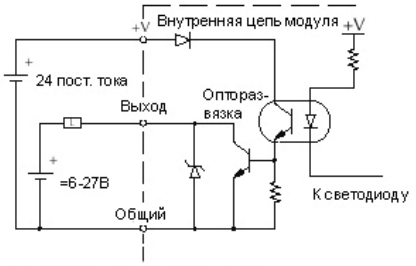
Входы объединены в пять групп по четыре канала. Каждой группе отводится отдельный общий провод и может быть подключена как приемник или источник. В приведенном ниже примере все общие цепи соединены в одну, но можно использовать отдельные источники питания и общие цепи.

Все выходы используют один и тот же общий провод. Обратите внимание на потребность во внешнем источнике

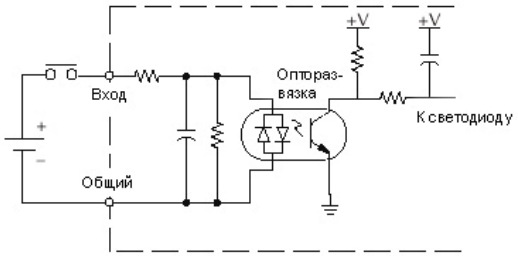




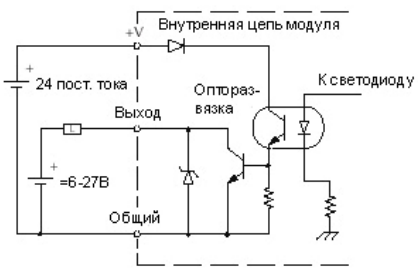
Быстродействующие входы (X0-X3)



Импульсные выходы (Y0-Y1)



Стандартные входы постоянного тока (X4-X23)



Стандартные выходы постоянного тока (Y2-Y17)

Основные характеристики D0-06DD1	
Потребление электроэнергии	~100 - 240В, макс. 40ВА
Порт связи 1: 9600 бод (фиксировано), 8 бит данных, 1 стоповый бит, проверка на нечетность	K-Sequence (Slave), DirectNET (Slave), MODBUS (Slave)
Порт связи 2: 9600 бод (по умолчанию), 8 бит данных, 1 стоповый бит, проверка на нечетность	K-Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence / на принтер, ASCII ввод/вывод
Тип программного кабеля	D2--DSCBL
Рабочая температура	от 0 до 55°C
Температура хранения	от -20 до 70°C
Относительная влажность	от 5 до 95% (без конденсации)
Воздушная среда	Без агрессивных газов
Вибрация	MIL STD 810C 514.2
Ударная нагрузка	MIL STD 810C 516.2
Помехоустойчивость	NEMA ICS3-304
Тип клеммной колодки	Съемный
Сечение провода	Один провод 1.3мм^2 или 2 провода 0.78мм^2, мин. 0.2 мм^2

Характеристики входов постоянного тока		
Параметр	**Быстродействующие входы, X0 – X3**	**Стандартные входы постоянного тока X4 –X23**
Диапазон напряжения (мин.-макс.)	=10.8 - 26.4 В	=10.8 - 26.4 В
Рабочий диапазон напряжения	=12 -24 В	=12 -24 В
Максимальное напряжение	=30В (макс. частота - 7 кГц)	=30В
Мин. длительность импульса	100 мкс	Нет
Напряжение Включения	>10В постоянного тока	>10В постоянного тока
Напряжение Выключения	<2.0 В постоянного тока	<2.0 В постоянного тока
Макс. потребляемый ток	6мА при =12В 13мА при =24В	4мА при =12В 8.5мА при =24В
Входное сопротивление	1.8 кОм при =12-24В	2.8кОм при =12-24В
Мин. ток Включения	>5 мА	>4 мА
Макс. ток Выключения	< 0.5 мА	< 0.5 мА
Время срабатывания ВЫКЛ-ВКЛ	<70 мкс	2 - 8 мс, обычно 4 мс
Время срабатывания ВКЛ-ВЫКЛ	<70 мкс	2 - 8 мс, обычно 4 мс
Срабатывание индикаторов	От логических цепей	От логических цепей
Общие	На 4 канала 1 общий, 5 групп (изолированые)	

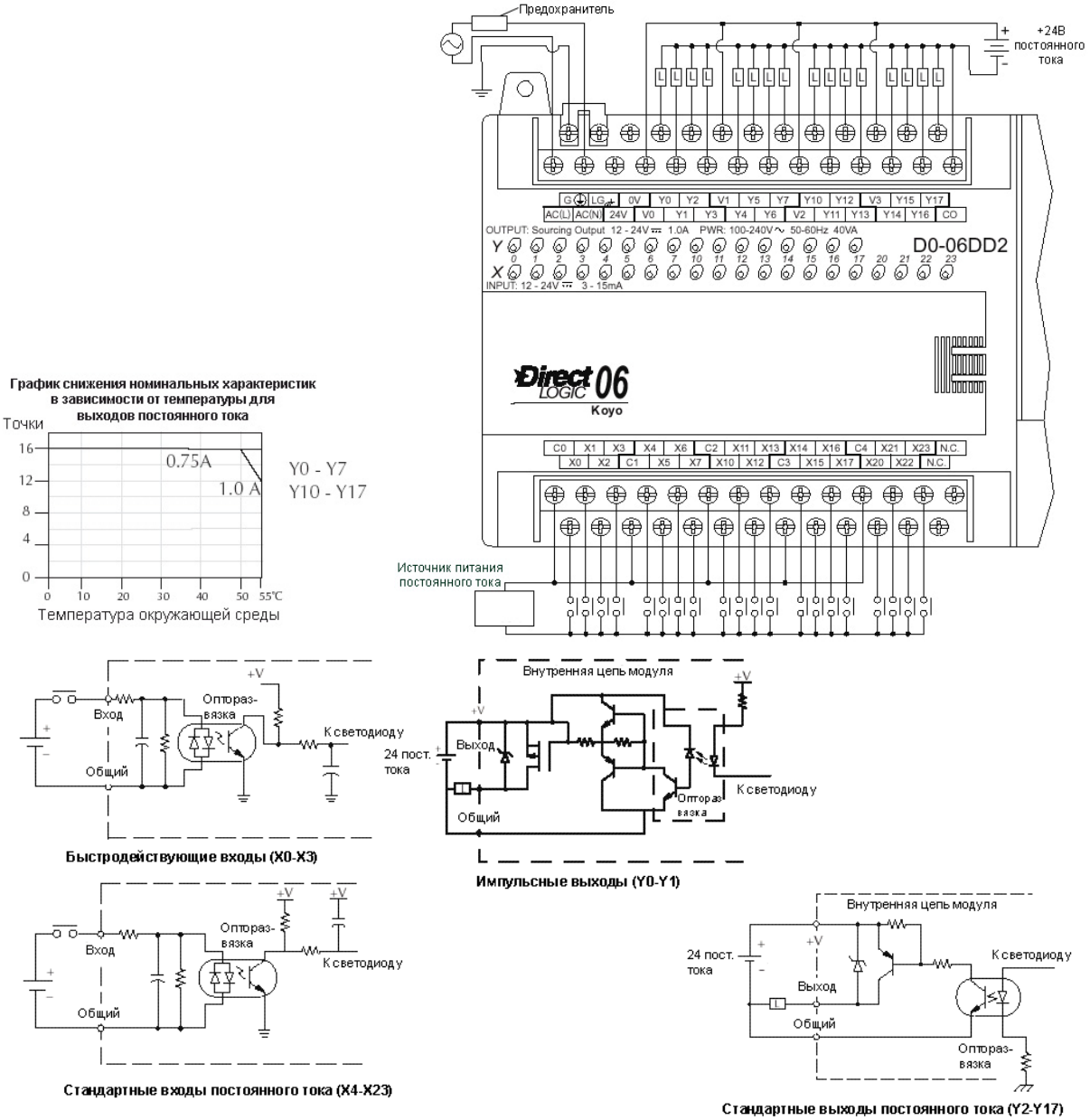
Характеристики выходов постоянного тока		
Параметр	**Импульсные выходы, Y0 – Y1**	**Стандартные выходы Y2 –Y17**
Диапазон выходного напряжения (мин.- макс.)	=5-30В	=5-30В
Рабочее напряжение	=6-27В	=6-27В
Максимальное напряжение	<50В постоянного тока (макс. частота - 10кГц)	<50В
Падение напряжения во вкл. состоянии	=0.3В при 1А	=0.3В при 1А
Максимальный ток (резистивная)	0.5 А/точка, 1А / точка как стандартная точка	1А / точка
Максимальный ток утечки	15мкА при =30В	15мкА при =30В
Максимальный пусковой ток	2 А за 100 мс	2 А за 100 мс
Требуется внешний источник питания постоянного тока	=20-28В, макс.150мА	=20-28В, макс.280мА (Вспом. питание =24В клемма V+ (выходы потребитель)
Время срабатывания ВЫКЛ-ВКЛ	<10мкс	<10мкс
Время срабатывания ВКЛ-ВЫКЛ	<20мкс	<60мкс
Срабатывание индикаторов состояния	От логических цепей	От логических цепей
Общие	На 4 канала 1 общий, 4 группы (неизолированые)	
Предохранители	Нет (рекомендуются внешние)	

Схема соединений входов/выходов D0- 06DD2

Микроконтроллер D0-06DD2 имеет двадцать входов постоянного тока (потребитель/источник) и шестнадцать выходов постоянного тока (источник). На схеме приведен пример типичного подключения внешних устройств. Как видно на схеме, для подвода внешнего источника питания используются четыре клеммы.

Входы объединены в пять групп по четыре канала. Каждой группе отводится отдельный общий провод и может быть подключена как приемник или источник. В приведенном ниже примере все общие цепи соединены в одну, но можно использовать отдельные источники питания и общие цепи.

Все выходы работают с одним и тем же общим каналом. Обратите внимание на потребность во внешнем источнике



Основные характеристики D0-06DD2	
Потребление электроэнергии	~100 - 240В, макс. 40ВА
Порт связи 1: 9600 бод (фиксировано), 8 бит данных, 1 стоповый бит, проверка на нечетность	Протокол – автоматический выбор(auto-select): K-Sequence (Slave), DirectNET (Slave), MODBUS (Slave)
Порт связи 2: 9600 бод (по умолчанию), 8 бит данных, 1 стоповый бит, проверка на нечетность	K-Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence / на принтер, ASCII ввод/вывод
Тип программного кабеля	D2--DSCBL
Рабочая температура	от 0 до 55°C
Температура хранения	от -20 до 70°C
Относительная влажность	от 5 до 95% (без конденсации)
Воздушная среда	Без агрессивных газов
Вибрация	MIL STD 810C 514.2
Ударная нагрузка	MIL STD 810C 516.2
Помехоустойчивость	NEMA ICS3-304
Тип клеммной колодки	Съемный
Сечение провода	Один провод 1.3мм^2 или 2 провода 0.78мм^2, мин. 0.2 мм^2

Характеристики входов постоянного тока		
Параметр	**Быстродействующие входы, X0 – X3**	**Стандартные входы постоянного тока X4 –X23**
Диапазон напряжения (мин.-макс.)	=10.8 - 26.4 В	=10.8 - 26.4 В
Рабочий диапазон напряжения	=12 -24 В	=12 -24 В
Максимальное напряжение	=30В (макс. частота - 7 кГц)	=30В
Мин. длительность импульса	70 мкс	Нет
Напряжение Включения	>10В постоянного тока	>10В постоянного тока
Напряжение Выключения	<2.0 В постоянного тока	<2.0 В постоянного тока
Макс. потребляемый ток	6мА при =12В 13мА при =24В	4мА при =12В 8.5мА при =24В
Входное сопротивление	1.8 кОм при =12-24В	2.8кОм при =12-24В
Мин. ток Включения	>5 мА	>4 мА
Макс. ток Выключения	< 0.5 мА	< 0.5 мА
Время срабатывания ВЫКЛ-ВКЛ	<70 мкс	2 - 8 мс, обычно 4 мс
Время срабатывания ВКЛ-ВЫКЛ	<70 мкс	2 - 8 мс, обычно 4 мс
Срабатывание индикаторов	От логических цепей	От логических цепей
Общие	На 4 канала 1 общий, 5 групп (изолированны)	

Характеристики выходов постоянного тока		
Параметр	**Импульсные выходы, Y0 – Y1**	**Стандартные выходы Y2 –Y17**
Диапазон выходного напряжения (мин.-макс.)	=10.8 - 26.4 В	=10.8 - 26.4 В
Рабочее напряжение	=12-24В	=12-24В
Максимальное напряжение	<50В постоянного тока (макс. частота - 10кГц)	<50В
Падение напряжения во вкл. состоянии	=0.5В при 1А	=1.2В при 1А
Максимальный ток (резистивная)	0.5 А/канал, 1А / канал, как стандартный выход	1А / канал
Максимальный ток утечки	15мкА при =30В	15мкА при =30В
Максимальный пусковой ток	2 А за 100 мс	2 А за 100 мс
Требуется внешний источник питания постоянного тока	Нет	Нет
Время срабатывания ВЫКЛ-ВКЛ	<10мкс	<10мкс
Время срабатывания ВКЛ-ВЫКЛ	<20мкс	<0.5мкс
Срабатывание индикаторов состояния	От логических цепей	От логических цепей
Общие	На 4 канала 1 общий, 4 группы (неизолированые)	
Предохранители	Нет (рекомендуются внешние)	

Схема соединений входов/выходов D0- 06DR

Микроконтроллер D0-06DR имеет двадцать входов постоянного тока и шестнадцать релейных выходов. На схеме приведен пример типичного подключения внешних устройств. Как видно на схеме, для подвода внешнего источника питания используются четыре клеммы.

Входы объединены в пять групп по четыре канала. Каждой группе отводится отдельный общий провод и может быть подключена как приемник или источник. В приведенном ниже примере все общие цепи соединены в одну, но можно использовать отдельные источники питания и общие цепи. Стандартные входы на эквивалентной схеме представлены внизу, быстродействующие входы показаны слева.

Выходы сгруппированы в четыре группы по четыре реле с нормально разомкнутыми контактами. Каждой группе отводится одна общая клемма. В приведенном примере все общие провода соединены в одной точке, но можно использовать отдельные источники питания и общие цепи. На эквивалентной схеме выходной цепи представлен один канал группы. Релейные контакты могут использоваться в цепях постоянного или переменного тока.

Стандартный ресурс реле (1000 операций)

Напряжение и вид нагрузки	Ток нагрузки	
	1A	2A
24В Активная	500К	250К
24В Активная	100К	50К
~110В Активная	500К	250К
~110В Активная	200К	100К
~220В Активная	350К	200К
~220В Активная	100К	50К

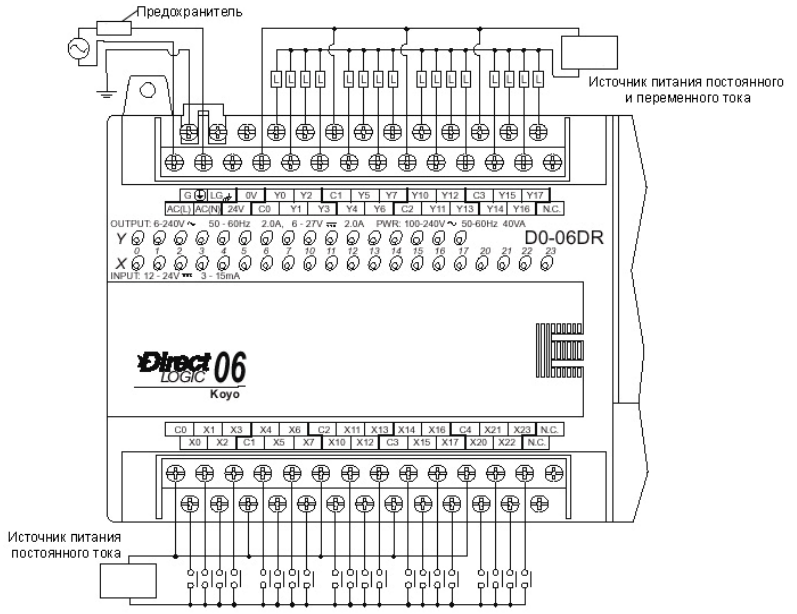
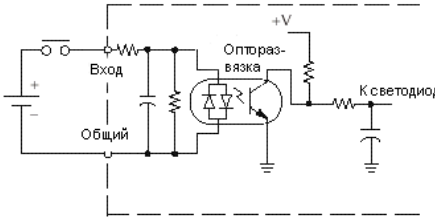
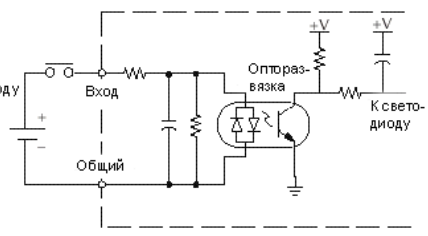


График снижения номинальных характеристик в зависимости от температуры для релейных выходов

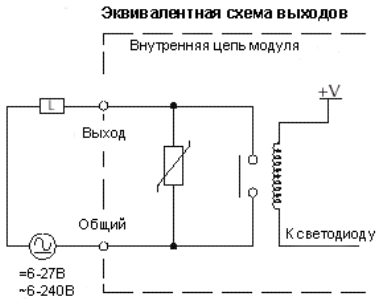
Эквивалентная схема
Быстродействующие входы (X0-X3)



Эквивалентная схема
Стандартные входы (X4-X23)



Эквивалентная схема выходов



Основные характеристики D0-06DR	
Потребление электроэнергии	~100 - 240В, макс. 40ВА
Порт связи 1: 9600 бод (фиксировано), 8 бит данных, 1 стоповый бит, проверка на нечетность	K-Sequence (Slave), DirectNET (Slave), MODBUS (Slave)
Порт связи 2: 9600 бод (по умолчанию), 8 бит данных, 1 стоповый бит, проверка на нечетность	K-Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence / на принтер, ASCII ввод/вывод
Тип программного кабеля	D2--DSCBL
Рабочая температура	от 0 до 55°C
Температура хранения	от -20 до 70°C
Относительная влажность	от 5 до 95% (без конденсации)
Воздушная среда	Без агрессивных газов
Вибрация	MIL STD 810C 514.2
Ударная нагрузка	MIL STD 810C 516.2
Помехоустойчивость	NEMA ICS3-304
Тип клеммной колодки	Съемный
Сечение провода	Один провод 1.3мм^2 или 2 провода 0.78мм^2, мин. 0.2 мм^2

Характеристики входов постоянного тока		
Параметр	**Быстродействующие входы, X0 – X3**	**Стандартные входы постоянного тока X4 –X23**
Диапазон напряжения (мин.-макс.)	=10.8 - 26.4 В	=10.8 - 26.4 В
Рабочий диапазон напряжения	=12 -24 В	=12 -24 В
Максимальное напряжение	=30В (макс. частота - 7 кГц)	=30В
Мин. длительность импульса	70 мкс	Нет
Напряжение Включения	>10В постоянного тока	>10В постоянного тока
Напряжение Выключения	<2.0 В постоянного тока	<2.0 В постоянного тока
Макс. потребляемый ток	6мА при =12В 13мА при =24В	4мА при =12В 8.5мА при =24В
Входное сопротивление	1.8 кОм при =12-24В	2.8кОм при =12-24В
Мин. ток Включения	>5 мА	>4 мА
Макс. ток Выключения	< 0.5 мА	< 0.5 мА
Время срабатывания ВЫКЛ-ВКЛ	<70 мкс	2-8мс, типично 4мс
Время срабатывания ВКЛ-ВЫКЛ	<70 мкс	2-8мс, типично 4мс
Срабатывание индикаторов состояния	От логических цепей	От логических цепей
Общие	На 4 канала 1 общий провод, 5 групп (изолированны)	

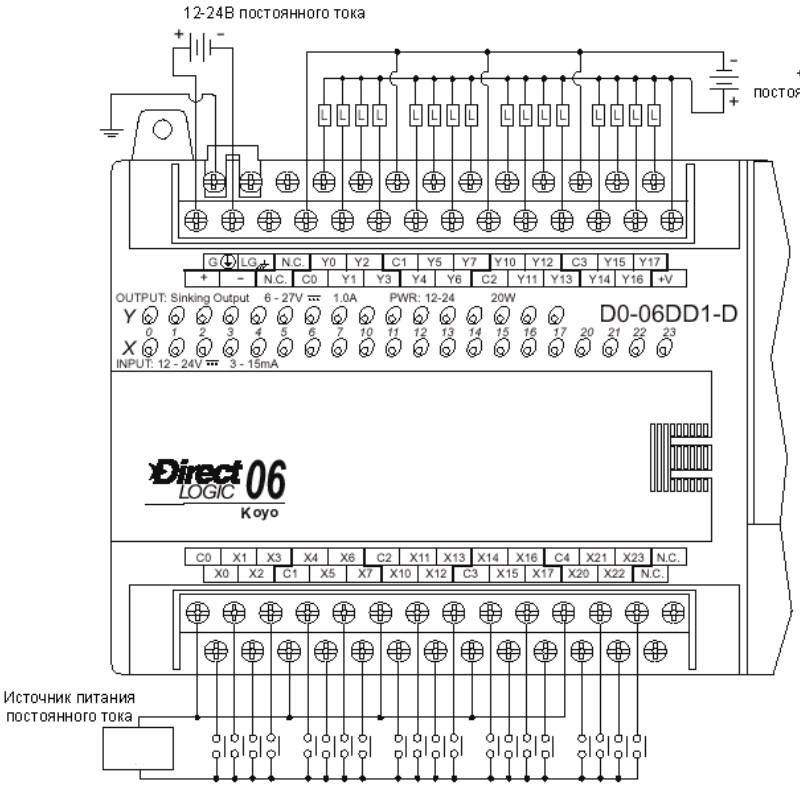
Характеристики релейных выходов	
Диапазон выходного напряжения (мин. – макс.)	~5 - 264В (47 -63 Гц), =5-30В
Рабочий диапазон напряжения	~6 - 240В (47 -63 Гц), =6-27В
Выходной ток	2А/канал, 6А / общий
Максимальное напряжение	~264В, =30В
Максимальный ток утечки	0.1 мА при ~264 В
Минимальная рекомендуемая нагрузка	5 мА
Время срабатывания ВЫКЛ-ВКЛ	<15 мс
Время срабатывания ВКЛ-ВЫКЛ	<10 мс
Срабатывание индикаторов состояния	От логических цепей
Общие	На 4 канала 1 общий, 4 группы (изолированые)
Предохранители	Нет (рекомендуются внешние)

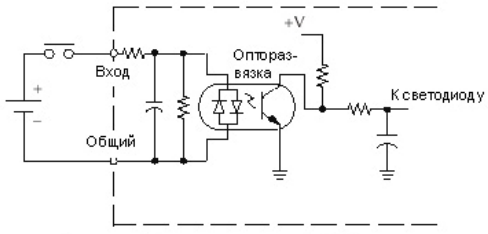
Схема соединений входов/выходов D0- 06DD1-D

Микроконтроллер D0-06DD1-D имеет двадцать входов постоянного тока и шестнадцать выходов постоянного тока, потребитель. На схеме приведен пример типичного подключения внешних устройств. Как видно на схеме, для подвода внешнего источника питания используются четыре клеммы.

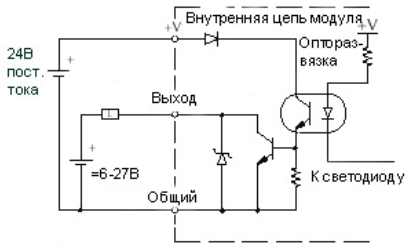
Входы объединены в пять групп по четыре канала. Каждой группе отводится отдельный общий провод и может быть подключена как приемник или источник. В приведенном ниже примере все общие цепи соединены в одну, но можно использовать отдельные источники питания и общие цепи.

Все выходы используют один и тот же общий провод. Обратите внимание на потребность во внешнем источнике.

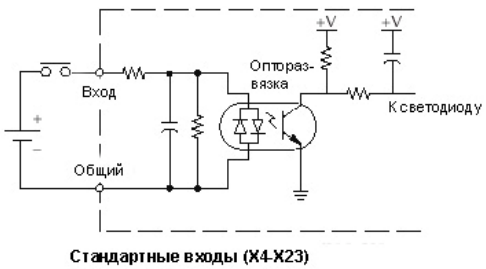




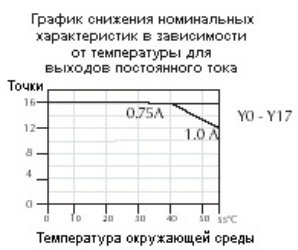
Быстродействующие входы (X0-X3)

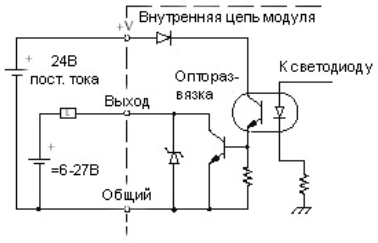


Импульсные выходы (Y0-Y1)



Стандартные входы (X4-X23)





Стандартные выходы постоянного тока (Y2-Y17)

Основные характеристики D0-06DD1-D	
Потребление электроэнергии	=12 - 24В, макс. 20Вт
Порт связи 1: 9600 бод (фиксировано), 8 бит данных, 1 стоповый бит, проверка на нечетность	K-Sequence (Slave), DirectNET (Slave), MODBUS (Slave)
Порт связи 2: 9600 бод (по умолчанию), 8 бит данных, 1 стоповый бит, проверка на нечетность	K-Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence / на принтер, ASCII ввод/вывод
Тип программного кабеля	D2--DSCBL
Рабочая температура	от 0 до 55°C
Температура хранения	от -20 до 70°C
Относительная влажность	от 5 до 95% (без конденсации)
Воздушная среда	Без агрессивных газов
Вибрация	MIL STD 810C 514.2
Ударная нагрузка	MIL STD 810C 516.2
Помехоустойчивость	NEMA ICS3-304
Тип клеммной колодки	Съемный
Сечение провода	Один провод 1.3мм^2 или 2 провода 0.78мм^2, мин. 0.2 мм^2

Характеристики входов постоянного тока		
Параметр	**Быстродействующие входы, X0 – X3**	**Стандартные входы постоянного тока X4 –X23**
Диапазон напряжения (мин.-макс.)	=10.8 - 26.4 В	=10.8 - 26.4 В
Рабочий диапазон напряжения	=12 -24 В	=12 -24 В
Максимальное напряжение	=30В (макс. частота - 7 кГц)	=30В
Мин. длительность импульса	70 мкс	Нет
Напряжение Включения	>10В постоянного тока	>10В постоянного тока
Напряжение Выключения	<2.0 В постоянного тока	<2.0 В постоянного тока
Макс. потребляемый ток	6мА при =12В 13мА при =24В	4мА при =12В 8.5мА при =24В
Входное сопротивление	1.8 КОм при =12-24В	2.8КОм при =12-24В
Мин. ток Включения	>5 мА	>4 мА
Макс. ток Выключения	< 0.5 мА	< 0.5 мА
Время срабатывания ВЫКЛ-ВКЛ	<70 мкс	2 - 8 мс, обычно 4 мс
Время срабатывания ВКЛ-ВЫКЛ	<70 мкс	2 - 8 мс, обычно 4 мс
Срабатывание индикаторов	От логических цепей	От логических цепей
Общие	На 4 канала 1 общий, 5 групп (изолированые)	

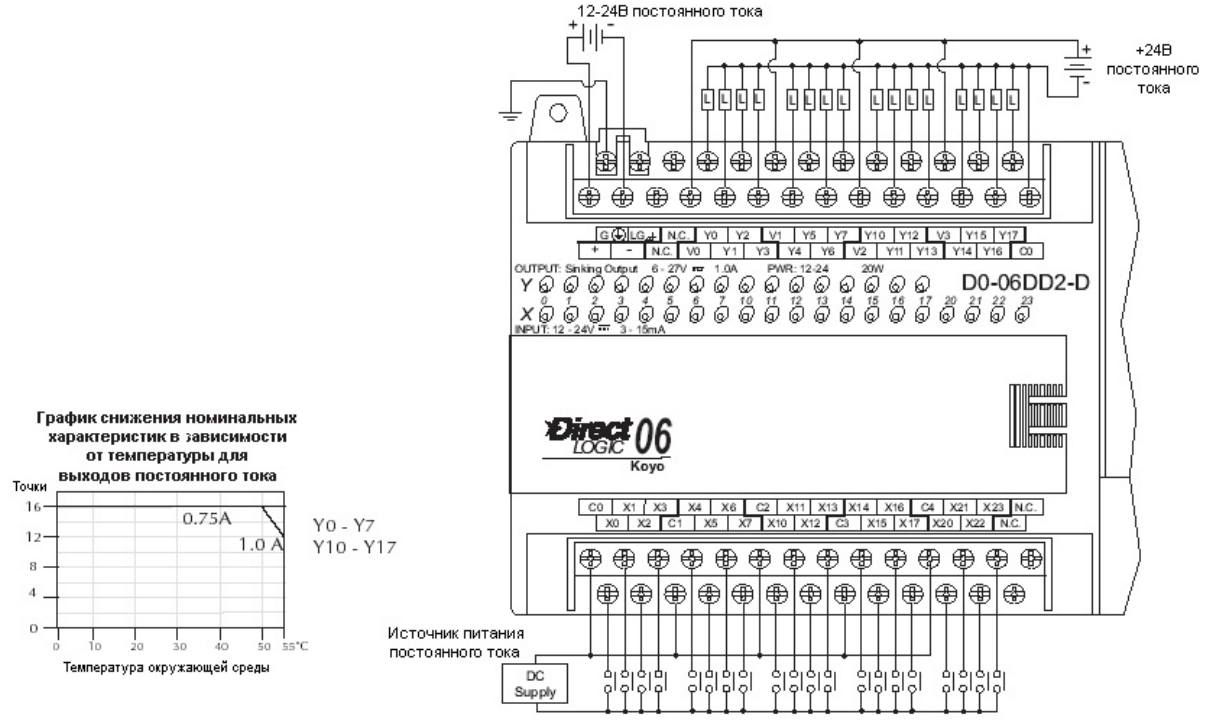
Характеристики выходов постоянного тока		
Параметр	**Импульсные выходы, Y0 – Y1**	**Стандартные выходы Y2 –Y17**
Диапазон выходного напряжения (мин.-макс.)	=5 - 30 В	=5 - 30 В
Рабочее напряжение	=6 - 27 В	=6 - 27 В
Максимальное напряжение	<50В постоянного тока (макс. частота – 10КГц)	<50В постоянного тока
Падение напряжения во вкл. состоянии	=0.3В при 1А	=0.3В при 1А
Максимальный ток (резистивная)	0.5 А/точка, 1А / канал, как стандартный выход	1А / канал
Максимальный ток утечки	15мкА при =30В	15мкА при =30В
Максимальный пусковой ток	2 А за 100 мс	2 А за 100 мс
Требуется внешний источник питания постоянного тока	=20-28В , макс. 150 мА	=20-28В , макс. 150 мА
Время срабатывания ВЫКЛ-ВКЛ	<10мкс	<10мкс
Время срабатывания ВКЛ-ВЫКЛ	<20мкс	<60мкс
Срабатывание индикаторов состояния	От логических цепей	От логических цепей
Общие	На 4 канала 1 общий, 4 группы (неизолированые)	
Предохранители	Нет (рекомендуются внешние)	

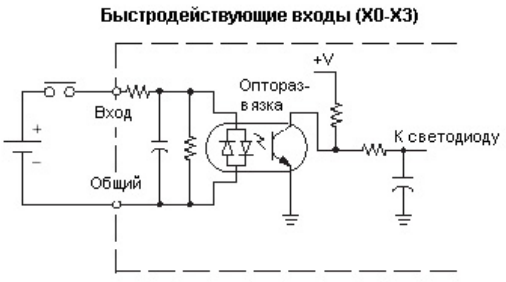
Схема соединений входов/выходов D0- 06DD2-D

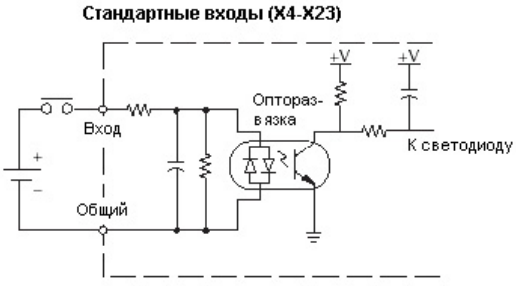
Микроконтроллер D0-06DD2-D имеет двадцать входов постоянного тока и шестнадцать выходов постоянного тока (источник). На схеме приведен пример типичного подключения внешних устройств. Как видно на схеме, для подвода внешнего источника питания используются четыре клеммы.

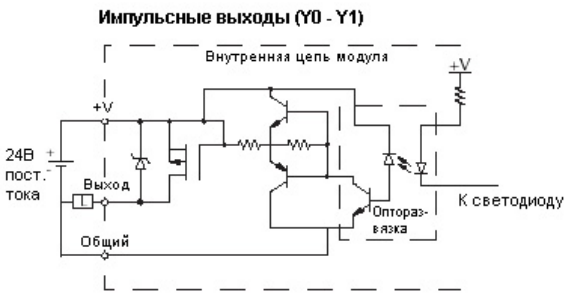
Входы объединены в пять групп по четыре канала. Каждой группе отводится отдельный общий провод и может быть подключена как приемник или источник. В приведенном ниже примере все общие цепи соединены в одну, но можно использовать отдельные источники питания и общие цепи.

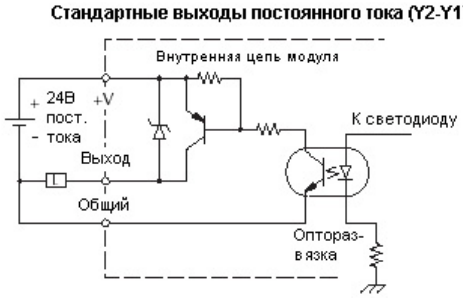
Все выходы используют один и тот же общий провод. Обратите внимание на потребность во внешнем источнике.











Основные характеристики D0-06DD2-D	
Потребление электроэнергии	=12 - 24В, макс. 20Вт
Порт связи 1: 9600 бод (фиксировано), 8 бит данных, 1 стоповый бит, проверка на нечетность	Протокол – автоматический выбор(auto-select): K-Sequence (Slave), DirectNET (Slave), MODBUS (Slave)
Порт связи 2: 9600 бод (по умолчанию), 8 бит данных, 1 стоповый бит, проверка на нечетность	Auto-select: K-Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence / на принтер, ASCII ввод/вывод
Тип программного кабеля	D2--DSCBL
Рабочая температура	от 0 до 55°C
Температура хранения	от -20 до 70°C
Относительная влажность	от 5 до 95% (без конденсации)
Воздушная среда	Без агрессивных газов
Вибрация	MIL STD 810C 514.2
Ударная нагрузка	MIL STD 810C 516.2
Помехоустойчивость	NEMA ICS3-304
Тип клеммной колодки	Съемный
Сечение провода	Один провод 1.3мм^2 или 2 провода 0.78мм^2, мин. 0.2 мм^2

Характеристики входов постоянного тока		
Параметр	**Быстродействующие входы, X0 – X3**	**Стандартные входы постоянного тока X4 –X23**
Диапазон напряжения (мин.-макс.)	=10.8 - 26.4 В	=10.8 - 26.4 В
Рабочий диапазон напряжения	=12 -24 В	=12 -24 В
Максимальное напряжение	=30В (макс. частота - 7 кГц)	=30В
Мин. длительность импульса	70 мкс	Нет
Напряжение Включения	>10В постоянного тока	>10В постоянного тока
Напряжение Выключения	<2.0 В постоянного тока	<2.0 В постоянного тока
Макс. потребляемый ток	15мА при =26.4В	11мА при =26.4В
Входное сопротивление	1.8 КОм при =12-24В	2.8КОм при =12-24В
Мин. ток Включения	5 мА	3 мА
Макс. ток Выключения	0.5 мА	0.5 мА
Время срабатывания ВЫКЛ-ВКЛ	<70 мкс	2 - 8 мс, обычно 4 мс
Время срабатывания ВКЛ-ВЫКЛ	<70 мкс	2 - 8 мс, обычно 4 мс
Срабатывание индикаторов	От логических цепей	От логических цепей
Общие	На 4 канала 1 общий, 5 групп (изолированые)	

Характеристики выходов постоянного тока		
Параметр	**Импульсные выходы, Y0 – Y1**	**Стандартные выходы Y2 –Y17**
Диапазон выходного напряжения (мин.-макс.)	=10.8 – 26.4 В	=10.8 – 26.4 В
Рабочее напряжение	=12 - 24 В	=12 - 24 В
Максимальное напряжение	=30В постоянного тока (макс. частота – 10КГц)	=30В постоянного тока
Падение напряжения во вкл. состоянии	=0.5В при 1А	=1.2В при 1А
Максимальный ток (резистивная)	0.5 А/точка, 1А / канал, как стандартный выход	1А / канал
Максимальный ток утечки	15мкА при =30В	15мкА при =30В
Максимальный пусковой ток	2 А за 100 мс	2 А за 100 мс
Требуется внешний источник питания постоянного тока	Нет данных	Нет данных
Время срабатывания ВЫКЛ-ВКЛ	<10мкс	<10мкс
Время срабатывания ВКЛ-ВЫКЛ	<20мкс	<0.5мс
Срабатывание индикаторов состояния	От логических цепей	От логических цепей
Общие	На 4 канала 1 общий, 4 группы (неизолированые)	
Предохранители	Нет (рекомендуются внешние)	

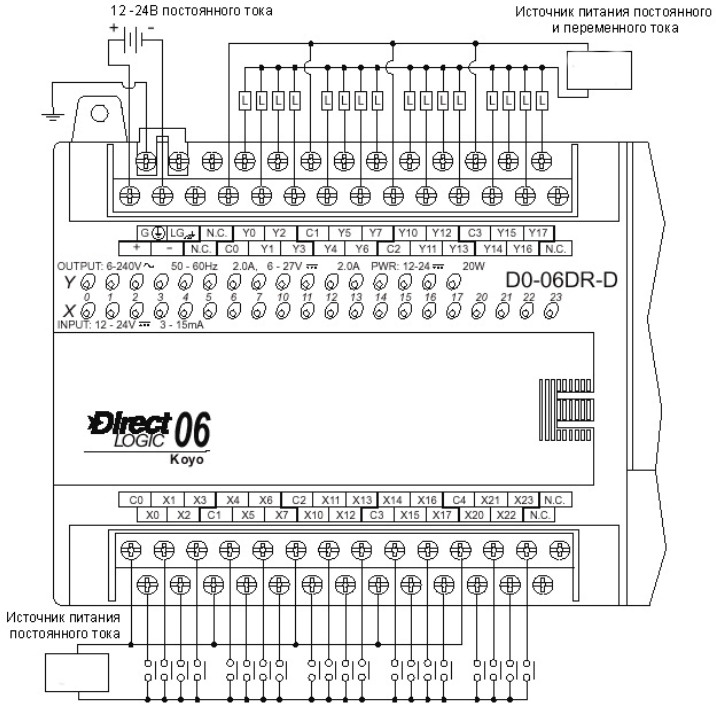
Схема соединений входов/выходов D0- 06DR-D

Микроконтроллер D0-06DR-D имеет двадцать входов постоянного тока и шестнадцать релейных выходов. На схеме приведен пример типичного подключения внешних устройств. Как видно на схеме, для подвода внешнего источника питания используются три клеммы.

Стандартный ресурс реле (1000 операций)

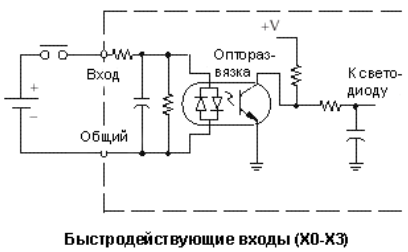
Напряжение и вид нагрузки	Ток нагрузки	
	1A	2A
24В Активная	500К	250К
24В Активная	100К	50К
~110В Активная	500К	250К
~110В Активная	200К	100К
~220В Активная	350К	200К
~220В Активная	100К	50К

График снижения номинальных характеристик в зависимости от температуры для релейных выходов

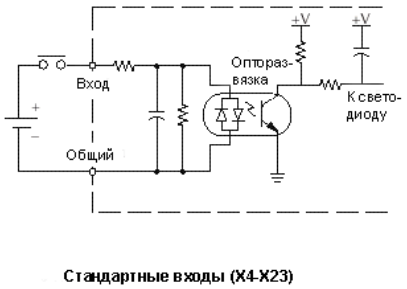


Входы объединены в пять групп по четыре канала. Каждой группе отводится отдельный общий провод и может быть подключена как приемник или источник. В приведенном ниже примере все общие цепи соединены в одну, но можно использовать отдельные источники питания и общие цепи.

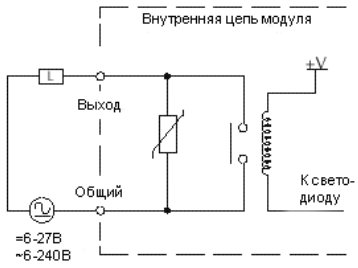
Выходы сгруппированы в четыре группы по четыре реле с нормально разомкнутыми контактами. Каждой группе отводится одна общая клемма. В приведенном примере все общие провода соединены в одной точке, но можно использовать отдельные источники питания и общие цепи. На эквивалентной схеме выходной цепи представлен один канал группы. Релейные контакты могут использоваться в цепях постоянного или переменного тока.



Быстродействующие входы (X0-X3)



Стандартные входы (X4-X23)



Стандартные выходы

Основные характеристики D0-06DR-D	
Потребление электроэнергии	=12-24В, макс. 20Вт
Порт связи 1: 9600 бод (фиксировано), 8 бит данных, 1 стоповый бит, проверка на нечетность	K-Sequence (Slave), DirectNET (Slave), MODBUS (Slave)
Порт связи 2: 9600 бод (по умолчанию), 8 бит данных, 1 стоповый бит, проверка на нечетность	K-Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence / на принтер, ASCII ввод/вывод
Тип программного кабеля	D2--DSCBL
Рабочая температура	от 0 до 55°C
Температура хранения	от -20 до 70°C
Относительная влажность	от 5 до 95% (без конденсации)
Воздушная среда	Без агрессивных газов
Вибрация	MIL STD 810C 514.2
Ударная нагрузка	MIL STD 810C 516.2
Помехоустойчивость	NEMA ICS3-304
Тип клеммной колодки	Съемный
Сечение провода	Один провод 1.3мм^2 или 2 провода 0.78мм^2, мин. 0.2 мм^2

Характеристики входов постоянного тока		
Параметр	**Быстродействующие входы, X0 – X3**	**Стандартные входы постоянного тока X4 –X23**
Диапазон напряжения (мин.-макс.)	=10.8 - 26.4 В	=10.8 - 26.4 В
Рабочий диапазон напряжения	=12 -24 В	=12 -24 В
Максимальное напряжение	=30В (макс. частота - 7 КГц)	=30В
Мин. длительность импульса	70 мкс	Нет
Напряжение Включения	>10В постоянного тока	>10В постоянного тока
Напряжение Выключения	<2.0 В постоянного тока	<2.0 В постоянного тока
Макс. потребляемый ток	6мА при =12В 13мА при =24В	4мА при =12В 8.5мА при =24В
Входное сопротивление	1.8 КОм при =12-24В	2.8КОм при =12-24В
Мин. ток Включения	>5 мА	>4 мА
Макс. ток Выключения	< 0.5 мА	< 0.5 мА
Время срабатывания ВЫКЛ-ВКЛ	<70 мкс	2-8мс, типично 4мс
Время срабатывания ВКЛ-ВЫКЛ	<70 мкс	2-8мс, типично 4мс
Срабатывание индикаторов состояния	От логических цепей	От логических цепей
Общие	На 4 канала 1 общий провод, 5 групп (изолированые)	

Характеристики релейных выходов	
Диапазон выходного напряжения (мин. – макс.)	~5 - 264В (47 -63 Гц), =5-30В
Рабочий диапазон напряжения	~6 - 240В (47 -63 Гц), =6-27В
Выходной ток	2А/канал, 6А / общий
Максимальное напряжение	~264В, =30В
Максимальный ток утечки	0.1 мА при ~264 В
Минимальная рекомендуемая нагрузка	5 мА
Время срабатывания ВЫКЛ-ВКЛ	<15 мс
Время срабатывания ВКЛ-ВЫКЛ	<10 мс
Срабатывание индикаторов состояния	От логических цепей
Общие	На 4 канала 1 общий, 4 группы (изолированые)
Предохранители	Нет (рекомендуются внешние)

Глава 3

3. Технические характеристики высокоскоростного ввода и импульсного выхода

В этой главе…

Введение ..... **3—2**
Выбор рабочего режима высокоскоростного ввода/вывода ..... **3—4**
Режим 10: Высокоскоростной счетчик ..... **3—7**
Режим 20: Реверсивный счетчик ..... **3—24**
Режим 30: Импульсный выход ..... **3—38**
Режим 40: Высокоскоростные прерывания ..... **3—64**
Режим 50: Вход с защелкой импульсов ..... **3—69**
Режим 60: Дискретные входы с фильтром ..... **3—74**

Введение

Встроенное решение по управлению движением

Для многих приложений по управлению машинами требуются различные типы простого высокоскоростного слежения и управления. Эти приложения обычно включают отдельные типы управления движением или высокоскоростные прерывания для критичных по времени событий. Микроконтроллер DL06 решает эти традиционно сложные задачи с помощью встроенных технических возможностей процессора.

Функциями высокоскоростного ввода являются:

- Высокоскоростной счетчик (максимум 7КГц) с 24 заданными параметрами и встроенной подпрограммной прерывания, со счетом в одном направлении и сбросом.
- Входы квадратурного кодировщика угла поворота (энкодера) для отсчета в направлениях по часовой стрелке или против часовой стрелки (до 7КГц), со счетом в прямом/обратном направлениях и сбросом.
- Высокоскоростной вход прерывания для немедленной реакции в критичных по времени задачах.
- Функция защелки (фиксации) импульсов при отслеживании одной входной точки с длительностью импульса до 100 мкс (0.1 мс).
- Программируемая дискретная фильтрация (с задержкой по времени как на включение, так и на выключение сигнала до 99 мс), обеспечивающая целостность сигнала (этот режим установлен по умолчанию для входов X0 – X3).

Функциями импульсного вывода являются:

- Одноканальный программируемый импульсный выход (максимум 10КГц) с тремя типами профилей, включая трапецеидальное движение, точное совмещение и управление скоростью.

Возможность использования функций высокоскоростного ввода/вывода (HSIO)

ВАЖНО! Применение функций имеет следующие ограничения:

- Варианты высокоскоростного ввода доступны только для DL06 с входами постоянного тока.
- Варианты импульсного вывода доступны только для DL06 с выходами постоянного тока.
- Одновременно может использоваться только одна функция высокоскоростного ввода/вывода. Вы не можете использовать высокоскоростной вход и импульсный выход в одно и то же время.

Номер модели DL06	Тип дискретного входа	Тип дискретного выхода	Высокоскоростной ввод	Импульсный выход
D0-06AA	Переменный ток	Переменный ток	Нет	Нет
D0-06AR	Переменный ток	Релейный	Нет	Нет
D0-06DA	Постоянный ток	Переменный ток	Да	Нет
D0-06DD1	Постоянный ток	Постоянный ток	Да	Да
D0-06DD2	Постоянный ток	Постоянный ток	Да	Да
D0-06DR	Постоянный ток	Релейный	Да	Нет
D0-06DD1-D	Постоянный ток	Постоянный ток	Да	Да
D0-06DD2-D	Постоянный ток	Постоянный ток	Да	Да
D0-06DR-D	Постоянный ток	Релейный	Да	Нет

Специализированная схема высокоскоростного ввода/вывода

Основной внутренней задачей процессора является выполнение пользовательской программы и чтение/запись входов и выходов в каждом цикле сканирования. Для обслуживания событий высокоскоростного ввода/вывода DL06 имеет специальную схему, которая предназначена для части входных/выходных точек. См. блок-схему на рисунке ниже.

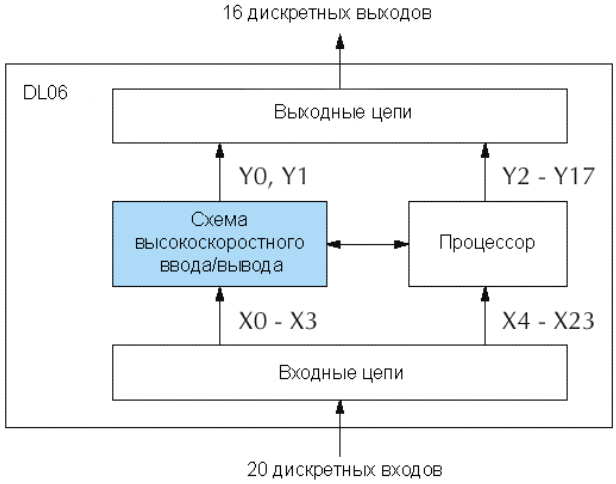
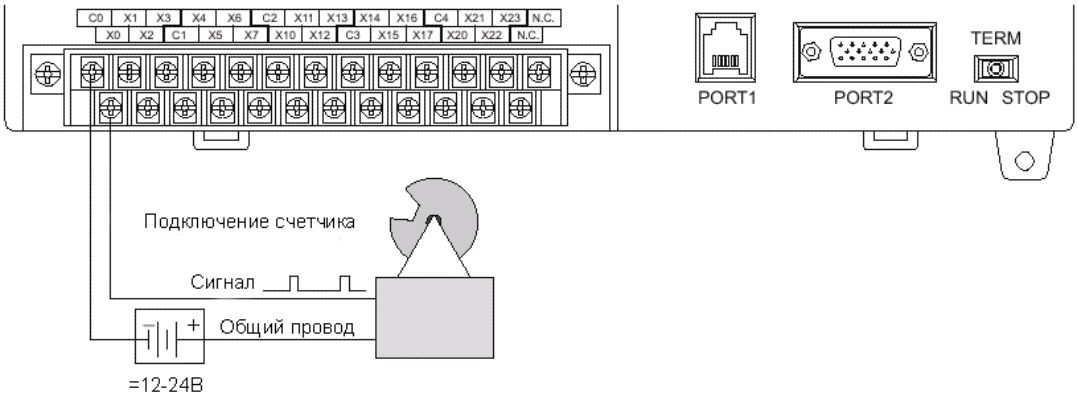


Схема высокоскоростного ввода/вывода (HSIO) предназначена для первых четырех входов (X0 – X3) и первых двух выходов (Y0 - Y1). Эту схему можно рассматривать как «помощник процессора». В режиме по умолчанию (называемым «Режим 60») схема высокоскоростного ввода/вывода только пропускает входные/выходные сигналы в процессор и из процессора, поэтому все двадцать входов работают одинаково, и все шестнадцать выходов также работают одинаково. Когда процессор конфигурируется для какого-либо режима высокоскоростного ввода/вывода, то эта схема получает специализированную функцию для указанной части входов и выходов. Схема высокоскоростного ввода/вывода *функционирует независимо от сканирования программ процессором.* Это позволяет проводить точные измерения и осуществлять работу высокоскоростного ввода/вывода, пока процессор занят выполнением пользовательской программы.

Схемы подключения для каждого режима высокоскоростного ввода/вывода

После того, как Вы выбрали режим высокоскоростного ввода/вывода для вашего приложения, вам следует обратиться к соответствующему разделу данной главы по этому режиму. Каждый раздел включает схему подключения, которая поможет вам правильно подключить точки высокоскоростного ввода/вывода к полевым устройствам. Ниже показана схема для режима высокоскоростного счетчика.



Выбор рабочего режима высокоскоростного ввода/вывода

Шесть режимов высокоскоростного ввода/вывода

Схема высокоскоростного ввода/вывода работает в одном из 6 базовых режимов, перечисленных ниже в таблице. Число в левом столбце является номером режима (позже мы будем использовать этот номер при конфигурировании). Выберите один из режимов в соответствии с основной функцией, которую Вы хотите получить от схемы высокоскоростного ввода/вывода. Вы можете просто использовать все двадцать входов и шестнадцать выходов как обычные точки ввода/вывода в режиме 60.

Наименование режима		Характеристика режима
10	Высокоскоростной счетчик	Два счетчика 7КГц с 24 заданными параметрами, со счетом в одном направлении, со сбросом и прерыванием по заданному значению
20	Реверсивный счетчик	Реверсивный счетчик 7КГц с 24 заданными параметрами, со сбросом и прерыванием по заданному значению
		Квадратурный вход 7КГц по каналу А / каналу В, со счетом в прямом и обратном направлениях
30	Импульсный выход	Управление шаговым приводом - импульсные сигналы и сигналы направления, программируемый профиль движения
40	Высокоскоростное прерывание	Генерирует прерывание исходя из перехода состояния входа или по времени
50	Защелка (фиксатор) импульсов	Захватывает короткие импульсы в выбранном входе
60	Фильтрованный вход	Отфильтровывает короткие импульсы в выбранном входе

При выборе одного из шести режимов высокоскоростного ввода/вывода перечисленные в таблице ниже точки входа/выхода реализуют только указанные в таблице функции. Если вход не используется специально для поддержания конкретного режима, то он обычно работает по умолчанию как фильтрованный вход. Аналогично выход функционирует как обычный выход, если не выбран режим импульсного выхода.

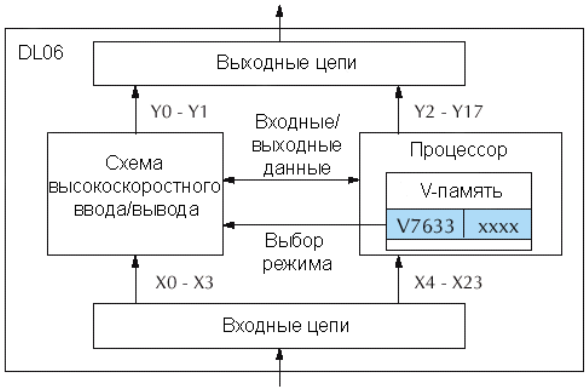
Использование физических входов/выходов							
Наименование режима		**Входы постоянного тока**				**Выходы постоянного то**	
		X0	**X1**	**X2**	**X3**	**Y0**	**Y1**
10	Высокоскоростной счетчик	Счетчик #1	Счетчик #2, прерывание, импульсный вход или фильтрованный вход	Сброс #1, прерывание, импульсный вход или фильтрованный вход	Сброс #2, прерывание, импульсный вход или фильтрованный вход	Обычный выход	Обычный выход
20	Реверсивный счетчик (стандартное вычисление)	Вычисление в прямом напраление	Вычисление в обратном напраление	Сброс, импульсный вход или фильтрованный вход	Импульсный вход или фильтрованный вход	Обычный выход	Обычный выход
	Реверсивный счетчик (квадратурное вычисление	Вход фазы А	Вход фазы В				
30	Импульсный выход	Импульсный вход или фильтрованный вход	Импульсный вход или фильтрованный вход	Импульсный вход или фильтрованный вход	Импульсный вход или фильтрованный вход	Импульс или импульс по часовой стрелке	Направление или импульс против часовой стрелке
40	Высокоскоростное прерывание	Прерывание	Прерывание, импульсный вход или фильтрованный вход	Прерывание, импульсный вход или фильтрованный вход	Прерывание, импульсный вход или фильтрованный вход	Обычный выход	Обычный выход
50	Фиксатор импульсов	Импульсный вход	Импульсный вход, прерывание, или фильтрованный вход	Импульсный вход, прерывание, или фильтрованный вход	Импульсный вход, прерывание, или фильтрованный вход	Обычный выход	Обычный выход
60	Фильтрованный вход	Фильтрованный вход	Фильтрованный вход	Фильтрованный вход	Фильтрованный вход	Обычный выход	Обычный выход

Режим по умолчанию

Режим 60 (Фильтрованные входы) является режимом по умолчанию. DL06 устанавливается в этом режиме в заводских условиях, и в любой момент Вы можете восстановить память с этой информацией. В режиме по умолчанию X0 – X3 являются фильтрованными входами (задержка 10мс), а Y0 - Y1 являются стандартными выходами.

Конфигурирование режима высокоскоростного ввода/вывода

Если Вы выбрали подходящий для вашего приложения режим высокоскоростного ввода/вывода, то можно произвести соответствующее конфигурирование работы ПЛК. На приведенной ниже блок-схеме обращаем внимание на элемент V-памяти в блоке Процессор. Ячейка V-памяти V7633 определяет функциональный режим высокоскоростного ввода/вывода. *Это — наиболее важная величина для выбора функций высокоскоростного ввода/вывода!*



Ячейка V7633 содержит 16-битовое слово, которое должно вводиться в двоично-десятичном формате. На рисунке ниже показаноо, что представляет собой каждая 4-битовая цифра этого двоично-десятичного слова.

Биты 0 - 7 определяют указанный ранее номер режима. Например, значение «0050» показывает выбор режима 50 - фиксация импульсов (в двоично-десятичном формате = 50).

Конфигурирование входов X0 – X3

Кроме настройки ячейки V7633 для режима HSIO, вам необходимо запрограммировать следующие четыре ячейки для заданного режима в соответствии с желаемой функцией входных точек X0 – X3. Может потребоваться конфигурирование других ячеек V-памяти в зависимости от режима HSIO (См. соответствующий раздел по конкретному режиму HSIO).

Режим	V-память	
	V7633	xxxx
X0	V7634	xxxx
X1	V7635	xxxx
X2	V7636	xxxx
X3	V7637	xxxx

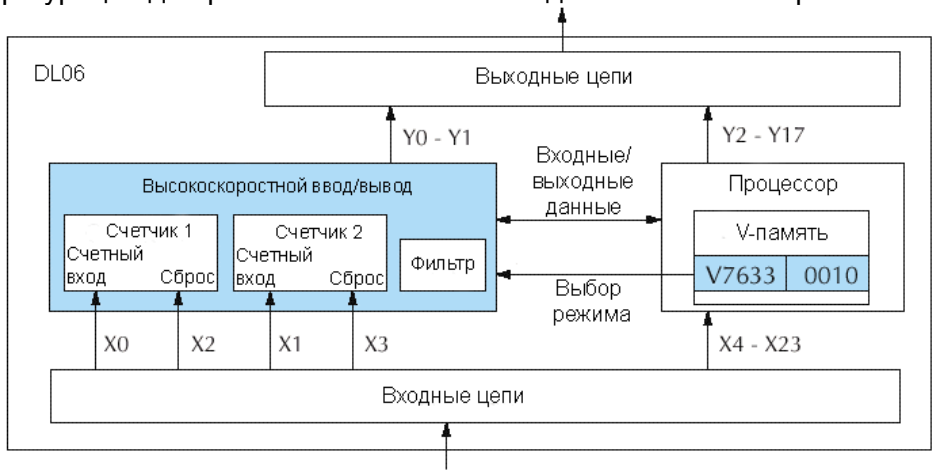
Режим 10: Высокоскоростной счетчик

Назначение

Схема высокоскоростного ввода/вывода предусматривает два высокоскоростных счетчика. Одна последовательность импульсов от внешнего источника (X0) включает в работу счетчик по каждому переднему фронту импульсов сигнала. Счетчик производит отсчет только в прямом направлении, от 0 до 99999999. Счетчик сравнивает текущее значение с 24 установленными значениями (Preset Value), которые Вы определяете. Предварительно установленные значения предназначены для немедленного выполнения определенных действий, когда текущее значение счетчика становится равным одному из этих значений. Это — идеальная возможность для таких приложений, как например, резка на равные части. Для заданных значений используются в процессоре регистры счетчика CT174 и CT177.

Функциональная блок-схема

Функциональная блок-схема приведена на рисунке ниже. Если младший байт регистра режимов HSIO V7633 содержит двоично-десятичное число «10», то в схеме HSIO включается высокоскоростной суммирующий счетчик. X0 и X1 автоматически становятся входами «генератора импульсов» для высокоскоростного счетчика, наращивая его при каждом переходе «выключено — включено». По умолчанию X2 и X3 в конфигурации для режима 10 являются входами внешнего сброса.



Вы можете сконфигурировать X2 и X3 как обычные фильтрованные входы, вместо того чтобы использовать их как специализированные входы сброса. В этом случае сброс счетчика должен вырабатываться программой релейной логики.

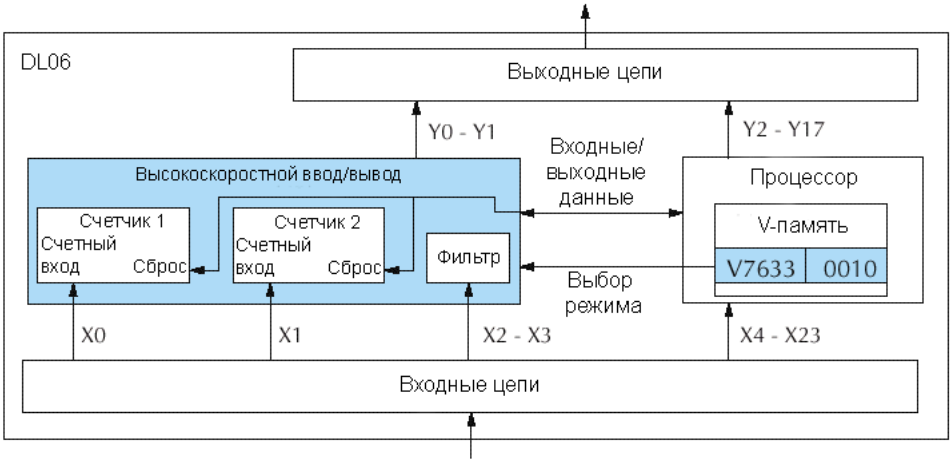
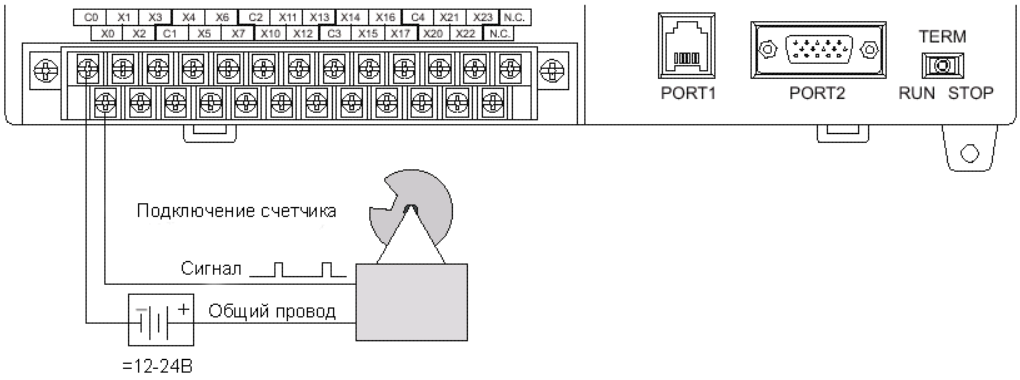


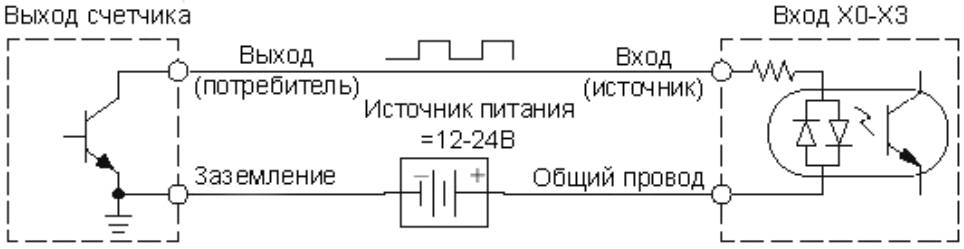
Схема подключения

Ниже показана общая схема подсоединения счетчиков/кодировщиков к DL06 в режиме 10. Могут использоваться многие типы устройств генерирующих импульсы, такие как бесконтактные переключатели, одноканальные кодировщики, магнитные и оптические чувствительные элементы и др. Устройства с выходами — потребителями тока (транзисторы с открытым коллектором NPN), по-видимому, являются наилучшим выбором для сопряжения. Если счетчик является источником питания для входов, он должен иметь на выходе от 12 до 24 В постоянного тока. Обращаем внимание на то, что устройства с выходами 5В-источник, не будут работать с входами DL06.

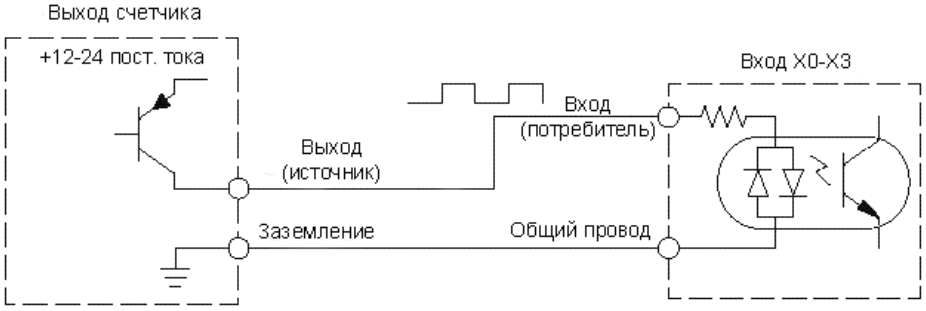


Сопряжение с выходами счетчика

Входы постоянного тока DL06 являются гибкими в том отношении, что они допускают ток в любом направлении, поэтому они могут подсоединяться к счетчику как выходы—источники или как выходы—приемники тока. В следующей схеме счетчик имеет выходы транзистора с открытым коллектором NPN. Он принимает ток от входной точки ПЛК, которая является источником тока. Источником питания может быть FA-24PS или любой другой источник (на +12В или на +24В), удовлетворяющий техническим условиям входа.

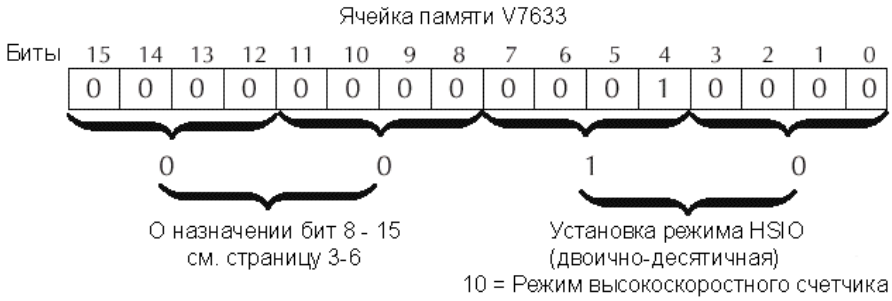


В следующей схеме кодировщик имеет выходы транзистора с открытым эмиттером PNP. Он является источником тока для входной точки ПЛК, которая обратным проводом соединена с заземлением. Поскольку кодировщик является источником тока, никакие дополнительные источники питания не требуются. Но следует отметить, что кодировщик должен иметь от 12 до 24 В (при выходе кодировщика 5 В схема не будет работать).



Настройка на режим 10

Ячейка V7633 является регистром выбора режимов HSIO. Используйте двоично-десятичное число «10» в младшем байте V7633 для выбора режима высокоскоростного счетчика.



Выберите из следующего списка наиболее удобный способ программирования ячейки V7633:

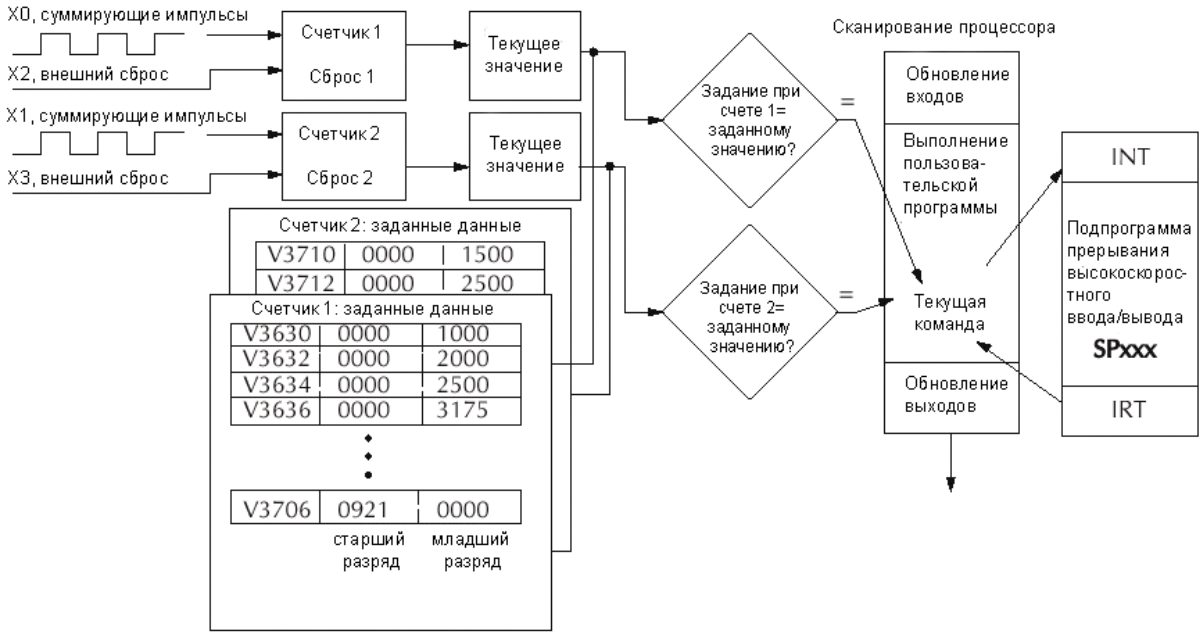
- Включить в свою программу команды LOAD и OUT.
- Использовать редактор памяти (Memory Editor) DirectSOFT или просмотр данных.
- Использовать ручной программатор D2-HPP.

Мы рекомендуем использовать первый способ, при котором настройка высокоскоростного ввода/вывода становится неотъемлемой частью вашей прикладной программы. В примере программы, приведенном далее в данном разделе, показывается, как это делается.

Заданные значения и специальные реле

Предварительная установка используется, чтобы выполнить определенное действие, когда при подсчете будет достигнуто заданное значение. Посмотрите рисунок ниже. Каждый счетчик может иметь до 24 предварительно заданных значений(Preset Value), которые могут быть запрограммированы. Предварительно заданные значения представляют двойное слово, так что они занимают два регистра V-памяти. Пользователь выбирает предварительно установленные значения, и счетчик непрерывно сравнивает текущий счет с предварительно установленным. Когда они равны, контакты специального реле замыкаются, и происходит переход к выполнению подпрограммы прерывания.

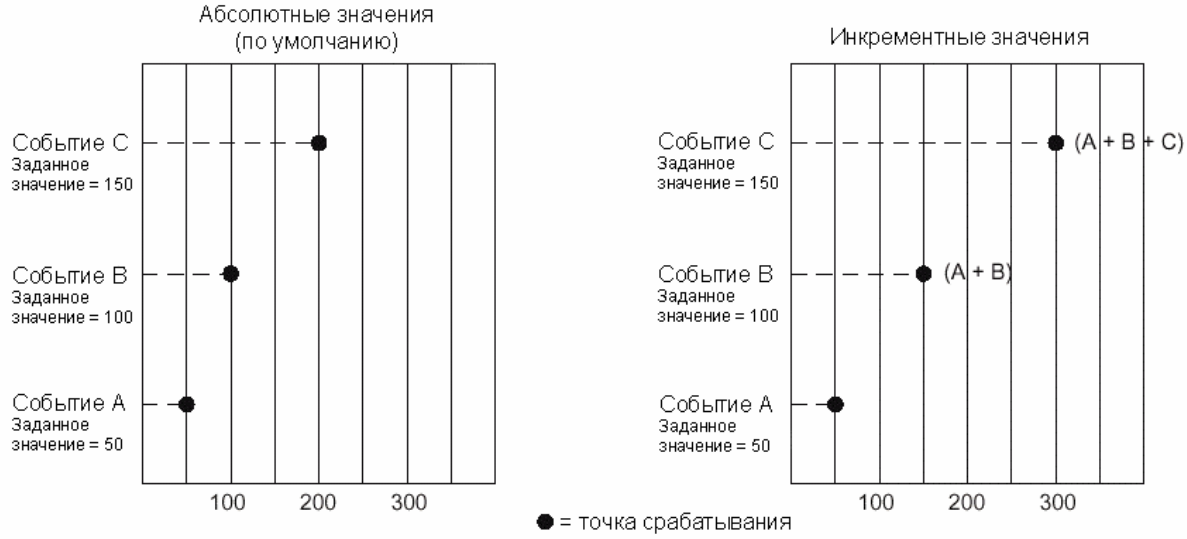
Мы рекомендуем использовать специальное(ые) реле в сервисной подпрограмме прерывания, чтобы обеспечить немедленное выполнение желательных действий. После завершения сервисной подпрограммы прерывания процессор возвращается в пользовательскую программу и продолжает ее выполнения с точки прерывания. Далее производится сравнение со следующим заданным значением.



Абсолютные и инкрементальные предустановленные значения

Доступны два режима предустановленных значений: абсолютный и инкрементальный.

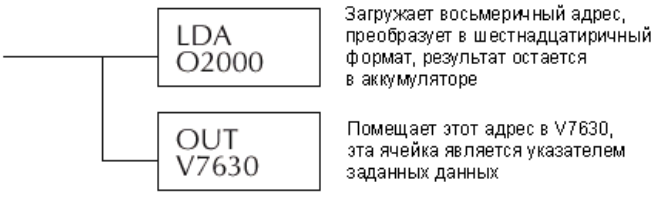
Предустановленные значения предварительно записываются в регистрах V-памяти. В абсолютном режиме, каждое значение обрабатывается как итоговое. В инкрементном режиме предварительно заданное значение как накопленное. Инкрементные значения представляют собой значения счетчика между событиями.

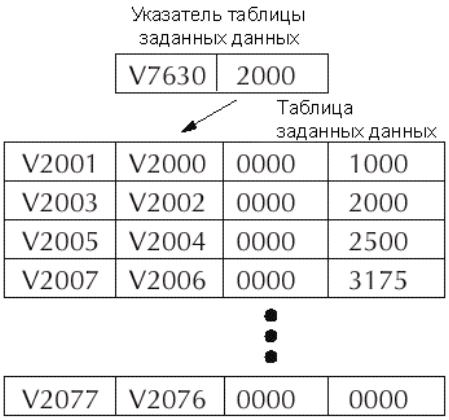


В приведенном примере заданные значения равны 50, 100 и 150. Различие между абсолютным и инкрементальным режимами показано. Абсолютные уставки запускают события по заданным значениям: 50, 100 и 200. Инкрементальные уставки запускают события по накопленным значениям: 50, 150 и 300.

Начальная ячейка заданных данных

V7630 является ячейкой V-памяти указателя, которая указывает на Таблицу Заданных Данных. По умолчанию начальной ячейкой для Таблицы Заданных Данных является V3630 (значение по умолчанию устанавливается после инициации оперативной V-памяти). Однако Вы можете программным путем изменить в V7630 адрес этой ячейки. Используйте команды LDA и OUT следующим образом:



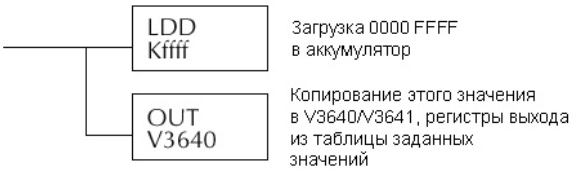


Использование менее 24 заданных значений

Если используются все 24 заданных значения , процессор заведомо знает, когда достигается конец таблицы заданных данных. Когда используется менее 24 значений необходимо оповестить процессор о том, что достигнуто последнее значение. Способ оповещения о конце блока предварительно заданных значений состоит в том, чтобы вставить один из следующих кодов «конеца таблицы» в следующую доступную пару регистров:

Код конеца таблицы	Применяемый режим	Значение
0000 FFFF	Абсолютный и инкрементный	Сообщение о достижении последнего значения
0000 00FF	Инкрементный	Сообщение о достижении последнего значения и повторный запуск заданных значений. Не сбрасывается накопленный отчет импульсов CT174 или CT176.
0000 FF00	Инкрементный	Сообщение о достижении последнего значения, повторный запуск заданных значений и сброс накопленного отчета импульсов CT174 или CT176.

Как показано в таблице выше, каждый из сигналов «конец таблицы» имеет различное значение. Используйте команду LDDKfff, чтобы вставить код конца таблицы в следующую пару регистров для выхода из таблицы. В примере используется четыре заданных значения. 0000 FFFF в V3641-V3640 указывает, что предыдущее заданное значение было последним значением.



Пример таблицы заданных значений по умолчанию

V3631	V3630	0000	1000
V3633	V3632	0000	2000
V3635	V3634	0000	2500
V3637	V3636	0000	3175
V3641	V3640	0000	FFFF

В инкрементном режиме, Вы можете выбрать: не сбрасывать счетчик или накопленное значение, можете сбросить только счетчик или сбросить и счетчик и накопленное значение, когда код таблица-конец прочитан. В примере FFFF был помещен в V3640, так как последнее заданное значение было в V3636 и мы используем меньше чем 24 предварительно заданных значений.

ПРИМЕЧАНИЕ. *В инкрементном режиме каждое последующее заданное значение должно быть больше предыдущего. Если заданное значение меньше заданного значения с меньшим номером, то процессор не может проводить сравнение с этим значением, так как счетчик ведет отсчет только в прямом направлении.*

Номера эквивалентных реле

В приводимой ниже таблице перечислены все 24 установленные по умолчанию ячейки с регистрами заданных значений для каждого высокоскоростного счетчика. Каждое такое значение занимает два 16-битовых регистра в V-памяти. Номера контактов соответствующих специальных реле приведены в следующем столбце. Можно назвать эти контакты реле «эквивалентными», так как они истинны (замкнуты), когда текущее значение высокоскоростного счетчика равно заданному значению. Каждый контакт остается закрытым до тех пор, пока значение счетчика не станет равным следующему заданному значению.

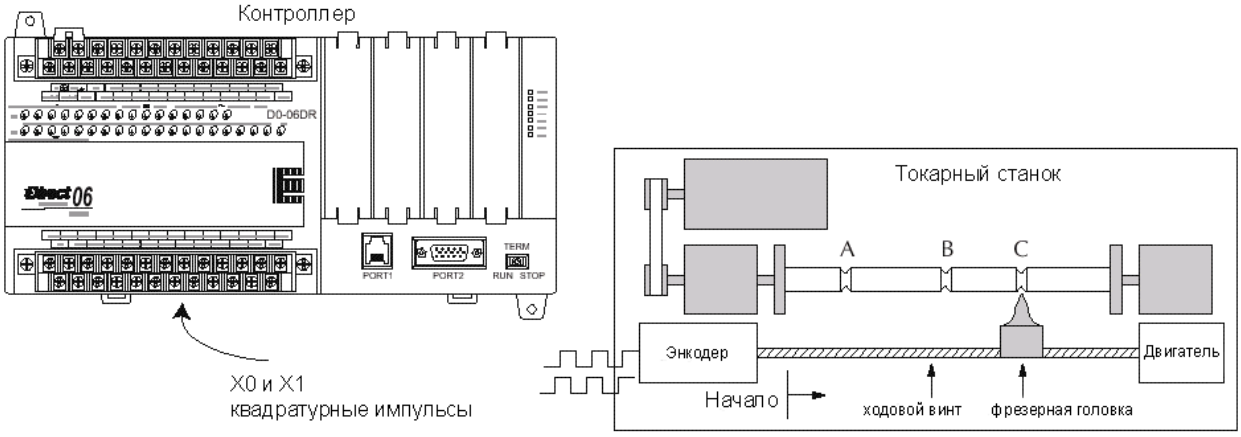
Счетчик 1 Заданное значение	Регистр заданного значения в V- памяти	Номер специального реле	Счетчик 2 Заданное значение	Регистр заданного значения в V- памяти	Номер специального реле
1	V3631 / V3630	SP540	1	V3711/V3710	SP570
2	V3633 / V3632	SP541	2	V3713/V3712	SP571
3	V3635 / V3634	SP542	3	V3715/V3714	SP572
4	V3637 / V3636	SP543	4	V3717/V3716	SP573
5	V3641 / V3640	SP544	5	V3721/V3720	SP574
6	V3643 / V3642	SP545	6	V3723/V3722	SP575
7	V3645 / V3644	SP546	7	V3725/V3724	SP576
8	V3647 / V3646	SP547	8	V3727/V3726	SP577
9	V3651 / V3650	SP550	9	V3731/V3730	SP600
10	V3653 / V3652	SP551	10	V3733/V3732	SP601
11	V3655 / V3654	SP552	11	V3735/V3734	SP602
12	V3657 / V3656	SP553	12	V3737/V3736	SP603
13	V3661 / V3660	SP554	13	V3741/V3740	SP604
14	V3663 / V3662	SP555	14	V3743/V3742	SP605
15	V3665 / V3664	SP556	15	V3745/V3744	SP606
16	V3667 / V3666	SP557	16	V3747/V3746	SP607
17	V3671 / V3670	SP560	17	V3751/V3750	SP610
18	V3673 / V3672	SP561	18	V3753/V3752	SP611
19	V3675 / V3674	SP562	19	V3755/V3754	SP612
20	V3677 / V3676	SP563	20	V3757/V3756	SP613
21	V3701 / V3700	SP564	21	V3761/V3760	SP614
22	V3703 / V3702	SP565	22	V3763/V3762	SP615
23	V3705 / V3704	SP566	23	V3765/V3764	SP616
24	V3707 / V3706	SP567	24	V3767/V3766	SP617

Последовательные адреса показанные выше для каждого реле – адреса назначенные процессором по умолчанию. Указатель начального адреса хранится процессором в ячейке V7630. Если Вы уже использовали эти адреса, Вы можете изменить блок адрессов по умолчанию программным способом изменяя указатель значения в ячейке V7630. Для изменения положения таблицы используйте команды LDA и OUT, как показано на предыдущей странице.

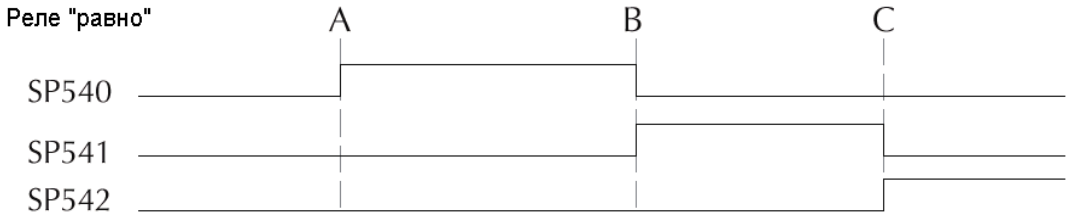
Расчет заданных значений

Заданные значения занимают два слова каждое. Они могут изменяться в пределах от -8388608 до 8388607, также как и значения высокоскоростного счетчика. Все 24 значения являются абсолютными значениями, что означает, что каждое из них отсчитывается от нулевого значения счетчика.

Заданные значения должны устанавливаться конкретно для каждого приложения. На приведенном ниже рисунке ПЛК отслеживает положение подающего винта токарного станка посредством отсчета импульсов. В точках A, B и C при линейном движении резцовая головка входит в рабочий материал и делает проточку.



Ниже показана временная диаграмма длительности закрытия контактов эквивалентных реле. Каждый контакт остается замкнутым до момента, пока не будет замкнут следующий контакт. Все они отключаются при сбросе счетчика.



ПРИМЕЧАНИЕ. *Каждое последующее заданное значение должно быть больше предыдущего на два значения. В примере токарного станка B > A + 2 и C> B + 2.*

Конфигурирование входа X

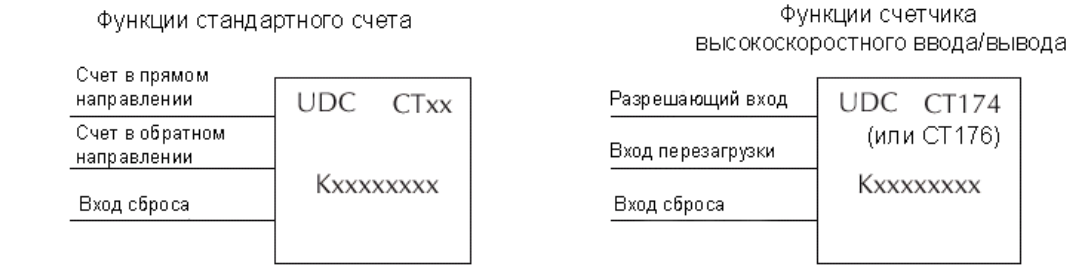
Варианты конфигурируемого дискретного входа для режима высокоскоростного счетчика приведены в таблице ниже. Вход X0 выделен как вход импульсов первого счетчика. Вход X1 может быть импульсным входом для второго счетчика или фильтрованным. В разделе по функционированию режима 60 в конце данной главы описывается программирование временных констант фильтра. Входы X2 и X3 могут быть сконфигурирован как сброс счетчика, с прерыванием или без него. Вариант с прерыванием позволяет входу сброса (X2 и X3) вызвать прерывание аналогично тому, как это делает заданное значение, но без замыкания контактов специального реле (вместо этого X2 и X3 будут в состоянии включен в течение работы подпрограммы прерывания в одном цикле сканирования). Наконец, X2 и X3 могут просто остаться как фильтрованный вход.

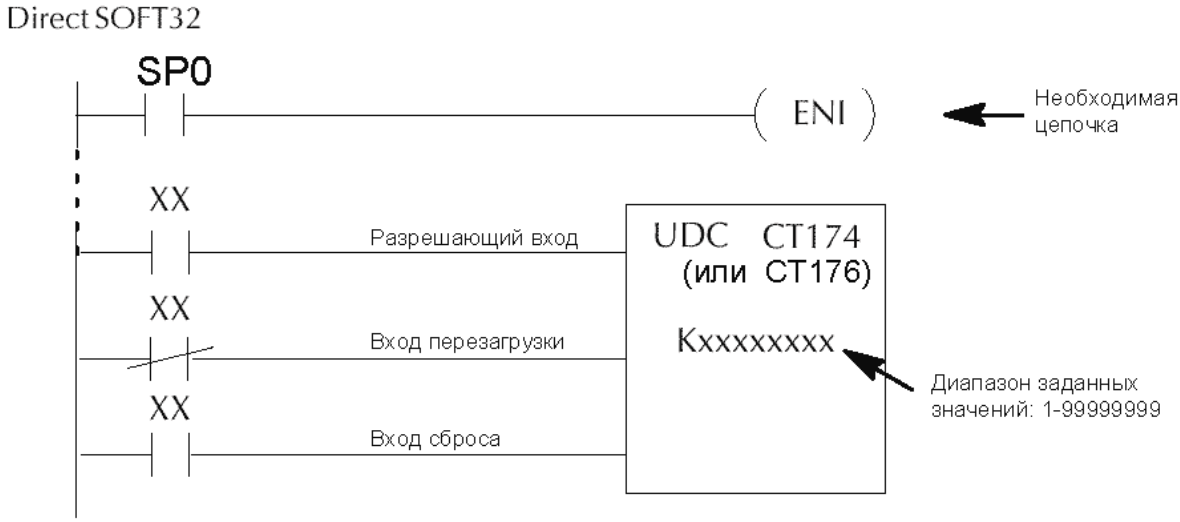
Вход	Регистр конфигурации	Функция	Требуемый шестнадцатеричный код
X0	V7634	Счетные импульсы #1	0001 (абсолютный) (по умолчанию)
			0101 (инкрементный)
X1	V7635	Счетные импульсы #2	0001(абсолютный) (по умолчанию)
			0101 (инкрементный)
		Прерывание	0004
		Импульсный вход	0005
		Фильтрованный вход	xx06, xx= время фильтрации от 0 до 99 мс (двоично-десятичных)
X2	V7636	Сброс счетчика #1 (без прерывания)	0007* (по умолчанию) 0207*
		Сброс счетчика #1 (с прерыванием)	0107* 0307*
		Прерывание	0004
		Импульсный вход	0005
		Фильтрованный вход	xx06, xx= время фильтрации от 0 до 99 мс (двоично-десятичных)
X3	V7637	Сброс счетчика #2 (без прерывания)	0007* (по умолчанию) 0207*
		Сброс счетчика #2 (с прерыванием)	0107* 0307*
		Прерывание	0004
		Импульсный вход	0005
		Фильтрованный вход	xx06, xx= время фильтрации от 0 до 99 мс (двоично-десятичных)

* Вы имеете варианты обычного или быстрого сброса счетчика. Но при быстром сбросе не распознаются измененные заданные значения, установленные в течение работы программы. Если в ячейке V7636 или V7637 установлены значения '0007' или '0107', и заданные значения изменялись в течение работы программы, то DL06 распознает эти измененные заданные значения в момент сброса. Когда в ячейке V7636 или V7637 установлены значения '0207' или '0307', процессор не проверяет изменение заданных значений, поэтому DL06 имеет меньшее время сброса.

Написание управляющей программы

Счетчик имеет мнемоническое обозначение UDC (реверсивный счетчик). DL06 может иметь 128 счетчиков с обозначениями от CT0 до CT177. Высокоскоростной счетчик в схеме HSIO доступен программе релейной логики при использовании только UDC CT174 и CT176. Он занимает регистры счетчика от CT174 до CT177 только тогда, когда режим 10 HSIO активен (иначе счетчики от CT174 до CT177 будут стандартными счетчиками). Счетчик HSIO требует два регистра, так как он является счетчиком с двойным словом. Он имеет, как показано, три входа. Первый вход (Разрешающий-Enable Input) позволяет запустить счет, когда счетчик находится в активном состоянии. Средний вход (Preload Input) используется для предустановки значения счетчика. Нижний вход является сигналом сброса (Reset Input). Вход предустановки должен быть отключен, когда счетчик работает.
На следующем рисунке показано, как вставить высокоскоростной счетчик в программу релейной логики. Обратите внимание, что команду разрешения прерывания (ENI), необходимо выполнить прежде, чем счетчик достигнет первого предустановленного значения. Мы делаем это при включении питания, используя реле первого сканирования SP0. При использовании счетчика без предустановленных значений и без прерываний команду ENI можно пропустить.

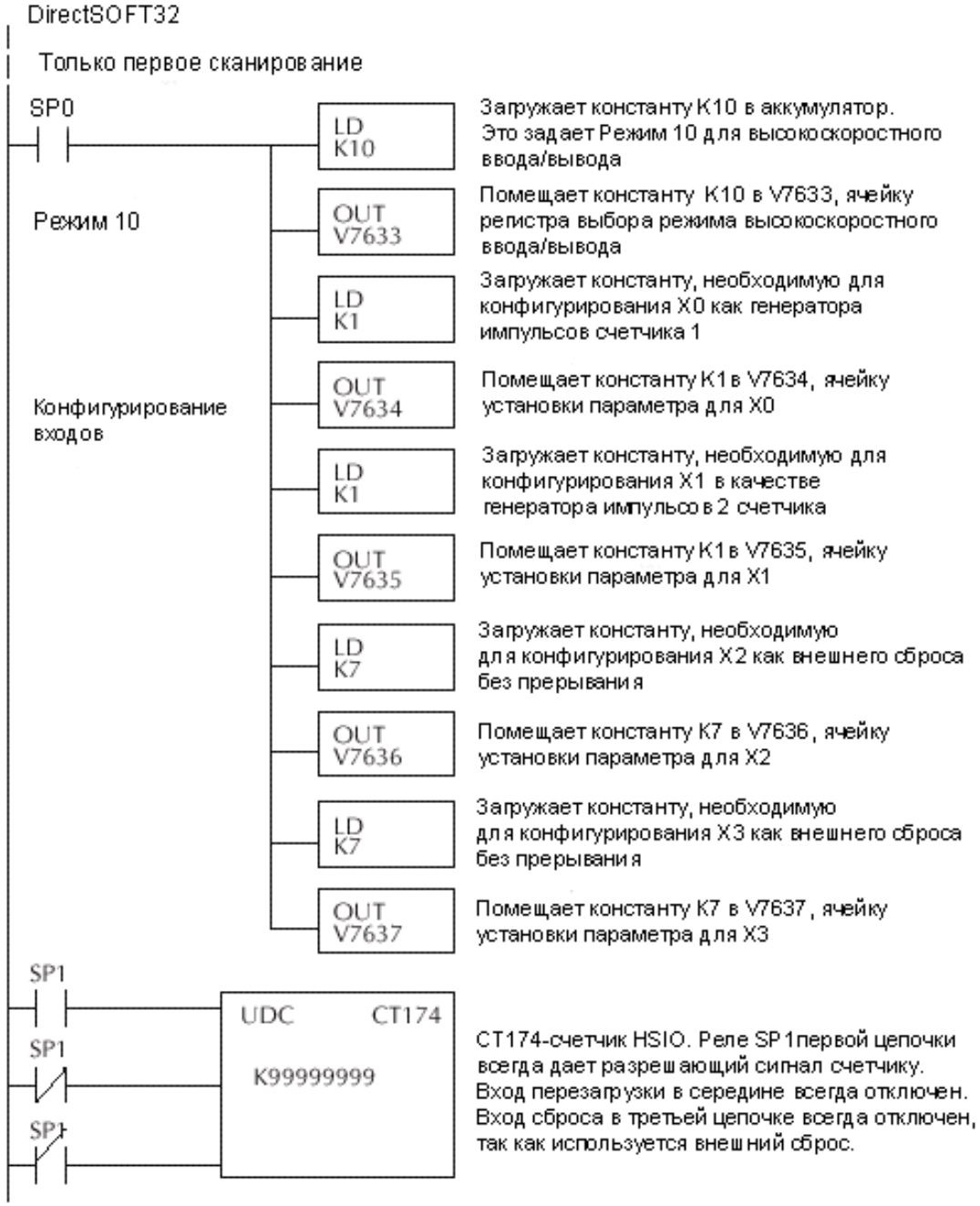




При включении разрешающего входа реверсивный счетчик CT174 начнет реагировать на импульсы X0 и наращивать значение счетчика. Реверсивный счетчик CT176 начнет реагировать на импульсы X1 и наращивать значение счетчика. Контакты входа сброса работают в режиме логического ИЛИ (OR) с физическим входом сброса X2, если выбран сброс счетчика 1, и X3, если выбран сброс счетчика 2. Поэтому высокоскоростной счетчик может получать сигнал сброса либо от контактов программной цепочки сброса, ИЛИ от внешнего входа сброса X2 и X3, если вы сконфигурировали X2 или X3 как сигналы внешнего сброса.

Пример программы 1: счетчик без заданных значений

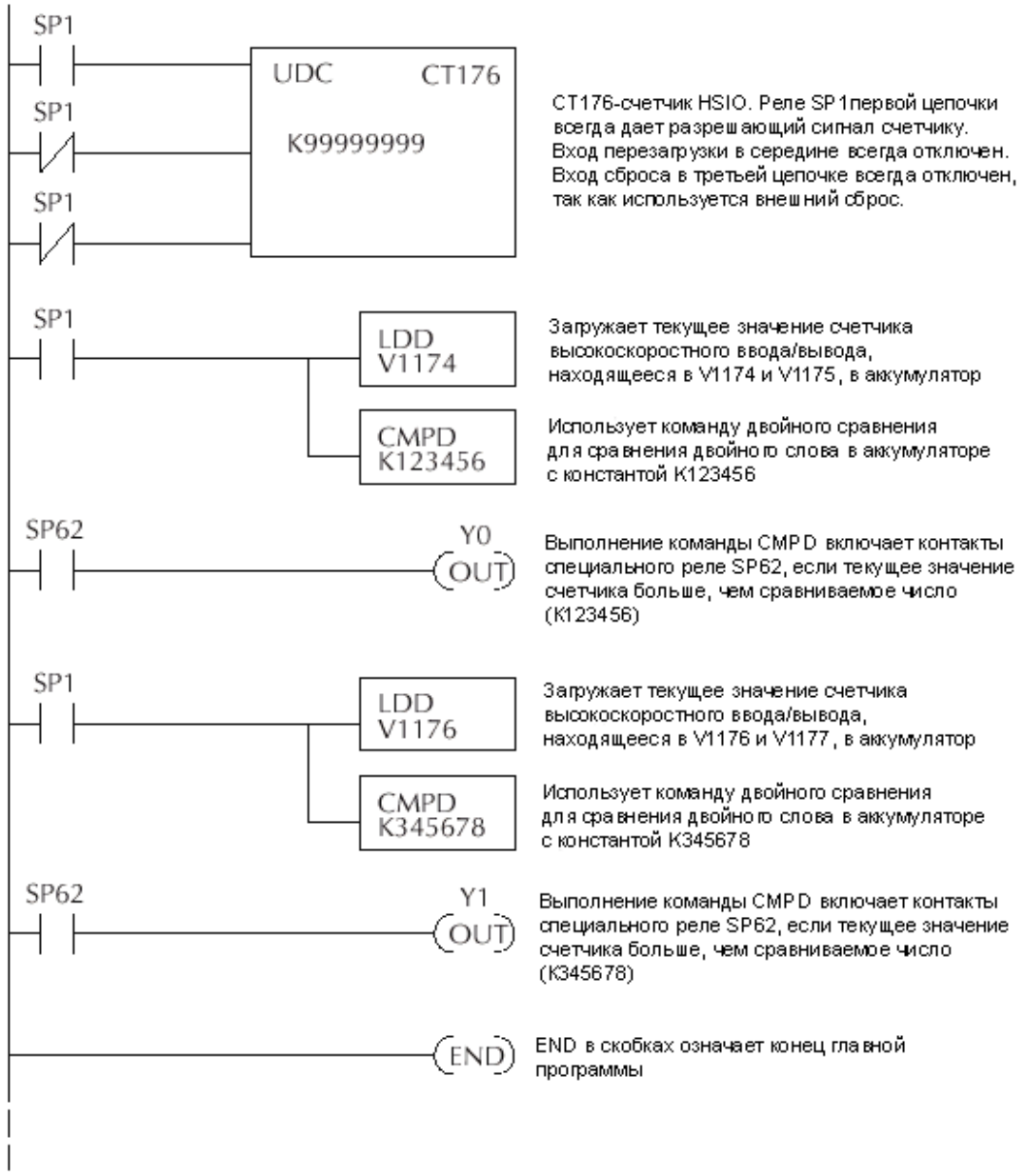
Следующий пример иллюстрирует самый простой способ применения высокоскоростных счетчиков, в котором не используются заданные значения и специальные реле в программе прерывания. В программе схема высокоскоростного ввода/вывода конфигурируется для работы в режиме 10, поэтому X0 автоматически становится генератором импульсов первого счетчика и X1 генератором импульсов для второго счетчика. В примере используется команда двойного сравнения (CMPD) для включения действий при определенных значениях счетчика. Следует отметить, что эта команда позволяет вам иметь более 24 «заданных значений». В этом случае X2 и X3 конфигурируются как внешний сброс счетчика.



продолжение на следующей странице

Продолжение примера программы

Команда двойного сравнения использует текущее значение счетчика высокоскоростного ввода/вывода для включения Y0 и Y1. Данный метод позволяет делать более 24 сравнений, но их число зависит от продолжительности цикла сканирования. Используйте 24 встроенных заданных значений с программой прерывания, если ваше приложение требует быстрой реакции, как показано в следующем примере.



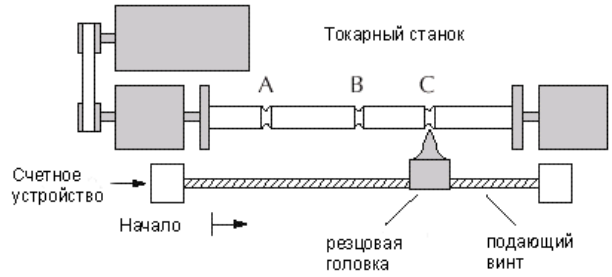
Пример программы 2: счетчик с заданными значениями

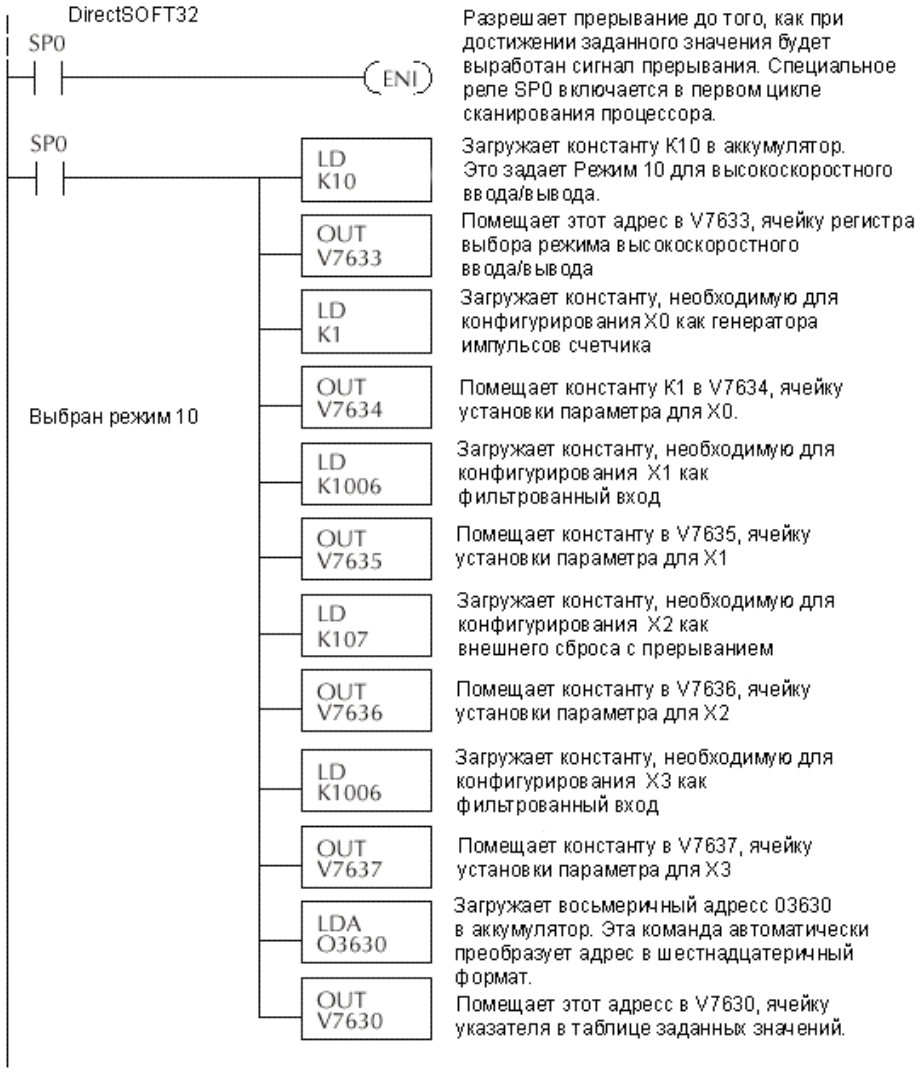
В следующей программе показывается, как запрограммировать схему высокоскоростного ввода/вывода для трех заданных значений. В ней рассматривается пример токарного станка, приведенный в начале данный главы. В приводимой программе показывается, как управлять резцовой головкой станка, чтобы сделать три проточки в обрабатываемом материале в точно указанных местах. При вращении подающего винта устройство счетчика генерирует импульсы, которые DL06 подсчитывает. Три заданные переменные A, B и C представляют положения (в виде числа импульсов), соответствующие каждой из трех проточек. В этом примере, используется только один высокоскоростной счетчик. Второй счетчик может использоваться таким же способом.

Заданные данные

A	V3630	0000	1500
B	V3632	0000	3780
C	V3634	0000	4850
	V3636	0000	FFFF

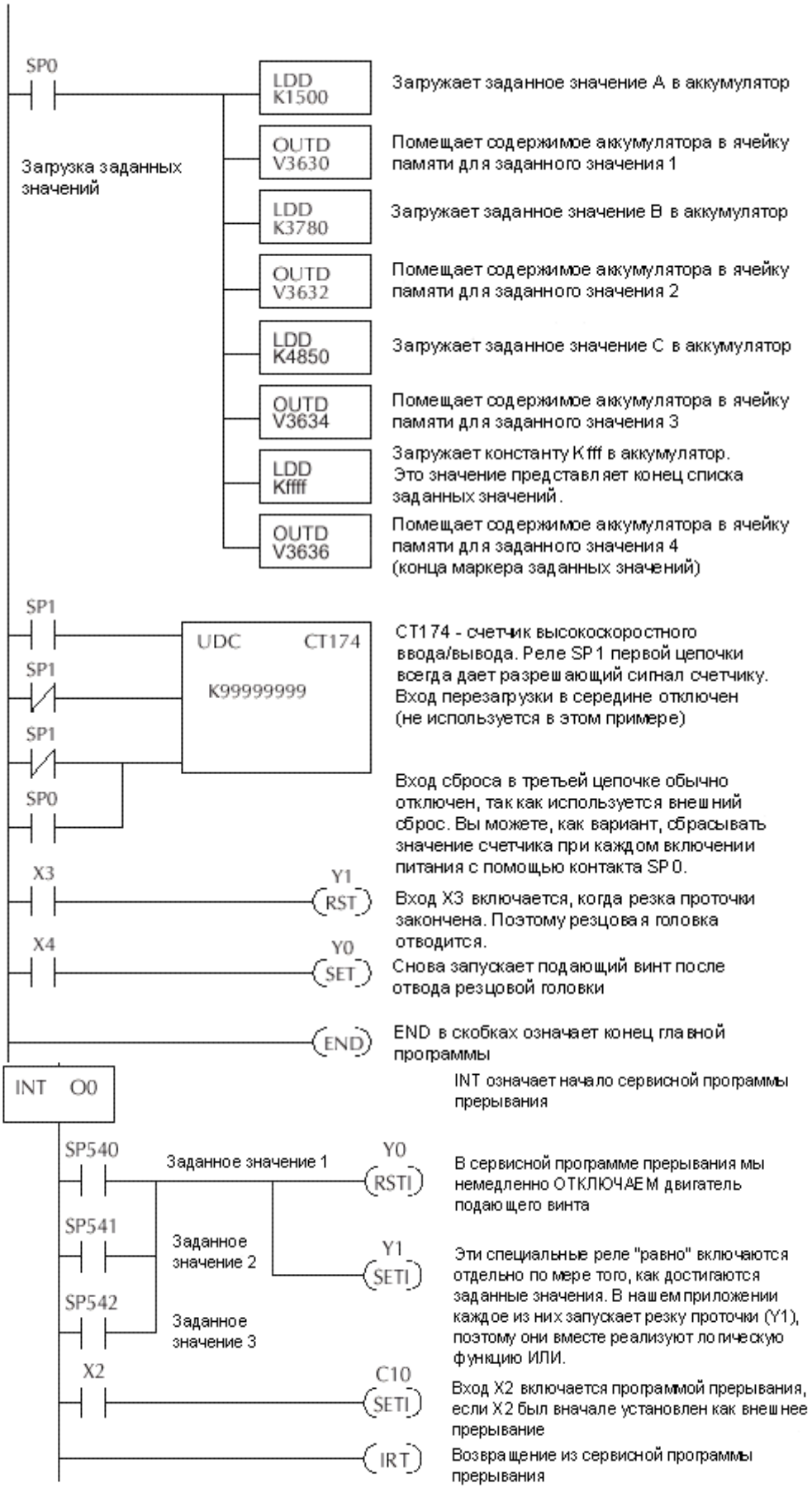
Обозначение ввода/вывода
X3 - резцовая головка, введенная
X4 - резцовая головка, отведенная
Y0 - двигатель подающего винта
Y1 - соленоид резцовой головки





продолжение на следующей странице

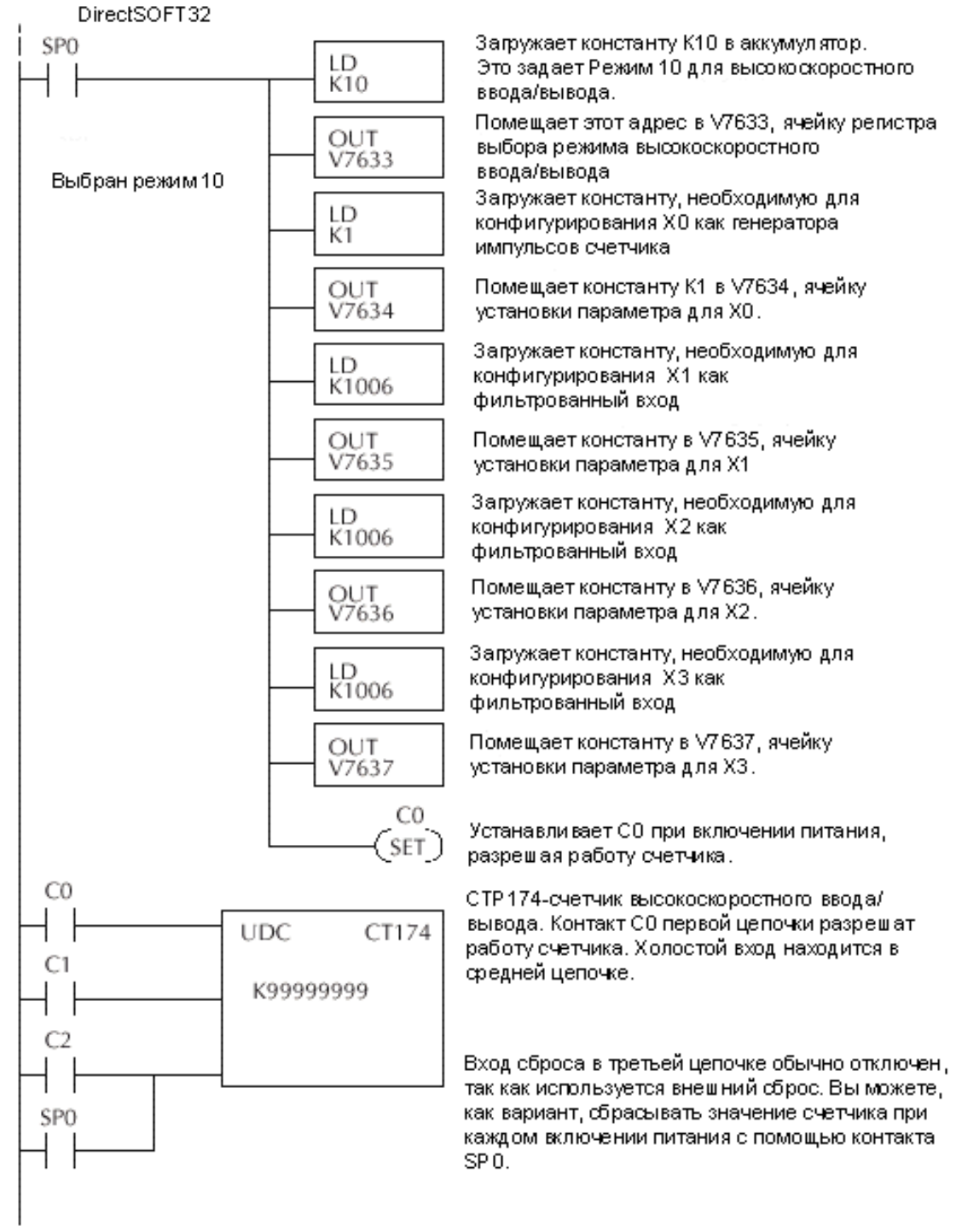
начало на предыдущей странице



В разных приложениях требуются различные типы действий при достижении каждого заданного значения. Программа прерывания имеет возможность различать события заданных значений, включая однозначно определенный выход для каждого контакта эквивалентного реле SPxxx. Можно определить источник прерывания при проверке конкретных контактов эквивалентных реле, а также X2. Контакт X2 включается (только программой прерывания), если прерывание было вызвано внешним сбросом, входом X2.

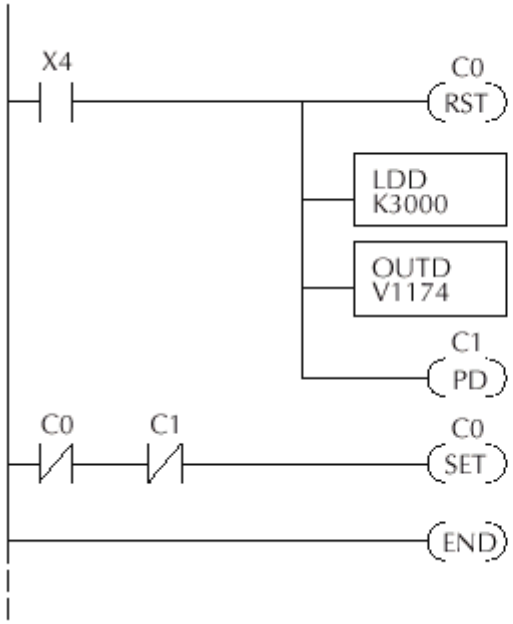
Пример программы 3: счетчик с перезагрузкой

В следующем примере программы показывается, как Вы можете перезагрузить текущее значение счетчика на другое значение. Когда вход команды перезагрузки (X4 в нашем примере) включается, происходит блокировка счета в счетчике от значения C0. Далее мы записываем значение K3000 в регистр счетчика (V1076 - V1077). Мы перегружаем текущее значение счетчика на K3000. После отключения команды перезагрузки (X4) счетчик продолжает считать импульсы, но начиная с K3000. В этом примере, используется только один высокоскоростной счетчик. Второй счетчик может использоваться таким же способом.



продолжение на следующей странице

Начало на следующей странице



Когда делается запрос на перезагрузку, пользователь включает X4. Вначале мы блокируем счет посредством сброса C0, входа разрешения работы счетчика.

Загружает двоично-десятичное значение K3000 в аккумулятор.

Помещает константу в V1174/V1175, ячейку накапливаемых отсчетов для CT76.

Генерирует входной импульс для перезагруженного счетчика, который вызывает перезагрузку счетчика из V1174-V1175.

Разрешает работу счетчика посредством установки C0, когда приходит импульс на перезагрузку на C1 (C1 отключен).

END в скобках означает конец главной программы.

Руководство по поиску неисправностей в режиме 10

Если при работе в режиме 10 у вас возникли неисправности, просмотрите следующие их признаки и возможные причины. Наиболее часто встречающие неисправности перечислены ниже.

Признак: счетчик не считает.

Возможные причины:

1. **Датчик или подключение.** Убедитесь, что кодировщик, бесконтактный переключатель или счетчик действительно включаются и состояние X0 (счетчик 1) и X2 (счетчик 2) высвечивается на светодиоде. Неисправность может быть из-за неправильного подсоединения приемников/источников тока и др. Не забудьте проверить подключение сигнала к заземлению. Убедитесь также, что длительность импульсов достаточна для распознавания их ПЛК.
2. **Конфигурация**. Используйте окно просмотра данных (Data View), чтобы проверить параметры конфигурации. Ячейка V7633 должна быть установлена в значение 10, а ячейка V7634 должна иметь значение 1 или 101 для разрешения первого высокоскоростного счетчика. Ячейка V7635 должна иметь значение 1 или 101 для разрешения второго высокоскоростного счетчика.
3. **Зависание сброса.** Проверьте входное состояние входа сброса X2 и X3. Если X2 включен, то счетчик не должен считать, так как он находится в состоянии сброса.
4. **Программа.** Убедитесь, что Вы используете счетчик CT174 и CT176 в вашей программе. Верхний вход является разрешающим сигналом для счетчика. Средний вход — это холостой вход. Нижний вход — сброс счетчика, он должен быть отключен при работе счетчика.

Признак: **счетчик работает, но заданные значения не функционируют.**

Возможные причины:

1. **Конфигурация.** Убедитесь, что заданные значения правильные. Заданными значениями являются 32-разрядные двоично-десятичные числа в пределах от 0 до 99999999. Убедитесь также в том, что все 32 бита записаны в зарезервированные ячейки с использованием команд LDD и OUTD. Используйте адреса только с четными номерами, от V3630 до V3767. Если Вы используете меньше 24 заданных значений, проверьте, что поместили «0000FFFF», «0000FF00» или «000000FF» в ячейку после последнего используемого заданного значения.
2. **Программа прерывания.** Используйте только прерывание #0. Убедитесь, что прерывание разрешено командой ENI , выполняемой до того, как появляется потребность в прерывании. Программа прерывания должна помещаться после главной программы, начинаться с метки INT и заканчиваться возвратом прерывания IRT.
3. **Специальные реле.** Проверьте номера специальных реле в вашей программе. Используйте SP540 для заданного значения 1, SP541 для заданного значения 2 и т. д. Не забудьте, что одновременно может быть включен контакт только одного специального реле «равно». Когда значение счетчика достигнет следующего заданного значения, контакт SP, который был включен в это время, выключится, а следующее реле включится.

Признак: счетчик правильно считает в прямом направлении, но не сбрасывается.

Возможные причины:

1. Проверьте состояние светодиода-индикатора X2 (счетчик 1) и X3 (счетчик 2), чтобы убедиться в том, что он активен тогда, когда вы ожидаете сброс. Или, если Вы используете внутренний сброс, примените режим просмотра состояний на DirectSOFT, чтобы отследить вход сброса в счетчик.

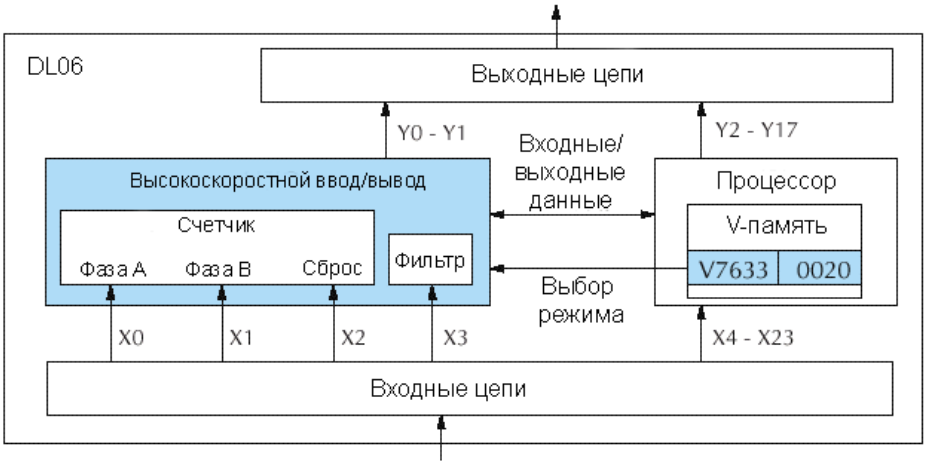
Режим 20: Реверсивный счетчик

Назначение

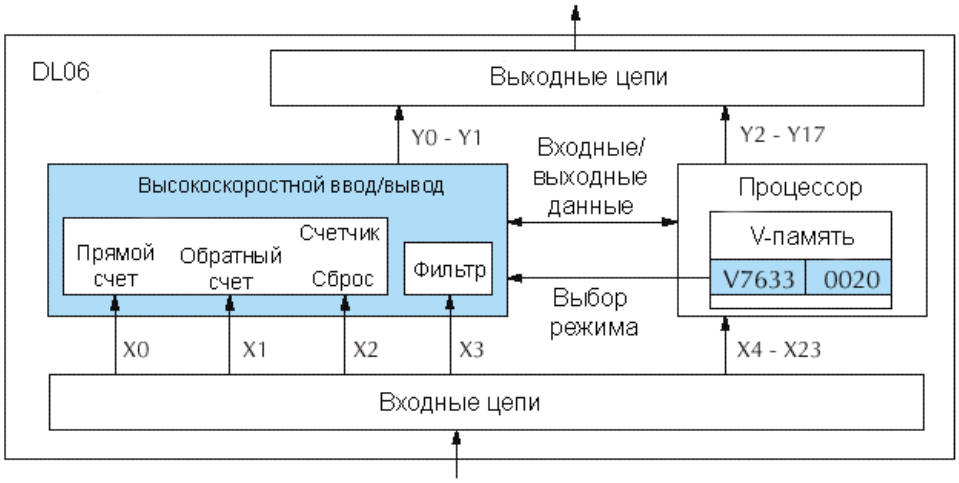
В схеме высокоскоростного ввода/вывода счетчик может считать реверсивные сигналы с двух отдельных источников (т.е. с двух отдельных каналов энкодеров) или два квадратурных импульсных сигнала. Квадратурные сигналы обычно вырабатываются инкрементальными энкодерами, которые могут быть вращательными или линейными. Реверсивный счетчик имеет диапазон от —8388608 до 8388607. Используя CT174 и CT175, квадратурный счетчик может считать со скоростью до 7КГц.

Функциональная блок-схема

Функциональная блок-схема, приведенная на рисунке ниже, показывает возможности высокоскоростного ввода/вывода в режиме 20. Если младший байт регистра режимов высокоскоростного ввода/вывода V7633 содержит двоично-десятичное число «20», то в схеме высокоскоростного ввода/вывода включается реверсивный счетчик. Для квадратурного счета вход X0 предназначается для квадратурного сигнала фазы A, а вход X1 получает сигнал фазы B. Вход X2 предназначается для сброса счетчика в нулевое значение, когда он включается.



Для стандартного реверсивного счета вход X0 предназначается для сигнала прямого счета и вход X2 для сигнала обратного счета. Вход X2 предназначен для сброса счетчика в нулевое значение, при включаении.

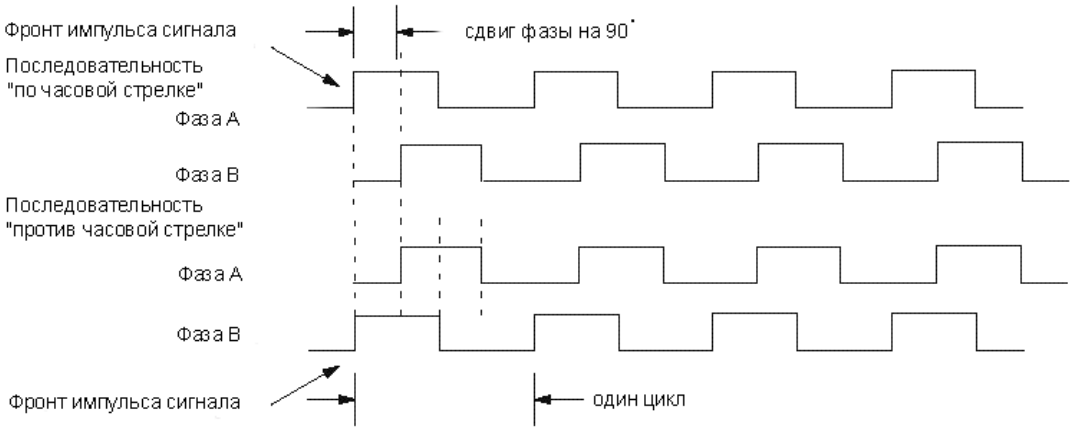


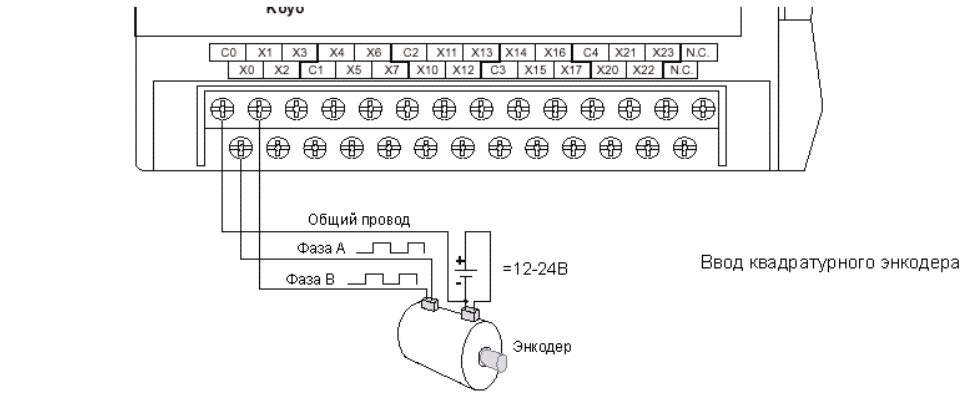
Квадратурные сигналы энкодера

Квадратурные сигналы энкодера содержат информацию о положении и направлении, в то время как их частота представляет скорость движения. Показанные ниже сигналы фазы A и B сдвинуты на 90 градусов, поэтому и появилось наименование «квадратурный». Когда фронт импульса фазы A предшествует фронту импульса фазы B (условно указывает движение по часовой стрелке), то счетчик высокоскоростного ввода/вывода ведет счет в прямом направлении. Если фронт импульса фазы B предшествует фронту импульса фазы A (условно указывает движение против часовой стрелки), то счетчик ведет счет в обратном направлении.

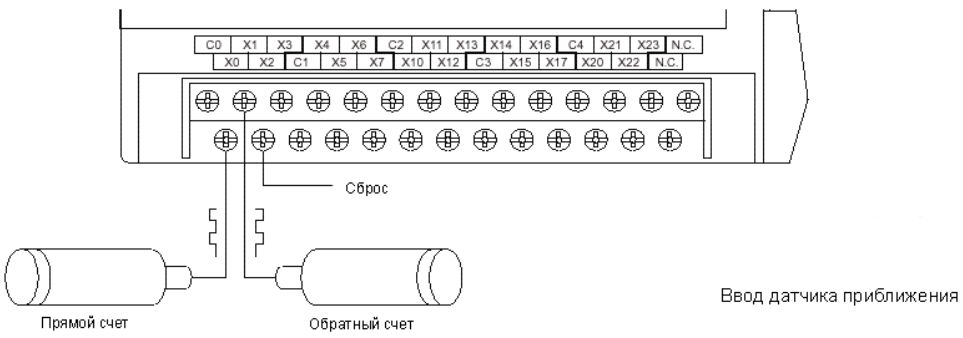
Схема подключения

Ниже показана общая схема подсоединения счетчиков/энкодеров к DL06 в режиме 20. Энкодеры с выходами — потребителями тока (транзисторы с открытым коллектором NPN), являются наилучшим выбором для сопряжения. Если энкодер является источником питания для входов, он должен иметь на выходе от 12 до 24 В постоянного тока. Обращаем внимание на то, что энкодеры с выходами 5В-источник, не будут работать с входами DL06.





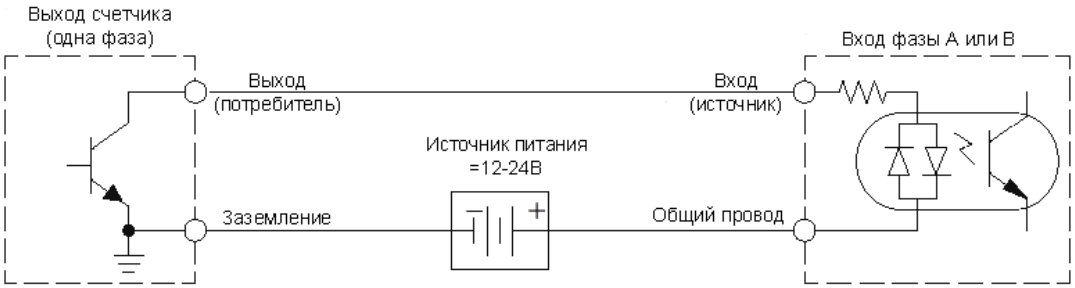
Ввод квадратурного энкодера



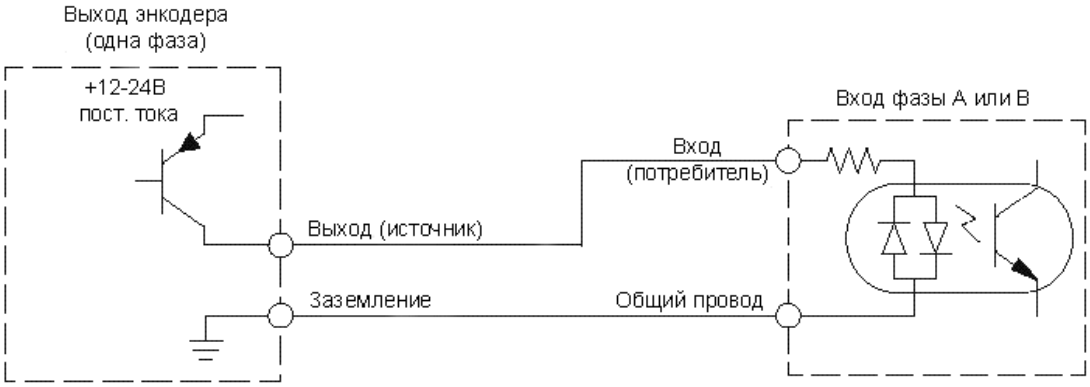
Ввод датчика приближения

Сопряжение с выходами энкодера

Входы постоянного тока DL06 являются гибкими в том отношении, что они определяют ток в любом направлении, поэтому они могут подсоединяться к счетчику как выходы-источники или как выходы-приемники тока. В следующей схеме энкодер имеет выходы транзистора с открытым коллектором NPN. Он принимает ток от входной точки ПЛК, которая является источником тока. Источником питания может быть FA-24PS или любой другой источник (на +12В или на +24В), удовлетворяющий техническим условиям входа.



В следующей схеме энкодер имеет выходы транзистора с открытым эмиттером PNP. Он является источником тока для входной точки ПЛК, которая обратным проводом соединена с заземлением. Поскольку кодировщик является источником тока, никакие дополнительные источники питания не требуются. Но следует отметить, что энкодер должен иметь от 12 до 24В (при выходе энкодера 5 В схема не будет работать).



Настройка на режим 20

Напоминаем, что ячейка V7633 является регистром выбора режимов высокоскоростного ввода/вывода. Используйте двоично-десятичное число «20» в младшем байте V7633 для выбора режима высокоскоростного счетчика.

Выберите из следующего списка наиболее удобный способ программирования ячейки V7633:

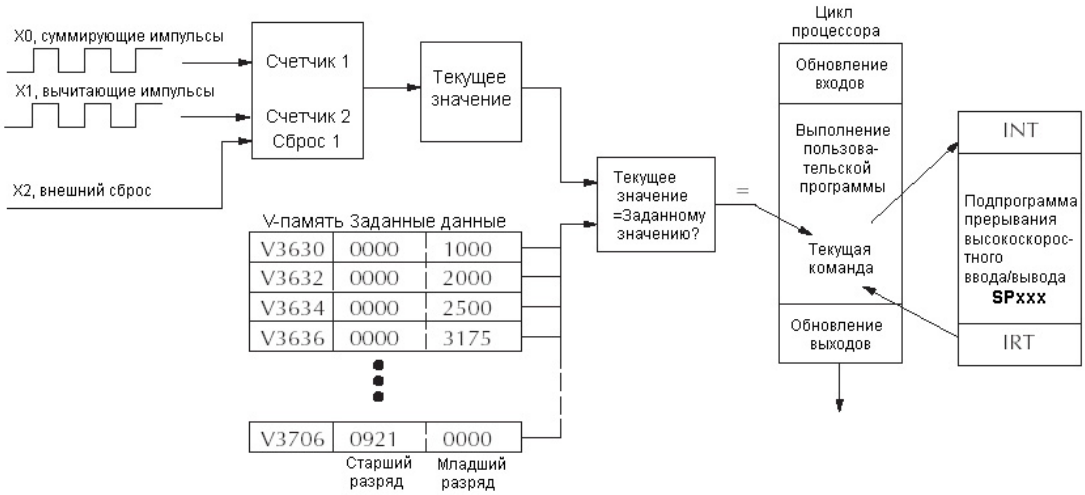
- Включить в свою программу команды LOAD и OUT.
- Использовать редактор памяти (Memory Editor)DirectSOFT.
- Использовать ручной программатор D2-HPP.

Мы рекомендуем использовать первый способ, при котором настройка HSIO становится неотъемлемой частью вашей прикладной программы. В примере программы, приведенном далее в данном разделе, показывается, как это делается.

Заданные значения и специальные реле

Цель предварительной установки состоит в том, чтобы выполнить определенное действие, когда при подсчете будет достигнуто заданное значение(Preset Value). Посмотрите рисунок ниже. Каждый счетчик может иметь до 24 предварительно заданных значений, которые могут быть запрограммированы. Пользователь выбирает предварительно установленные значения, и счетчик непрерывно сравнивает текущий счет с предварительно установленным. Когда они равны, контакты специального реле замыкаются, и происходит переход к выполнению подпрограммы прерывания.

Мы рекомендуем использовать специальное(ые) реле (SPxxx) в сервисной подпрограмме прерывания, чтобы обеспечить немедленное выполнение желательных действий. После завершения сервисной подпрограммы прерывания процессор возвращается в пользовательскую программу и продолжает ее выполнения с точки прерывания. Далее производится сравнение со следующим заданным значением.



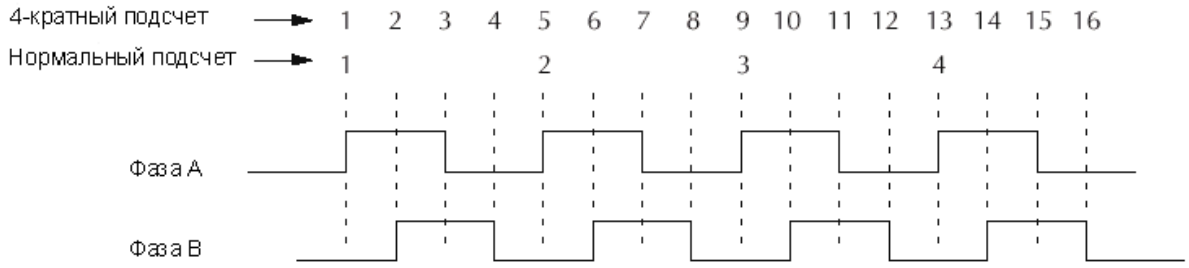
Конфигурирование входа X

Варианты конфигурируемого дискретного входа для режима высокоскоростного счетчика приведены в таблице ниже. В разделе по функционированию режима 60 в конце данной главы описывается программирование временных констант фильтра.

Режим 20 реверсивного счетчика

Вход	Регистр конфигурации	Функция	Требуемый шестнадцатеричный код
X0	V7634	Счет в прямом направлении	0202 (стандартный, абсолютный)
			0302 (стандартный, инкрементный)
		Фаза А	0002 (квадратурный, абсолютный) (по умолчанию)
			0102 (квадратурный, инкрементный)
			1002 (квадратурный, абсолютный) 4-кр подчет*
			1102 (квадратурный, инкрементный) 4-кр подчет*
X1	V7635	Счет в обратном направлении или фаза В	0000
X2	V7636	Сброс счетчика (без прерывания)	0007** (по умолчанию) 0207**
		Сброс счетчика (с прерыванием)	0107** 0307**
		Импульсный вход	0005
		Фильтрованный вход	xx06, (xx= время фильтрации от 0 до 99 мс (двоично-десятичных)
X3	V7637	Импульсный вход	0005
		Фильтрованный вход	xx06, xx= время фильтрации от 0 до 99 мс (двоично-десятичных) (по умолчанию)

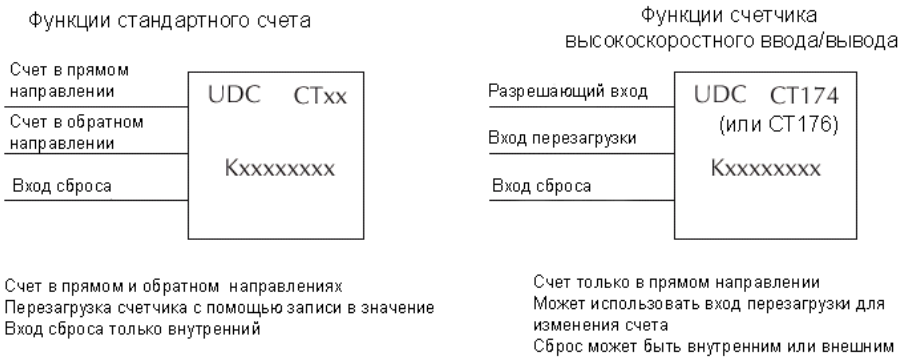
* - Счет в 4 раза больше с тем же самым энкодером.



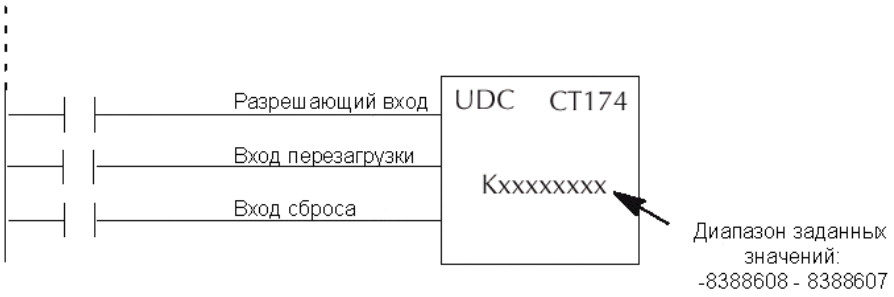
** - У счетчика Вы можете выбрать между нормальным сбросом и быстрым сбросом. Однако при быстром сбросе не опознаются изменения предварительно заданных значений сделанные во время выполнения программы. Когда «0007» или «0107» установлены в V7636 и заданные значения изменены во время выполнения программы, микроконтроллер DL06 распознает изменения во время сброса. Когда «0207» или «0307» установлены в V7636 процессор не проверяет изменения заданных значений, поэтому DL06 имеет более быстрое время сброса.

Написание управляющей программы

Счетчик имеет мнемоническое обозначение UDC (реверсивный счетчик). DL06 может иметь до 128 счетчиков с обозначениями от CT0 до CT177. Квадратурный счетчик в схеме высокоскоростного ввода/вывода доступен программе релейной логики при использовании только UDC CT174. Он занимает регистры счетчика CT174 и CT175 только тогда, когда режим 20 высокоскоростного ввода/вывода активен (иначе CT174 и CT175 доступен стандартному счетчику). Счетчик требует два регистра, так как он является счетчиком с двойным словом. Он имеет, как показано, три входа. Первый вход является разрешающим сигналом, средний вход предназначен для перезагрузки (записи), а нижний вход является сигналом сброса. Разрешающий вход должен быть отключен при перезагрузке.



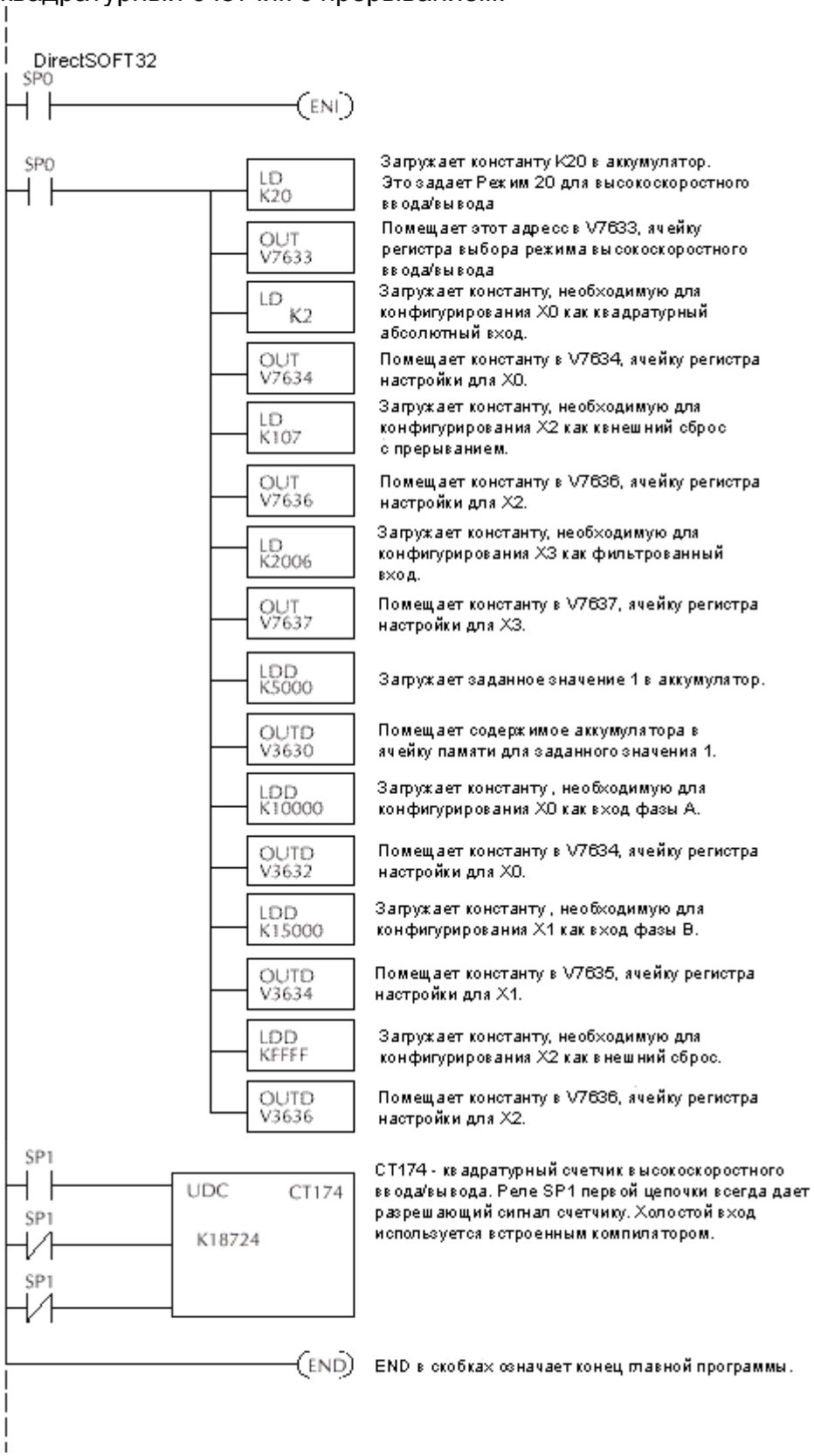
На следующем рисунке показано, как квадратурный счетчик может использоваться в пользовательской программе.



При включении разрешающего входа счетчик начнет реагировать на квадратурные импульсы X0 и X1, наращивая и уменьшая значение счетчика в CT174 – CT175. Контакты входа сброса работают в режиме логического ИЛИ (OR) с физическим входом сброса X2. Это означает, что квадратурный счетчик может получить сигнал сброса либо от контактов в цепочки программы, ИЛИ от внешнего сброса X2.

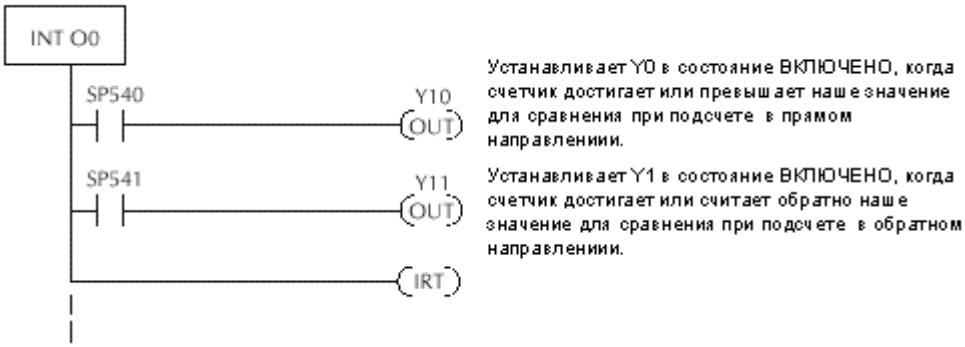
Пример программы 1: Квадратурный счетчик с прерыванием.

В примере программы показывано, как может быть запрограммирован квадратурный счетчик с прерыванием.



продолжение на следующей странице

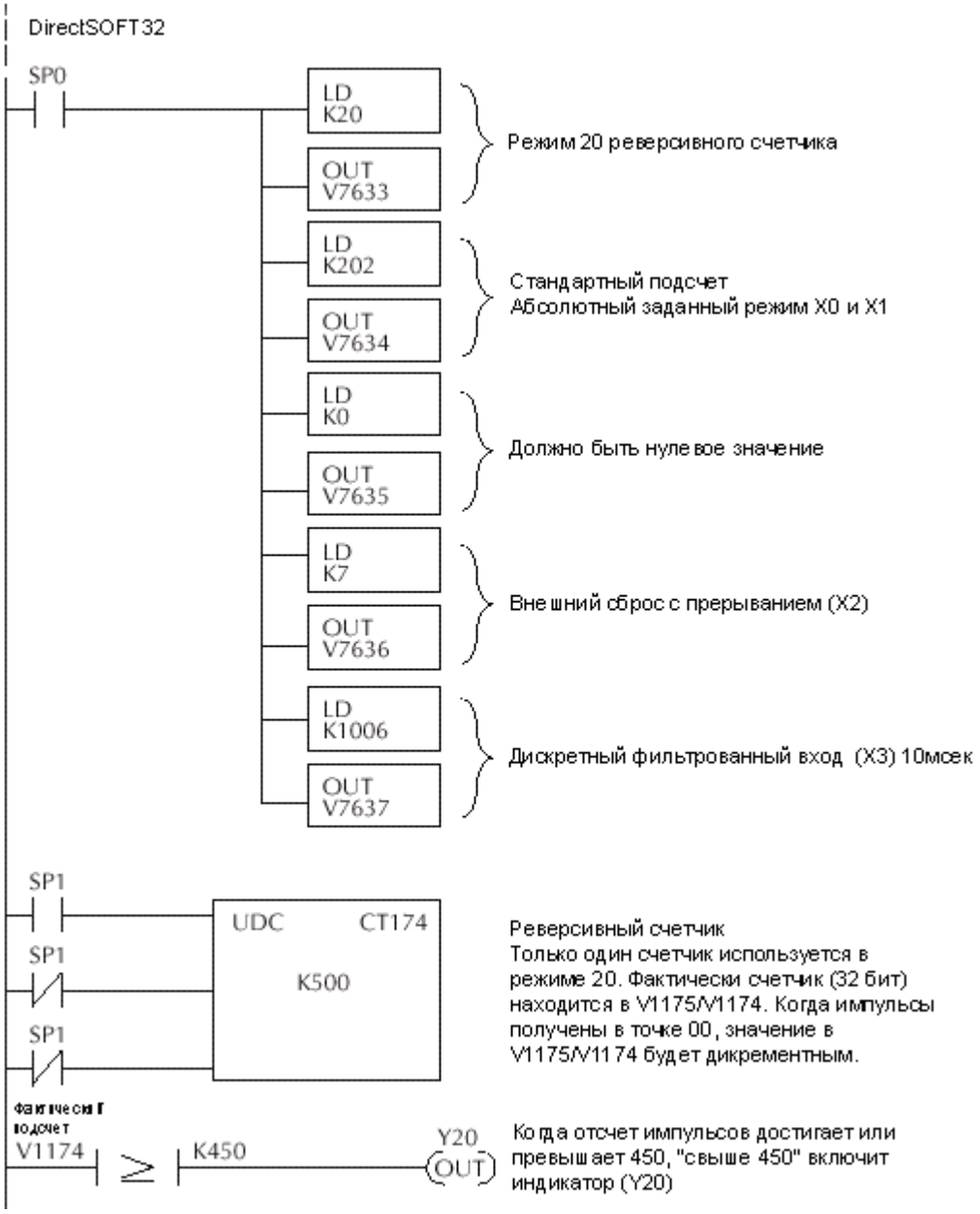
Начало на предыдущей странице



Команды Загрузка аккумулятора установили V-память, как требуется, т.е. 20 в V7633 для выбора режима и 0202 в V7634 для определения стандартного прямого/обратного направления с абсолютным заданным режимом. Помещая 0107 в V7636 возникает внешний сброс для счетчика CT174 и выполнится прерывание 0 при превышении границы сброса. Заданные значения прямого/обратного направления счета были сохранены в ячейках памяти с V3630 до V3635. Следующая ячейка имеет значение равное FFFF, чтобы указать, что мы не имеем больше предустановленных значений.

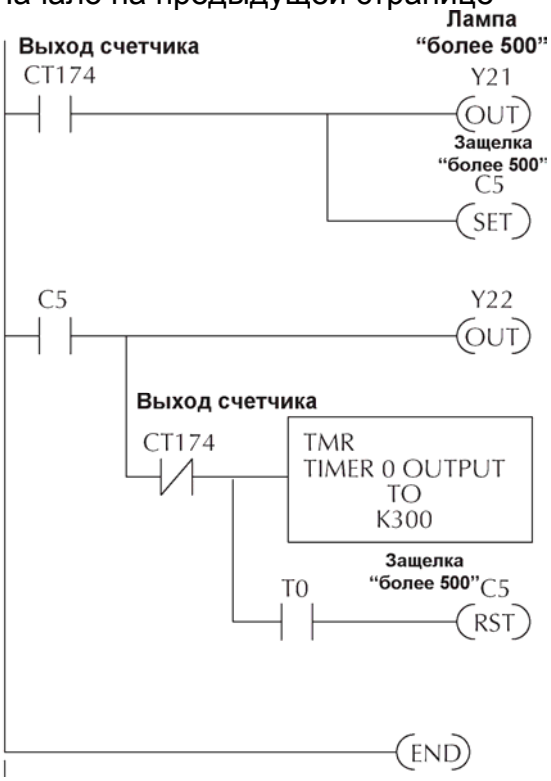
Пример программы 2: Реверсивный счетчик со стандартными входами.

В этом примере имеется лента конвеера “A”, которая транспортирует баллоны на участок контроля. Во время процесса, один датчик следит за баллонами, которые идут на ленту “A” для проверки, а другой датчик следит, сколько баллонов прошло на линию готовой продукции.
При достижении числа 500 баллонов, включается лампа “более 500”, и происходит перемаршрутизация входящего конвеера на ленту транспортера “B”. Данная перемашрутизация останется активной в течение 30 секунд после того, как лента транспортера “A” станет содержать менее 500 баллонов.
Программа ниже показывает, как написать на релейной логике требуемую обработку. Обратите внимание на использование V1174. Эта ячейка памяти сохраняет текущее значение счетчика CT174, который используется в DL06.



продолжение на следующей странице

Начало на предыдущей странице



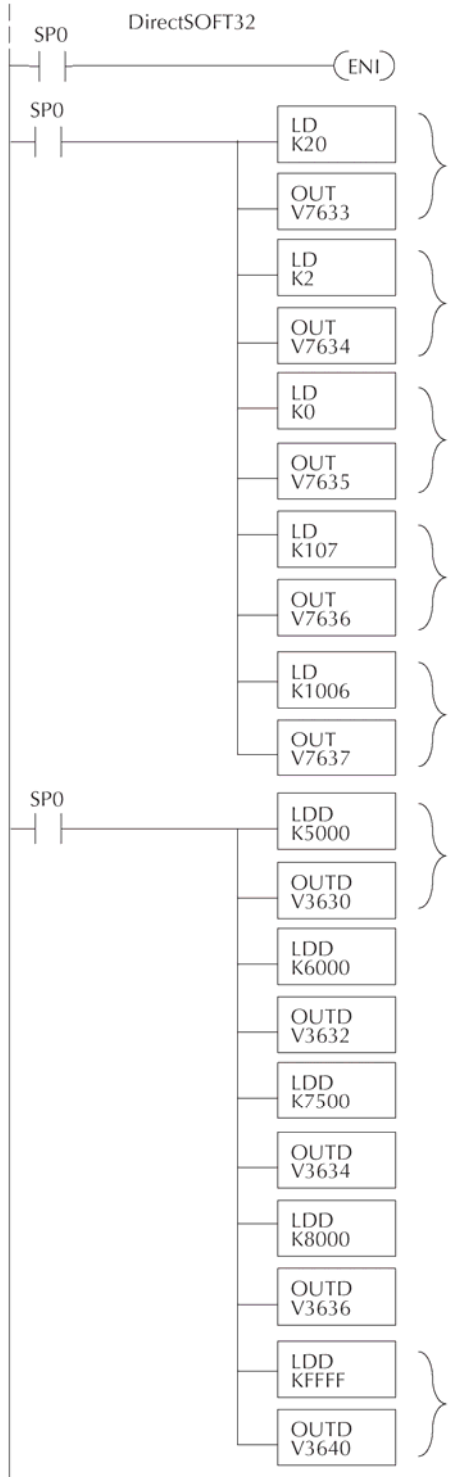
Когда счетчик импульсов достигает или превышает предустановленное значение 500, выход счетчика (CT174) устанавливается и включает лампочку "более 500" (Y21) и защелку C5.

Когда счетчик импульсов равен 500 или больше, выход Y22 включается и происходит перемаршрутизация и останется включенной в течение 30 секунд после снижения счетчика ниже 500.

END означает конец программы.

Пример программы 3: Квадратурный счетчик

В этом примере, деревянная заготовка сверлится с 3 отверстиями и затем отверстия наполняются клеем для шипов, которые будут вставлены на другой рабочей станции. Квадратурный энкодер подключен к координатному столу, который перемещает сверлильный станок горизонтально поверх заготовки.
Координатный стол останавливается, и сверлильный станок будет опускаться для сверления отверстия в соответствии с заданными координатами. После того, как три отверстия будут просверлены в заготовке, координатный стол совершает обратное движение для наполнения тех же самых отверстий клеем.



Разрешение прерываний

Режим 20

Квадратурный счет
абсолютный режим со входов X0 и X1

Должен быть 0

Внешняя перезагрузка по прерыванию (X2)

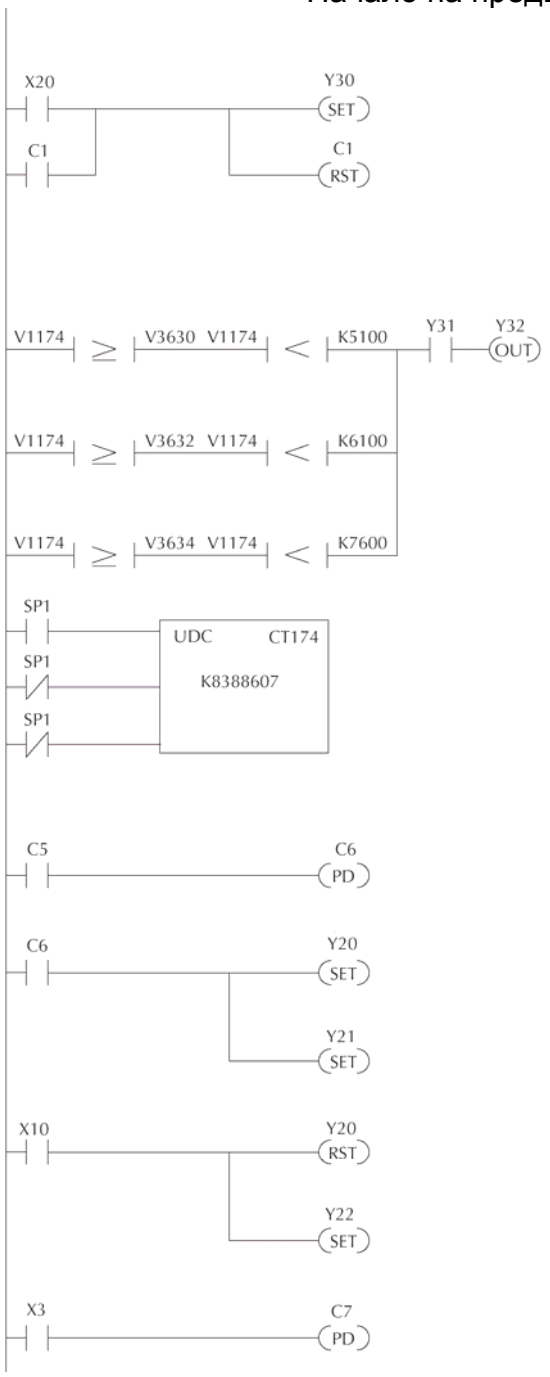
Дискретный фильтрованный ввод (X3)
10 мс.

Предустановка счетчика.
Это загрузка ячеек предустановленными значениями. В случае, если был выбран абсолютный предустановленный режим.

Сообщает процессору, что больше нет предустановленных значений.

ПРОДОЛЖЕНИЕ НА СЛЕДУЮЩЕЙ СТРАНИЦЕ.

Начало на предыдущей странице



Ручной старт сверления от кнопки (X20).

Перемещение координатного стола (Y30).
Завершенние последовательности сверления (только одно отверстие) (C1).

1-ое предустановленное значение счетчика(V1174).
Возвращение координатного стола(Y31).
Наполнение клеем (Y32).

2-ое предустановленное значение счетчика(V1174).

3-е предустановленное значение счетчика(V1174).

Инициирование последовательности сверления (C5)

Управляющий бит (C6)
Опускание дрели (Y20).

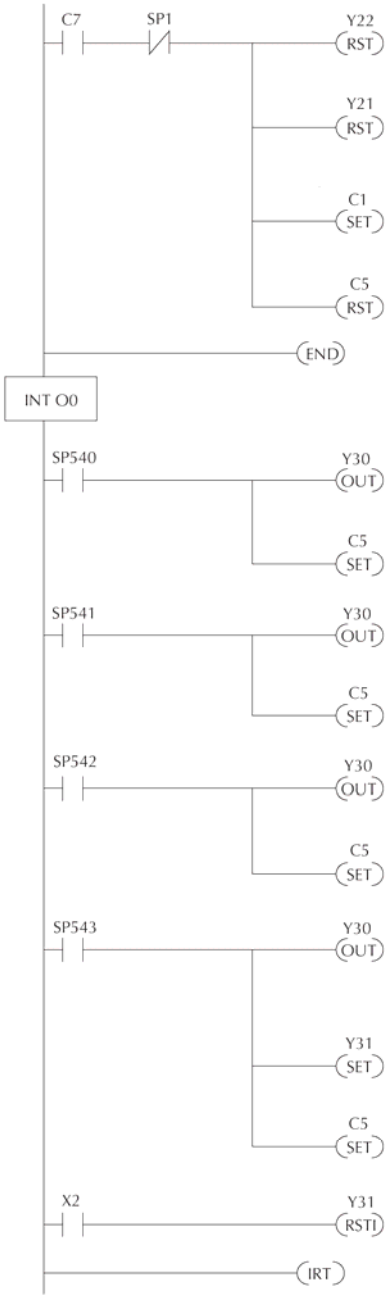
Сверление (Y21).

Сверление на полную глубину до срабатывания концевика 1 (X10).
Опускание дрели (Y20)/
Вращение дрели (Y22).

Дрель полность поднята (исходная позиция)
Концевик 2.
Управляющий бит C7

ПРОДОЛЖЕНИЕ НА СЛЕДУЮЩЕЙ СТРАНИЦЕ.

Начало на предыдущей странице



Управляющий бит C7.
Бит первого сканирования SP1.
Дрель опустить (Y22).

Вращение дрели (Y21).

Последовательность сверления закончена (только для одного отверстия) (C1).

Инициализация последовательности сверления (C5).

Следующая позиция координатного стола (Y30).
Инициализация последовательности сверления (C5).

Следующая позиция координатного стола (Y30).
Инициализация последовательности сверления (C5).

Следующая позиция координатного стола (Y30).
Инициализация последовательности сверления (C5).

Следующая позиция координатного стола (Y30).

Предыдущая позиция координатного стола (Y31).
Инициализация последовательности сверления (C5).
Координатный стол замкнул концевик исходного положения (X2).
Предыдущая позиция координатного стола (Y31).

Руководство по поиску неисправностей в режиме 20

Если при работе в режиме 20 у вас возникли неисправности, просмотрите следующие их признаки и возможные причины. Наиболее часто встречающие неисправности перечислены ниже.

Признак: счетчик не считает.

Возможные причины:

1. Датчик или подсоединение. Убедитесь, что входы энкодера или другого полевого устройства действительно включаются и состояние X0 и X1 высвечивается на светодиодах. Стандартный инкрементальный энкодер будет попеременно включать светодиоды для X0 и X1 при медленном вращении (1 оборот в мин). Неисправность может быть также из-за неправильного подсоединения приемников/источников тока и др. Не забудьте проверить подключение сигнала к заземлению. Убедитесь также, что длительность импульсов, рабочий цикл, уровень напряжения и частота находятся в пределах технических требований для входа.

2. Конфигурация. Убедитесь, что все параметры конфигурации установлены правильно. Ячейка V7633 должна быть установлена в значение 20, а ячейка V7634 должна иметь значение «0002» для разрешения входа фазы A, а ячейка V7635 должна иметь значение «0000» для разрешения входа фазы B.

3. Зависание сброса. Проверьте входное состояние входа сброса X2. Если X2 включен, то счетчик не должен считать, так как он находится в состоянии сброса.

4. Программа. Убедитесь, что Вы используете счетчик CT174 в Вашей программе. Верхний вход является разрешающим сигналом для счетчика. Он должен быть включен до того, как счетчик начнет счет. Средний вход — это холостой вход, он должен быть выключен при работе счетчика. Нижний вход — сброс счетчика, он должен быть также отключен при работе счетчика.

Признак: счетчик работает в неправильном направлении (производит прямой счет вместо обратного и наоборот).

Возможные причины:

1. Назначение каналов A и B. Возможно, что каналы A и B назначены для проводов энкодера неправильно (наоборот) относительно желательной ориентации угла поворота/счета. Стоит только переставить входы X0 и X1, чтобы направление счета восстановилось.

Признак: счетчик правильно считает в прямом и обратном направлениях, но не сбрасывается.

Возможные причины:

1. Проверьте состояние светодиода индикатора X2, чтобы убедиться в том, что он активен тогда, когда вы ожидаете сброс. Также проверьте, чтобы регистр конфигурации X2-V7636 был установлен в значение 7. Или, если Вы используете внутренний сброс, примените режим просмотра состояний на DirectSOFT32, чтобы отследить вход сброса счетчика.

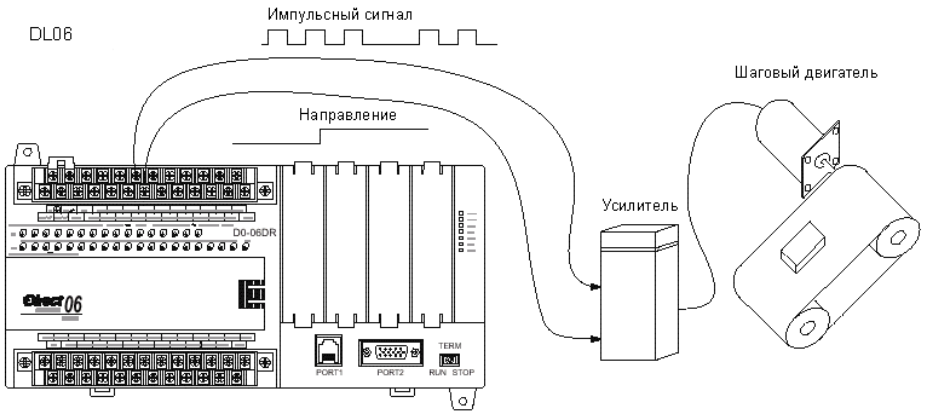
Режим 30: импульсный выход

Назначение

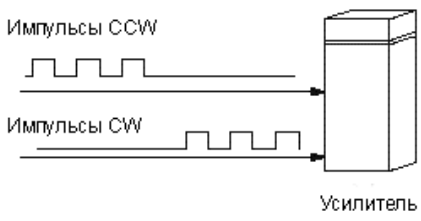
Схема высокоскоростного ввода/вывода в режиме 30 вырабатывает последовательность выходных импульсов, удобных для разомкнутых систем управления однонаправленным движением в системах позиционирования. Схема вырабатывает импульсы (шаговые приращения) и сигналы направления, которые Вы можете подсоединить к системам привода электродвигателя или к другим типам устройств управления движением. При использовании импульсного выхода режима 30 Вы можете выбрать один из трех профилей, детально описанных в данной главе:

- **Автоматический трапецеидальный.** От наклона ускорения к заданной скорости к наклону торможения.
- **Ступенчатый трапецеидальный**. Определяемый пользователем шаг ускорение/замедление и скорость .
- **Управление скоростью.** Только скорость и направление.

В режиме 30 схема высокоскоростного ввода/вывода становится высокоскоростным генератором импульсов (до 10КГц). Высокоскоростной ввод/вывод рассчитывает полный профиль движения по запрограммированным значениям ускорения и торможения, значениям положения и заданной скорости. На рисунке ниже показано, как DL06 генерирует импульсные сигналы и сигналы направления для усилителя привода системы шагового позиционирования. Импульсы вырабатываются в соответствии с профилем независимо и без прерывания выполнения пользовательской программы процессором.



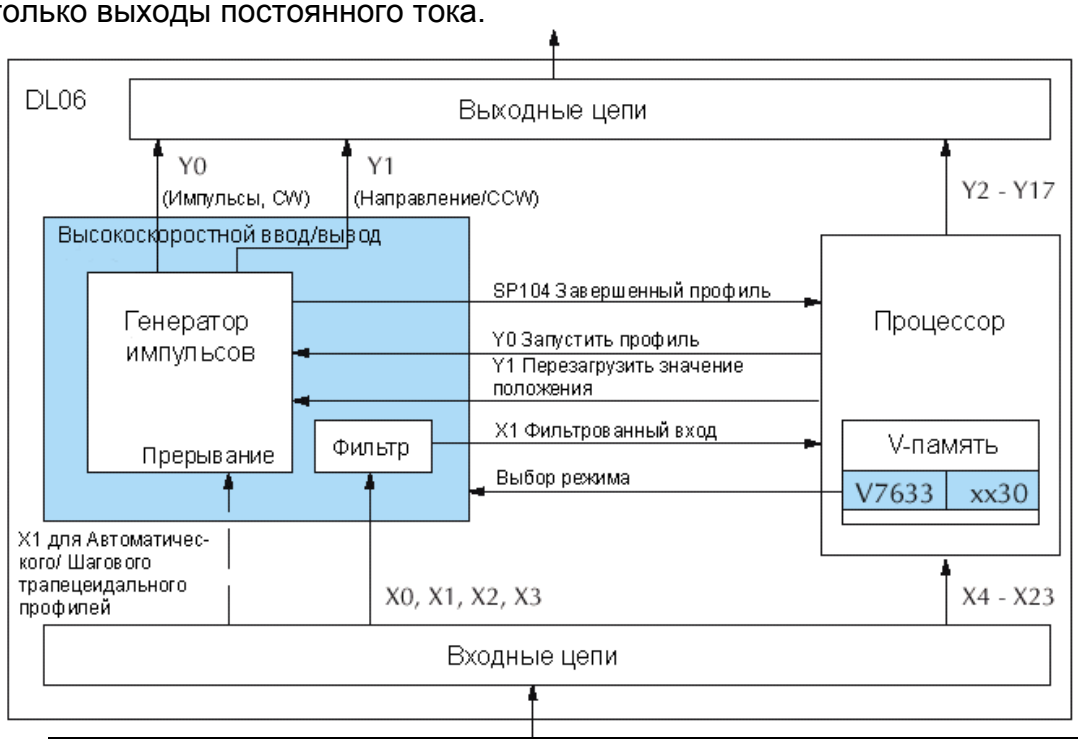
На рисунке выше DL06 вырабатывает сигналы импульсов и направления. Каждый импульс представляет собой наименьшее приращение движения в системе позиционирования (например, шага или микро-шага в шаговых системах). Как вариант, режим импульсного выхода может быть сконфигурирован для выдачи импульсных сигналов с поворотом против часовой стрелки (CCW) и с поворотом по часовой стрелке (CW), как показано справа.



ПРИМЕЧАНИЕ. *Импульсный выход предназначен для разомкнутых систем шаговых электродвигателей. Это обстоятельство, а также минимальная скорость — 40 импульсов в секунду делает его непригодным для управления сервомоторами.*

Функциональная блок-схема

На приведенной ниже блок-схеме показаны функциональные возможности высокоскоростного ввода/вывода в режиме 30. Если младший байт регистра режимов высокоскоростного ввода/вывода V7633 содержит двоично-десятичное число «30», то в схеме высокоскоростного ввода/вывода включаются функции импульсного выхода. Импульсные выходы используют клеммы Y0 и Y1 разъема выходов. Напоминаем, что так работать могут только выходы постоянного тока.



ВАЖНОЕ ПРИМЕЧАНИЕ. *В режиме импульсного выхода ссылки на Y0 и Y1 либо переопределяются, либо используются в двух разных смыслах. Физические ссылки относятся к клеммным зажимам, в то время как логические ссылки относятся к указанию входов/выходов в программе. Прочитайте, пожалуйста, следующие пункты, чтобы понять этот очень ответственный момент.*

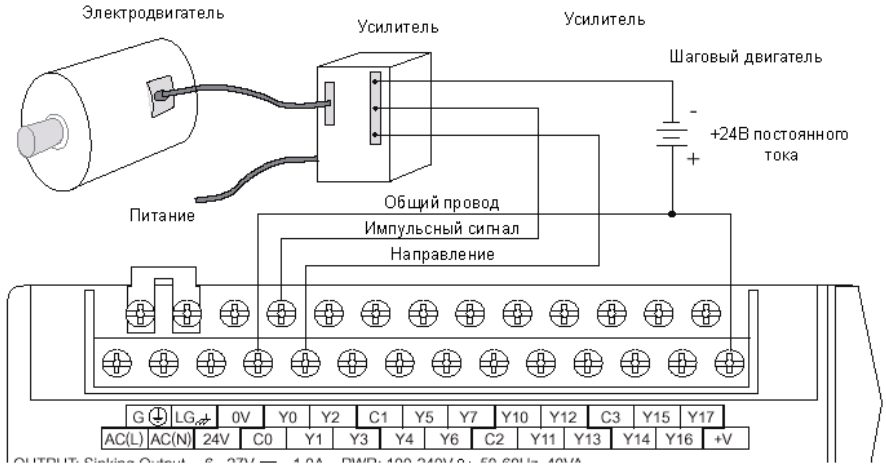
Назначение и использование входов/выходов в приведенных выше схемах:

- В режиме импульсного выхода X0, X1, X2 и X3 могут быть фильтрованными входами или импульсными входами, они доступны программе релейной логики как контакты входов.
- X1 работает как внешнее прерывание генератора импульсов для автоматического/ шагового трапецеидального профилей. В режимах других профилей он может использоваться как фильтрованный вход или импульсный вход, подобно X0 (конфигурация режима регистрации показана выше).
- Ссылки «Y0» и «Y1» используются в двух разных значениях. В разъеме дискретного выхода с клемм Y0 и Y1 подаются импульсы в систему управления движением. В программе релейной логики логические ссылки на Y0 и Y1 используются для инициализации функций высокоскоростного ввода/вывода в режиме 30 «Запустить профиль» и «Загрузить значение положения».

Надеемся, что это объясняет, почему ссылки на входы/выходы имеют двойное значение в режиме импульсного выхода. **Прочитайте данный раздел до конца**, чтобы исключить путаницу в действительных функциях входов/выходов.

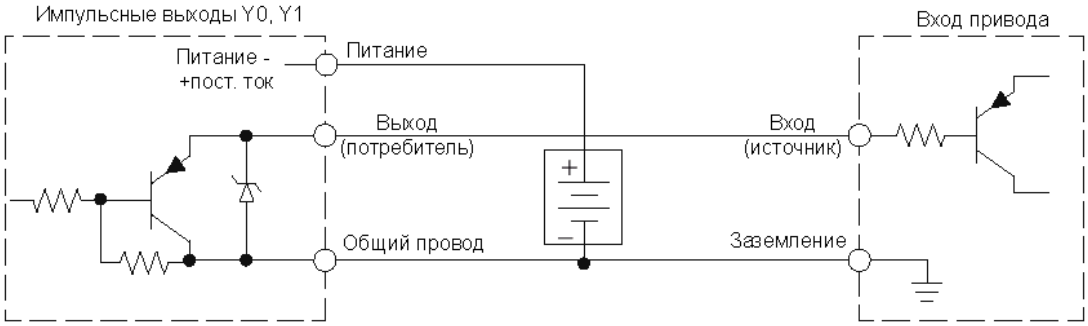
Схема подключения

Ниже показана обобщенная схема подсоединения импульсных выходов Y0 и Y1 к входам усилителя привода системы управления движением.

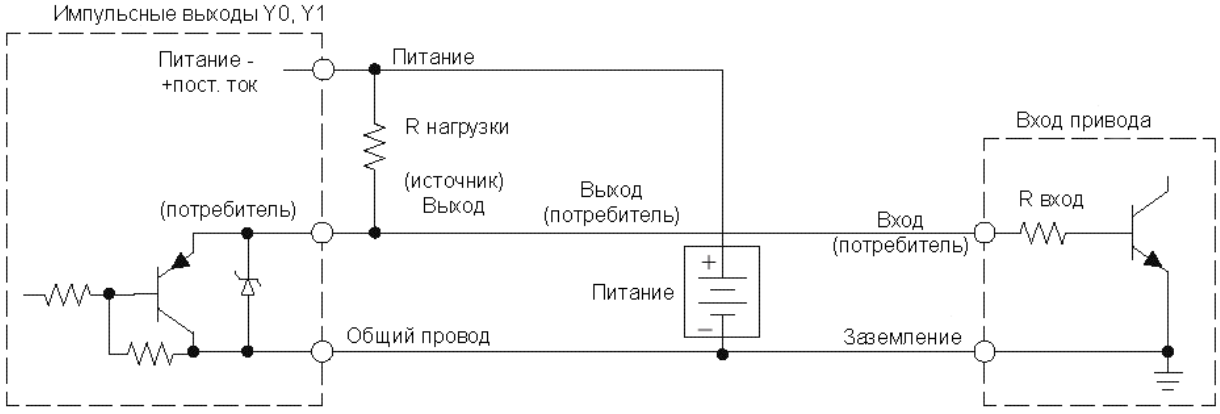


Сопряжение с выходами привода

Импульсные сигналы с выходов Y0 и Y1 обычно подводятся к входным схемам привода, как показано выше. Полезно расположить рядом эквивалентную принципиальную электрическую схему усилителя привода. На следующем рисунке показано, как выполняется подсоединение к выходной схеме привода — источника тока.



На следующей схеме показано, как выполняется подсоединение к выходной схеме привода — приемника тока с использованием нагрузочного резистора. Для рассчета и устанавки Rнагрузки смотрите главу 2.



Характеристики профилей движения

Профили управления движением, вырабатываемые в режиме импульсного выхода, имеют следующие характеристики :

Параметр	Характеристика
Профили	Автоматический трапецеидальный — наклон ускорения/заданная скорость/ наклон замедления
	Ступенчатый трапецеидальный — ступенчатое ускорение/замедление
	Управление скоростью - только скорость и направление
Диапазон положений	От - 8388608 до 8388607
Позиционирование	Абсолютная / относительная команда
Диапазон скоростей	От 40Гц до 10КГц
Регистры V-памяти	С V3630 по V3652 (Таблица параметров профиля)
Текущее положение	CT174 и CT175 (V1174 и V1175)

Конфигурирование физического ввода/вывода

Варианты конфигурируемого дискретного ввода/вывода для режима импульсного выхода приведены в таблице ниже. Процессор использует контакты SP 104 для определения «завершенного профиля». V7632 используется для выбора режимов импульс/направление или CCW/CW для импульсных входов. Вход X1 назначен как внешнее прерывание в режиме Совмещения.

Вход	Регистр конфигурации	Функция	Требуемый шестнадцатеричный код
-	V7632	Y0 = Импульсы Y1 = Направление	0103
		Y0 = Импульсы CW Y1 = Импульсы CCW	0003 (по умолчанию)
X0	V7634	импульсный вход	0005
		фильтрованный вход	xx06, xx = время фильтрации 0 -9 (двоично-десятичных) (по умолчанию)
X1	V7635	импульсный вход	0005
		фильтрованный вход	xx06, xx = время фильтрации 0 –99 (двоично-десятичных) (по умолчанию)
X2	V7636	импульсный вход	0005
		фильтрованный вход	xx06, xx = время фильтрации 0 –99 (двоично-десятичных) (по умолчанию)
X3	V7637	импульсный вход	0005
		фильтрованный вход	xx06, xx = время фильтрации 0 –99 (двоично-десятичных) (по умолчанию)

Функции логического ввода/вывода

Следующие логические ссылки ввода/вывода определяют функции, позволяющие высокоскоростному вводу/выводу взаимодействовать с программой релейной логики.

Логический вход/выход	Функция
SP 104	Профиль завершен: Высокоскоростной ввод/вывод включает в процессоре SP 104, когда профиль завершен. Выключает это реле, когда включается Начальный профиль (Y0).
X1	Внешнее прерывание: Если возможности прерывания выбраны для Автоматического трапецеидального профиля или шагового трапецеидального профиля, DI06 сохраняет импульсы вывода пока X1 не включится. После этого, контроллер начинает выводить импульсы определяющие заданную координаты.
Y0	Начальный профиль: программа релейной логики включает Y0 в начале движения. Если оно выключается до того, как движение завершено, движение прекращается. Его повторное включение будет запускать другой профиль, пока текущее положение не станет равным заданному.
Y1	Перезагрузка значения положения: Если движение остановлено, а начальный профиль выключен, Вы можете загрузить новое значение в CT174/CT175 и включить Y1. При этом значение в CT174/CT175 станет текущим положением.

Настройка для режима 30

Напоминаем, что ячейка V7633 является регистром выбора режимов высокоскоростного ввода/вывода. Используйте двоично-десятичное число «30» в младшем байте V7633 для выбора импульсного выхода.

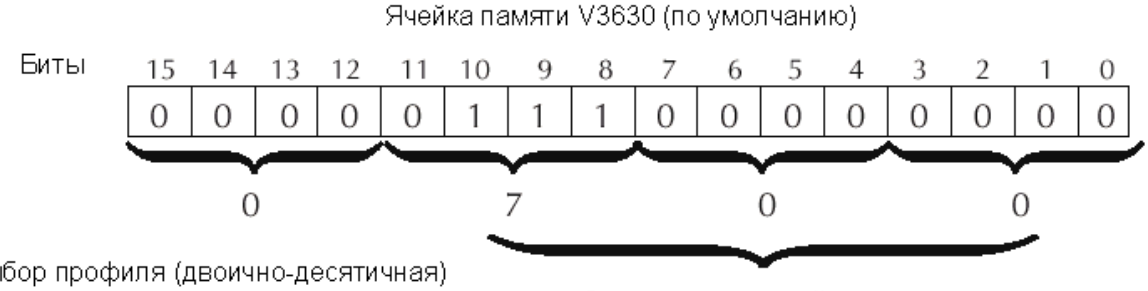
Смешанная установка (двоично-десятичная)
00 = Не используется (по умолчанию)
10 = Батарея включена
20 = Включенное питание в работе
30 = Батарея включена
и включенное питание в работе

Установка режима высокоскоростного ввода/вывода (двоично-десятичная)
30 = импульсный выход

Выберите из следующего списка наиболее удобный способ программирования ячейки V7633:

- Включить в свою программу команды LOAD и OUT.
- Использовать редактор памяти DirectSOFT.
- Использовать ручной программатор D2-HPP.

Мы рекомендуем использовать первый способ, при котором настройка высокоскоростного ввода/вывода становится неотъемлемой частью вашей прикладной программы. В примере программы, приведенном далее в данном разделе, показывается, как это делается.



Выбор профиля (двоично-десятичная)

с 4 до 7, с C до F = автоматический трапецеидальный профиль
0, 1, 8, 9 = шаговый трапецеидальный профиль
2 = профиль скорости

Значение заданной скорости

Диапазон = от 0, 4 до 999, представляющий частоту импульсов от 40Гц до 10КГц

Регистр выбора профиля/ скорости

В первой ячейке таблицы параметров профиля хранятся две ключевых части информации. Старшие четыре бита (12 - 15) задают тип требуемого профиля. В младших 12 битах (0 - 11) выбирается заданная скорость.

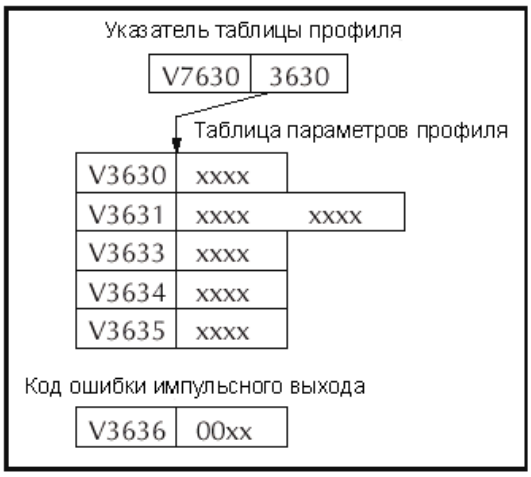
Это положение должно быть в программе релейной логики раньше, чем инициализация любого из трех профилей. Команды LD и OUT должны записывать все 16 битов, чтобы каждый раз обеспечивать полное определение четырехзначного значения (в двоично-десятичном формате) для регистров выбора Профиля/Скорости.

Выбор абсолютных и относительных значений определяется тем, как схема высокоскоростного ввода/вывода интерпретирует положение заданной координаты. Абсолютные координаты положения отсчитываются от нуля. Относительные координаты положения отсчитываются от текущего положения (предыдущей координаты положения). Вы можете выбрать более удобный для вашего приложения метод отсчета.

Таблица параметров профиля

V7630 — ячейка указателя начала Таблицы параметров профиля. По умолчанию начальной ячейкой для Таблицы параметров профиля является V3630. Однако Вы можете программным путем изменить в V7630 адрес этой ячейки. Не забудьте использовать команду LDA (загрузка адреса) для преобразования адреса из восьмеричного формата в шестнадцатеричный.

Высокоскоростной ввод/вывод использует регистр V-памяти, непосредственно следующий за таблицей параметров профиля, для указания ошибки профиля. Определение кодов ошибок см. в конце данного раздела.



Автоматический трапецеидальный профиль

V-память	Функция	Диапазон	Единицы
V3630, биты 12-15	Автоматический трапецеидальный профиль без конечной скорости (Конечная скорость установлена в 0.)	4 = абсолютный без прерывания 5 = абсолютный с прерыванием* C = относительный без прерывания D = относительный с прерыванием*	-
	Автоматический трапецеидальный профиль с конечной скоростью (используйте V3637 для установки конечной скорости)	6 = абсолютный без прерывания 7 = абсолютный с прерыванием* E = относительный без прерывания F = относительный с прерыванием*	-
V3630, биты 0-11	Заданная скорость	4 – 999 или 0 - 1000	x 10 импул./сек
V3631/V3632	Заданное положение**	-8388608 - 8388607	Импульсы
V3633	Начальная скорость	4 - 100	x 10 импул./сек
V3634	Время ускорения	1-100	x 100 мс
V3635	Время замедления	1-100	x 100 мс
V3636	Код ошибки	(смотрите в конце раздела)	-
V3637	Конечная скорость	4-100	x 10 импул./сек

* - Если Вами выбрано использование прерывания, то DL06 не начнет просмотр для Вашего заданного подсчета пока прерывание X1 не включено.

** - Чтобы установить отрицательное число, поместите 8 в наиболее значимую цифру. Например: -8388608 это 88388608 в V3631 и V3632.

Ступенчатый трапецеидальный профиль

V-память	Функция	Диапазон	Единицы
V3630, биты 12 - 15	Ступенчатый трапецеидальный профиль	0 = абсолютный без прерывания 7 = абсолютный с прерыванием* 8 = относительный без прерывания 9 = относительный с прерыванием*	-
V3630, биты 0 - 11	Заданная скорость	4 – 999 или 0 для 1000	x 10 импульсов/сек
V3631/ 3632	Заданное положение**	- 8388608 -8388607	Импульсы
V3633	Шаг 1 Ускорение	4-1000	x 10 импульсов/сек
V3634	Шаг 1 Расстояние	1-9999	Импульсы
V3635	Шаг 2 Ускорение	4-1000	x 10 импульсов/сек
V3636	Шаг 2 Расстояние	1-9999	Импульсы
V3637	Шаг 3 Ускорение	4-1000	x 10 импульсов/сек
V3640	Шаг 3 Расстояние	1-9999	Импульсы
V3641	Шаг 4 Ускорение	4-1000	x 10 импульсов/сек
V3642	Шаг 4 Расстояние	1-9999	Импульсы
V3643	Шаг 5 Замедление	4-1000	x 10 импульсов/сек
V3644	Шаг 5 Расстояние	1-9999	Импульсы
V3645	Шаг 6 Замедление	4-1000	x 10 импульсов/сек
V3646	Шаг 6 Расстояние	1-9999	Импульсы
V3647	Шаг 7 Замедление	4-1000	x 10 импульсов/сек
V3650	Шаг 7 Расстояние	1-9999	Импульсы
V3651	Шаг 8 Замедление	4-1000	x 10 импульсов/сек
V3652	Шаг 8 Расстояние	1-9999	Импульсы

* - Если Вами выбрано использование прерывания, DL06 не начнет просмотр для Вашего заданного подсчета пока прерывание X1 не включено.

** - Чтобы установить отрицательное число, поместите 8 в наиболее значимую цифру. Например: -8388608 это 88388608 в V3631 и V3632.

Управление скоростью

V-память	Функция	Диапазон	Единицы
V3630	Профиль скорости	только 2000	-
V3631/ 3632	Выбор направления	80000000 = CCW 0 = CW	Импульсы
V3633	Скорость	4-1000	x 10 импульсов/сек
V3636	Код ошибки	(см. в конце раздела)	-

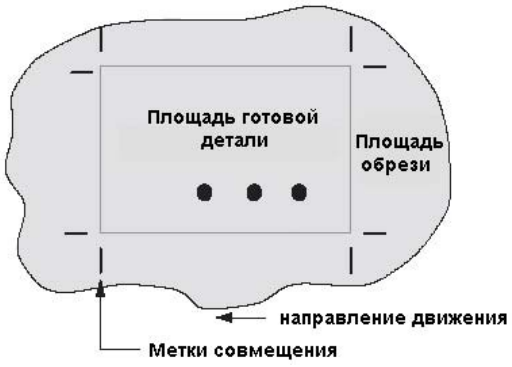
Выбор типа профиля

В режиме импульсного выхода вырабатывается три профиля движения. В большинстве приложений применяется один тип для многих шагов движения. Однако при необходимости можно каждый шаг сделать разным.

- Автоматический трапецеидальный. От наклона ускорения — к заданной скорости — к Наклону замедления.
- Ступенчатый трапецеидальный. От скорости — к управлению положением по прерыванию.
- Управление скоростью. Только скорость и направление.

Определение автоматического трапецеидального профиля

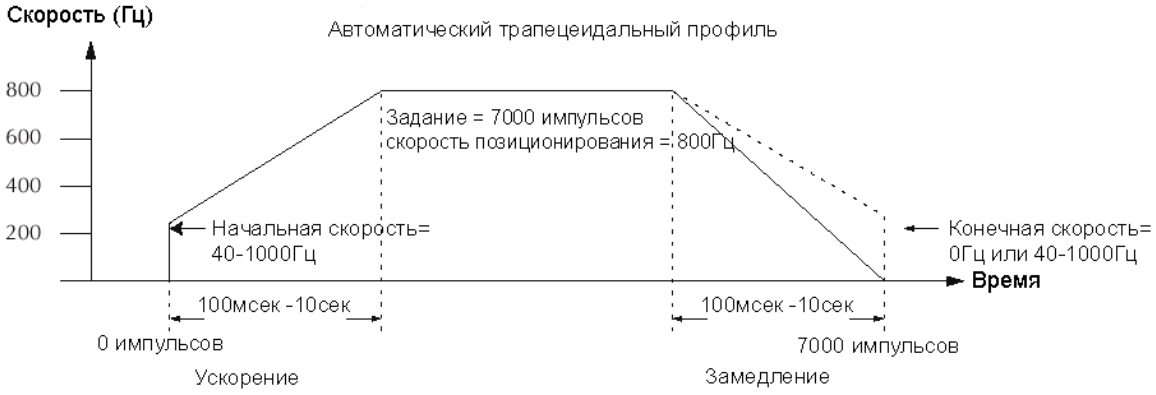
Автоматический трапецеидальный профиль — наиболее часто используемый профиль при позиционировании. Он описывает передвижение груза в заранее заданное конечное положение с помощью созданного профиля шагов движения.



С помощью профилей совмещения решается определенный класс задач движения. В некоторых приложениях материал обрабатываемого изделия перемещается мимо рабочего инструмента, например, сверлильного устройства. Как показано справа, метки совмещения на обрабатываемом материале позволяют совместить положение машинного инструмента по отношению к прямоугольнику готовой детали, чтобы просверлить ее в надлежащем месте.

Движения при поиске исходного положения позволяют разомкнутым системам управления движением точно определять (и загружать) текущее положение при запуске системы.

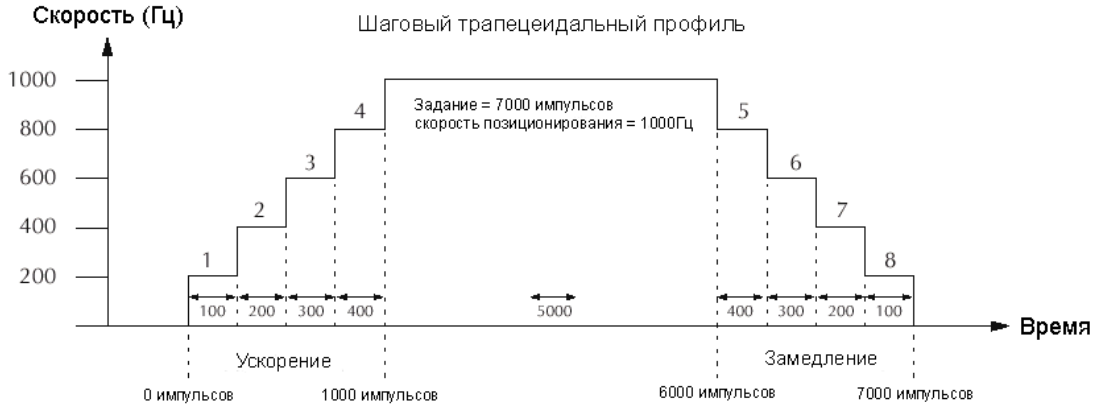
Профили Совмещения сочетают режимы управления скоростью и положением. Движение начинается с ускорения до запрограммированной скорости. Эта скорость поддерживается, а движение продолжается неопределенное время.



Пользователь определяет начальную скорость, время ускорения и торможения и число импульсов. Процессор вычисляет профиль с этими входными величинами.

Определение ступенчатого трапецеидального профиля

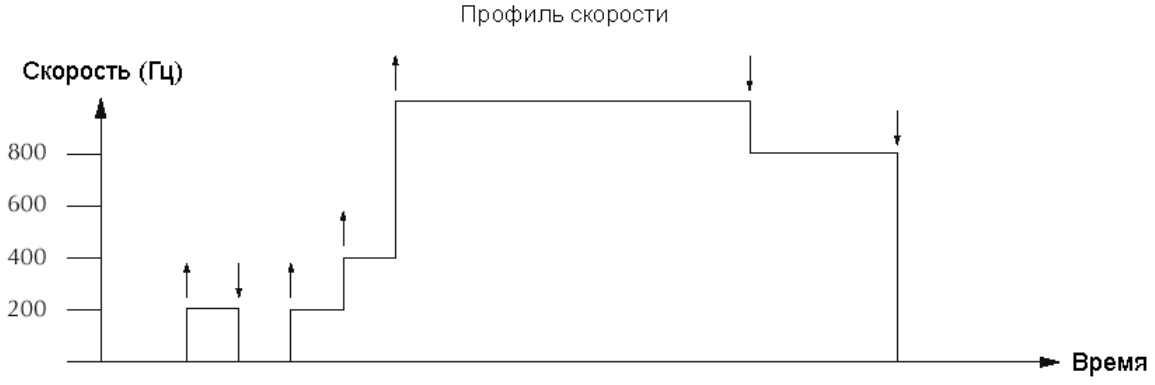
Ступенчатый трапецеидальный профиль сочетает режимы управления скоростью и положением. Движение начинается с ускорения до запрограммированной скорости. Эта скорость поддерживается, а движение продолжается неопределенное время. Когда появляется внешний сигнал прерывания (при распознавании совмещения), в профиле происходит переключение от управления скоростью к управлению положением. Такое движение заканчивается на предварительно заданном расстоянии от точки прерывания (например, позиции для сверления). Дальнейшее движение продолжается с линейным замедлением перед заданным конечным положением.



Определите шаги с 1 по 4 для постепенного ускорения к заданной скорости и шаги с 5 по 8 для постепенного торможения от заданной скорости. Этот тип профиля используется для управления большими шаговыми двигатели и/или большими инерциальными нагрузками. Он может, однако, использоваться для обеспечения постепенного разгона и торможения в приложениях, содержащих небольшие двигатели и нагрузки.

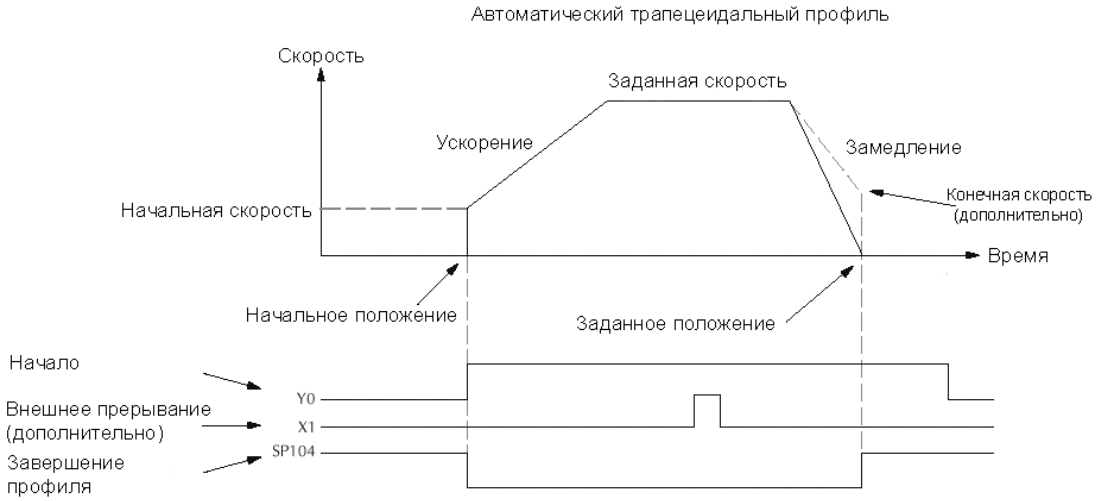
Определение профиля управления скоростью

В профиле управления скоростью управление выполняется только для направления и скорости движения. Здесь не определяется заданное конечное положение, поэтому движение имеет неопределенную продолжительность. Требуется задать только значение начальной скорости. Далее значение скорости вырабатывается в процессе движения. Стрелки в профиле указывают изменение скорости.



Работа с автоматическим трапецеидальным профилем

Начальные скорости должны записываться в диапазоне от 40 до 1000 импульсов/сек. Остальные параметры приводятся в таблице профиля.



Ниже графика профиля на временной оси представлен график состояния сигналов, который указывает порядок событий. В HSIO используется логический выход Y0 как начальный вход в HSIO, с которого начинается профиль. Немедленно HSIO включает сигнал Завершения профиля (SP104), таким образом, программа релейной логики может отслеживать процесс движения. Обычно программа релейной логики отслеживает этот бит, поэтому она знает, когда инициировать следующий шаг движения.

Вы также можете использовать внешнее прерывание (X1). Если выбран режим внешнего прерывание для профиля, то DL06 выполнит вывод импульсов только после включения включения X1. Затем DL06 выводит импульсы, определенные как заданное значение.

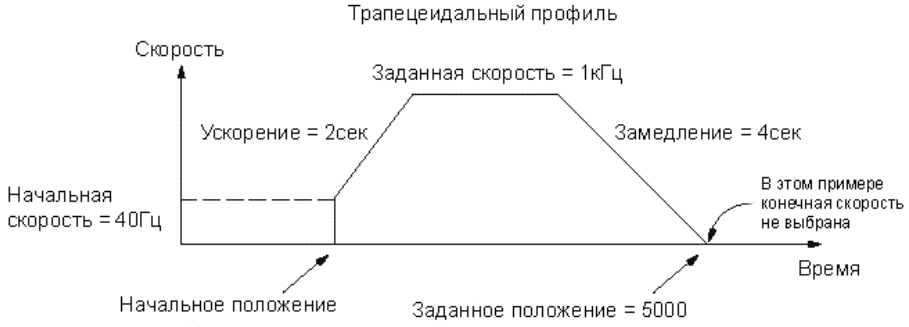
Если Вы знакомы с управлением движением, то можете заметить, что мы не определяем направление движения. Функция высокоскоростного ввода/вывода отслеживает заданное конечное положение относительно текущего положения и автоматически выдает приводу электродвигателя информацию о точном направлении.

Следует отметить, что в нашем графике движение немедленно ускоряется до начальной скорости. Этот отрезок важен для шаговых систем, поэтому мы не будем рассматривать область низких скоростей, когда малые моменты или точки резонанса могут привести к остановке. (При остановках шаговых электродвигателей теряется положение рабочего органа в незамкнутых системах позиционирования). Однако предпочтительнее не использовать слишком большие начальные скорости, так как шаговый электродвигатель может, кроме того, «проскакивать» некоторые импульсы из-за инерции системы. Вы можете также установить конечную скорость по той же самой причине.

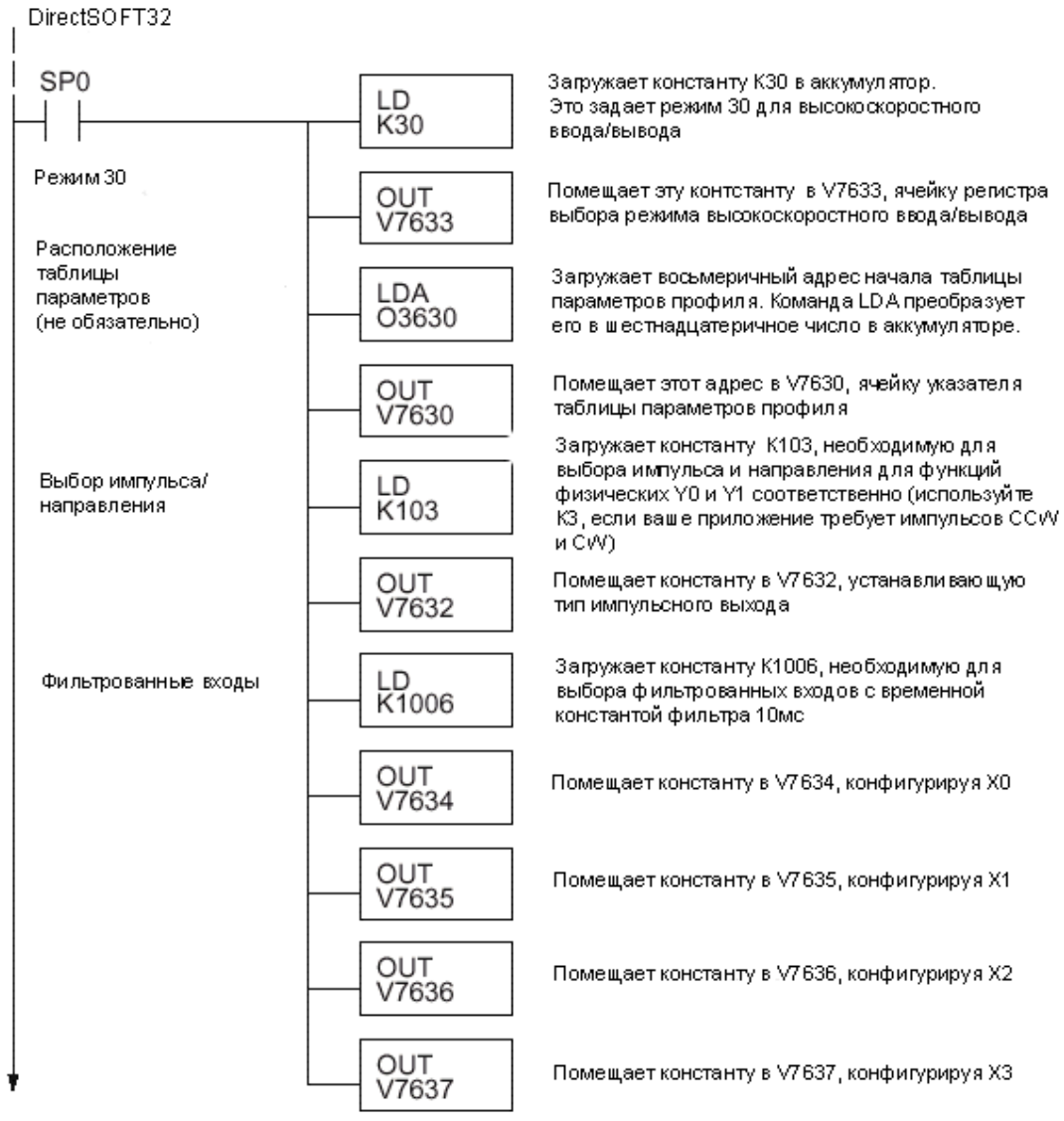
Если Вам необходимо изменить значение текущего положения, используйте обмотку логического выхода Y1 для загрузки нового значения в счетчик высокоскоросного ввода/вывода. Если программа релейной логики загружает новое значение в CT174/CT175 (V1174/V1175), то при включении Y1 это значение будет скопировано в счетчик схемы HSIO. Это должно быть сделано до начала отработки профиля, так как HSIO игнорирует Y1 при движении.

Пример 1: Автоматический трапецеидальный профиль без внешнего прерывания

Автоматический трапецеидальный профиль, который мы хотим реализовать, показан и размечен на следующем рисунке. Он включает ненулевую начальную скорость и среднюю заданную скорость.

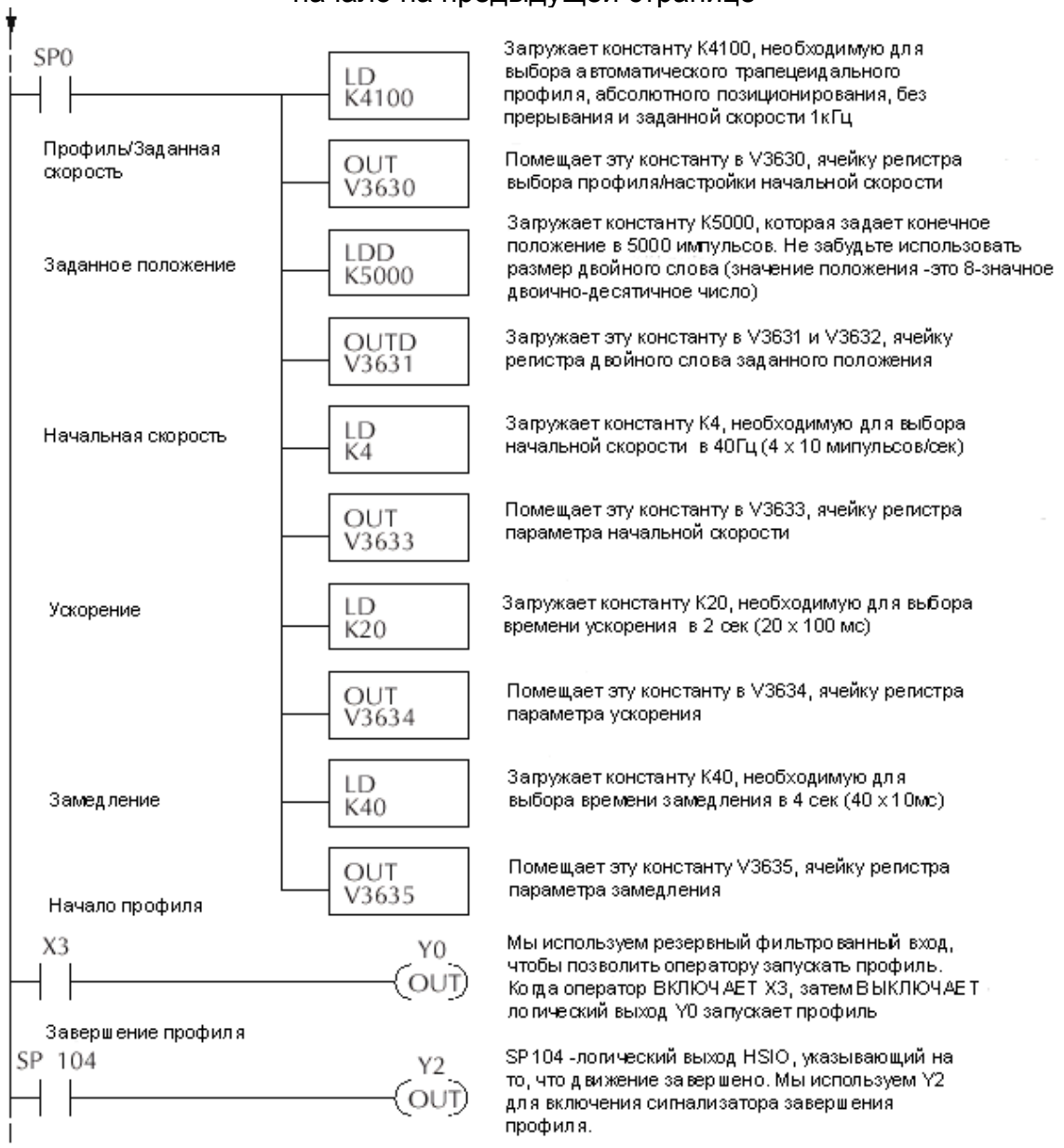


В следующей программе реализуется данный профиль. В начале программы вводятся все необходимые параметры настройки для режима 30 Импульсного выхода. Это должно выполняться в программе один раз, поэтому мы используем контакт первого сканирования SP0, чтобы запустить настройку



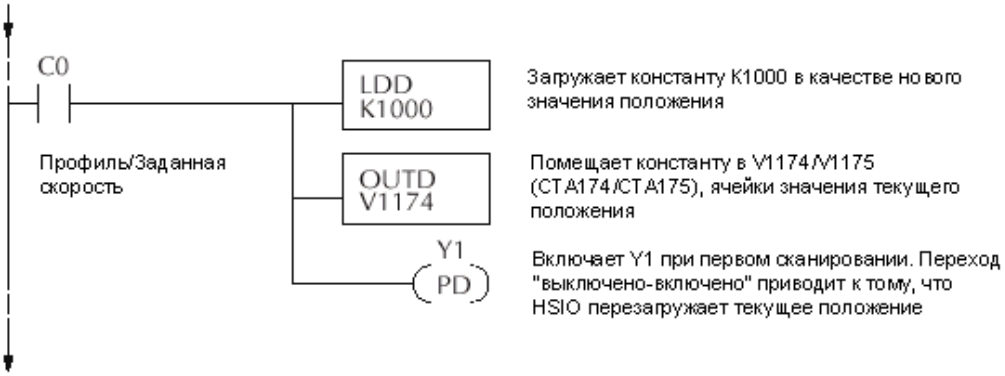
продолжение на следующей странице

начало на предыдущей странице



Перезагрузка значения положения

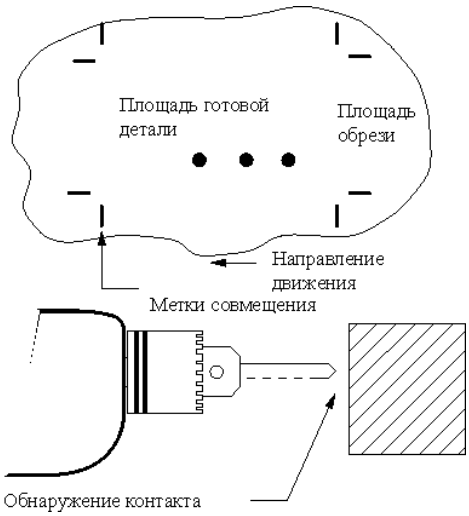
В любой момент Вы можете записать (перезагрузить) новое положение в значение текущего положения. Это часто делается после поиска исходного положения (см. примеры программ в режиме совмещения).



Пример 2: Автоматический трапецеидальный профиль с внешним прерыванием

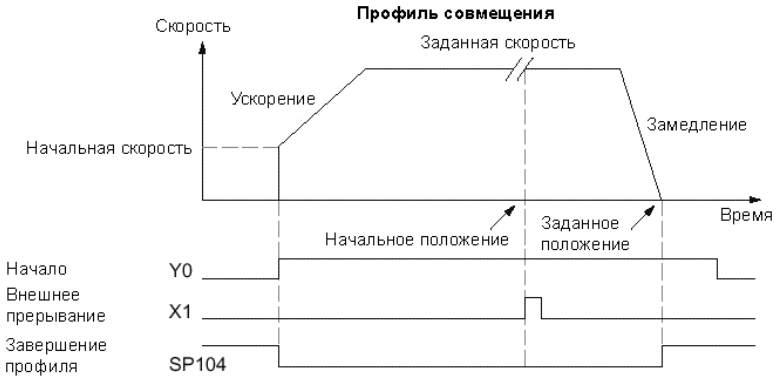
Приложения совмещения:

1. В типовых приложениях, показанных справа, материал обрабатываемого изделия перемещается мимо рабочего инструмента, например, сверлильного устройства. Метки совмещения на обрабатываемом материале позволяют совместить положение машинного инструмента по отношению к прямоугольнику готовой детали, чтобы просверлить ее в надлежащем месте.



2. В других примерах совмещения обрабатываемое изделие неподвижно, а рабочий инструмент перемещается. Сверло может подойти к поверхности обрабатываемой детали, готовое просверлить отверстие точной глубины. Однако длина сверла постепенно уменьшается из-за его износа. Метод, позволяющий преодолеть это, включает определение при каждом сверлении момента контакта сверла с поверхностью детали и погружение его в деталь после контакта на постоянную глубину.
3. Движение при поиске исходного положения позволяет системе управления движением при запуске точно определять ее местоположение. В этом случае система позиционирования делает неопределенные движения и ожидает, когда инструмент пройдет мимо концевого выключателя исходного положения. В момент, когда инструмент находится в известном положении, вырабатывается прерывание. Затем движение останавливается и происходит перезагрузка значения положения с вводом числа, равным физическому «начальному положению».

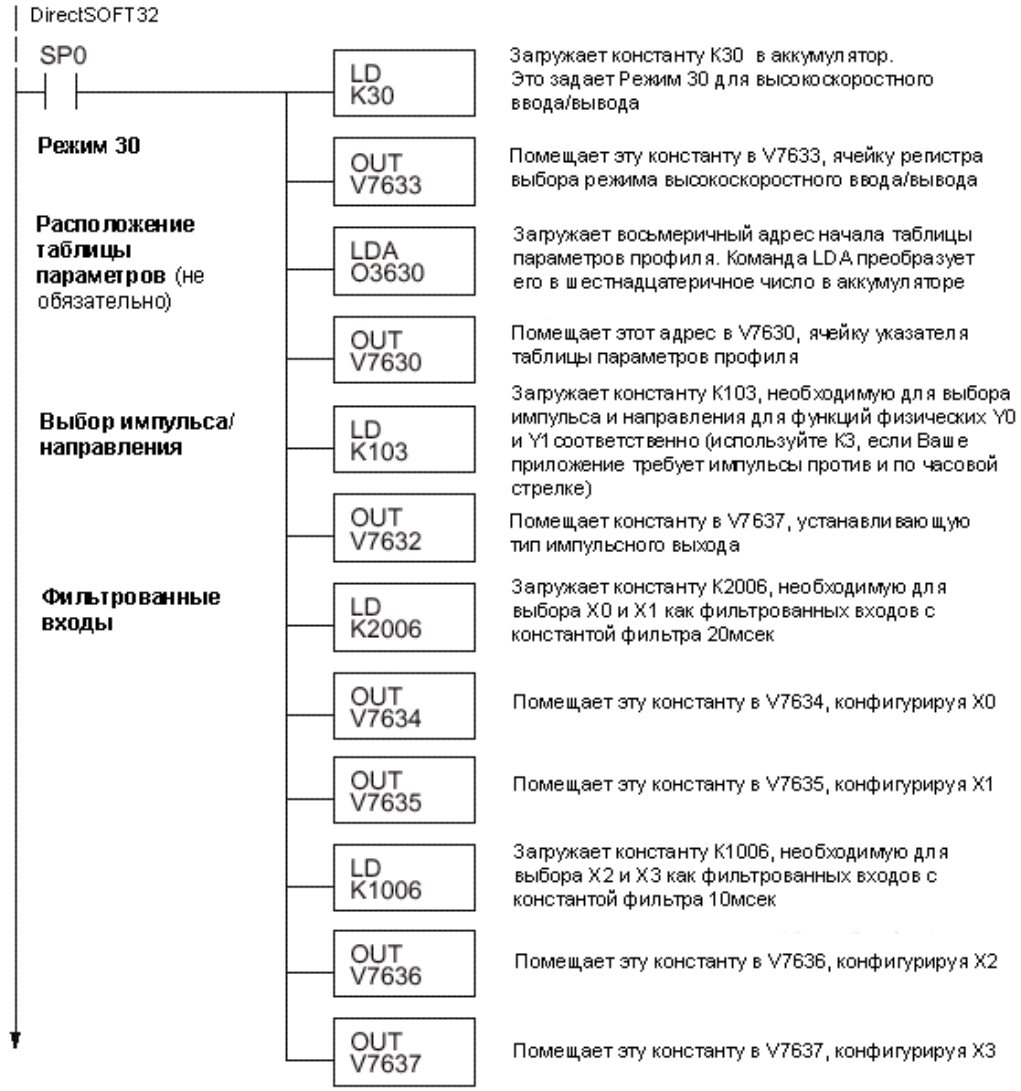
Когда физический вход X1 выдает импульс прерывания, начальным положением для отсчета становится текущее положение инструмента. Управление скоростью переключается на управление положением, при котором рабочий орган передвигается к конечному заданному положению. Напоминаем, что минимальная начальная скорость равна 40 импульсов/сек. Эта мгновенная скорость согласована с возможностью шаговых электродвигателей, которые могут останавливаться при малых скоростях.



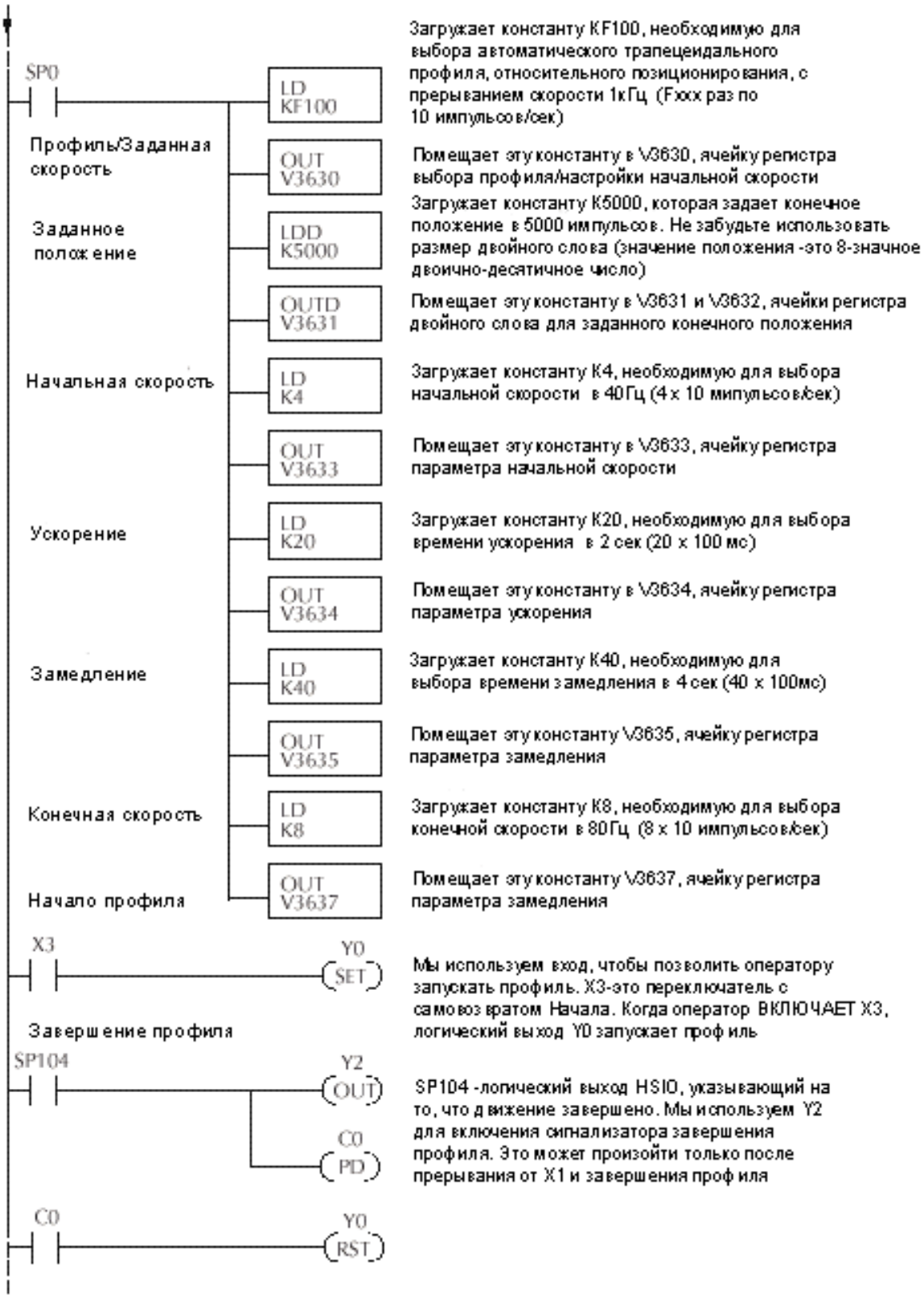
Ниже графика профиля на временной оси представлен график состояния сигналов, который указывает порядок событий. Процессор использует логический выход Y0 для начала профиля. Немедленно HSIO включает сигнал Завершения профиля (SP104), таким образом, программа релейной логики может отследить завершение движения, обнаружив включение этого сигнала.

Автоматический трапецеидальный профиль, который мы хотим реализовать, показан и размечен на следующем рисунке. Он включает ненулевую начальную скорость и среднюю заданную скорость.

В следующей программе реализуется данный профиль. В первой программной цепи вводятся все необходимые параметры настройки. Это должно делаться в программе один раз, поэтому мы используем контакт SP0 первого сканирования, чтобы запустить настройку.



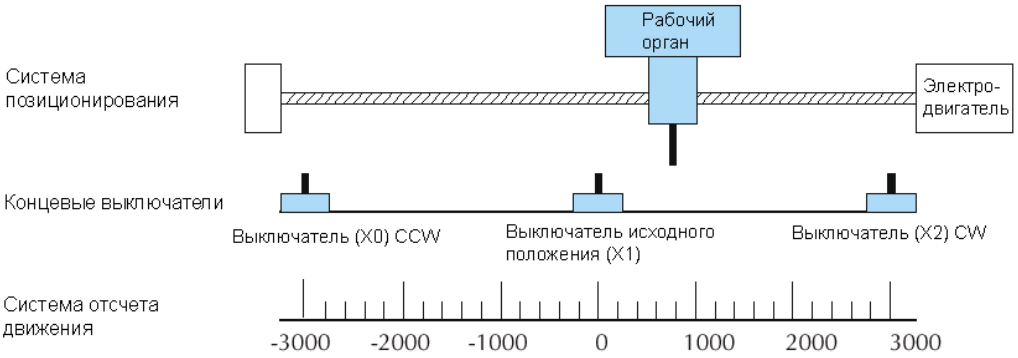
продолжение на следующей странице



Профиль начнется, когда будет задан вход начала X3. Далее движение начинается с неопределенных шагов, которые будут продолжаться до тех пор, пока не произойдет внешнее прерывание X1. Затем движение продолжится на 5000 и более импульсов до останова.

Пример 3: Автоматический трапецеидальный профиль с поиском исходного положения

Одним из наиболее сложных аспектов управления движением является установление фактического положения при запуске системы. Это особенно справедливо для разомкнутых систем, которые не имеют обратной связи по положению. Однако простой концевой выключатель с точным местоположением в механизме позиционирования может обеспечить «обратную связь по положению» в одной точке. Для большинства шаговых систем управления этот метод является хорошим и экономичным решением.



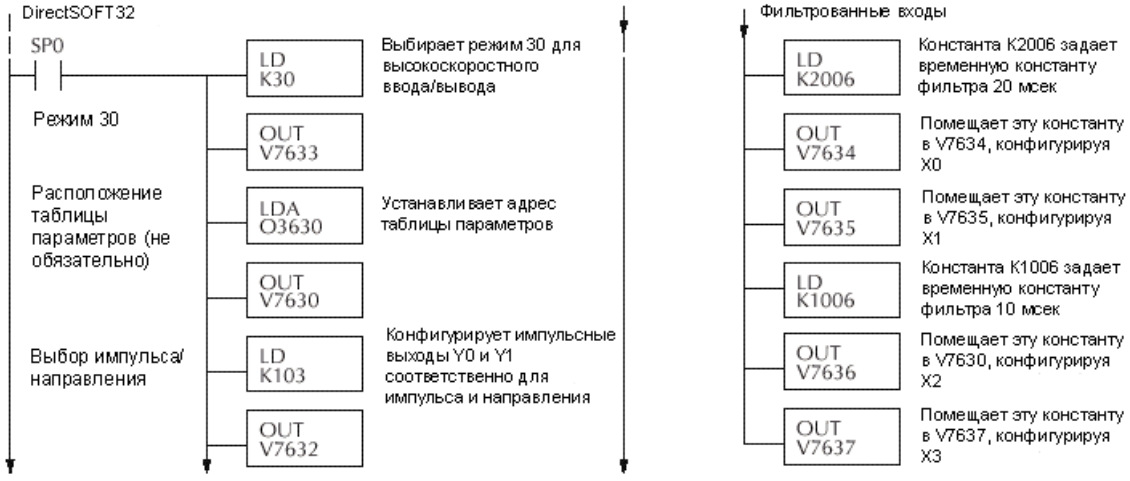
На приведенном рисунке рабочий орган передвигается влево или вправо в зависимости от направления вращения электродвигателя: против часовой стрелки (CCW) или по часовой стрелке (CW). Программа релейной логики воспринимает сигналы концевых выключателей при движении по часовой стрелке или против часовой стрелки и останавливает электродвигатель до того, как рабочий орган выйдет за пределы рабочего пространства и повредит машину. Концевой выключатель исходного положения используется при запуске для установления фактического исходного положения. Система отсчета — произвольна, зависит от единиц измерения, принятых в данном техническом устройстве.

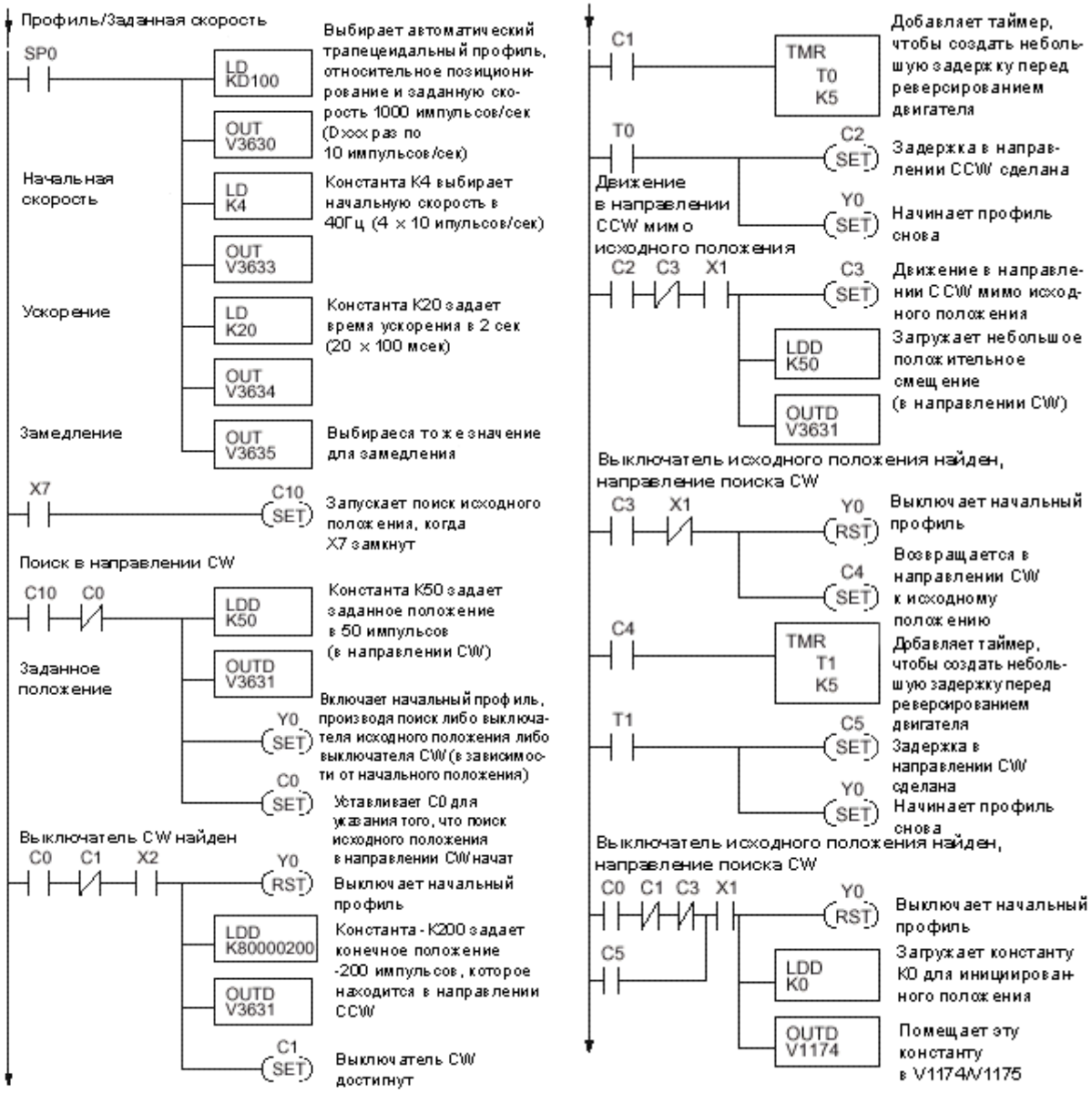
При включении системы мы не знаем, находится ли рабочий орган слева или справа от концевого выключателя исходного положения. Поэтому мы инициируем профиль поиска исходного положения, используя режим совмещения. Концевой выключатель исходного положения подсоединен к X1, который вызывает прерывание. Мы выбираем произвольное начальное направление поиска, двигаясь при вращении двигателя по часовой стрелке (слева направо).

Если концевой выключатель исходного положения будет замкнут первым, то происходит останов и инициализация исходного положения (обычно это значение равно «0», но оно может быть другим, если это удобно).

Однако если первым замкнется концевой выключатель CW, то необходимо реверсировать двигатель и двигаться, пока не замкнется выключатель исходного положения и не произойдет останов сразу после прохождения выключателя.

В последнем случае мы повторяем первое движение, так как нам всегда необходимо сделать последний подход к выключателю исходного положения с тем же самым направлением, так что конечное физическое положение является одним и тем же в любом случае.

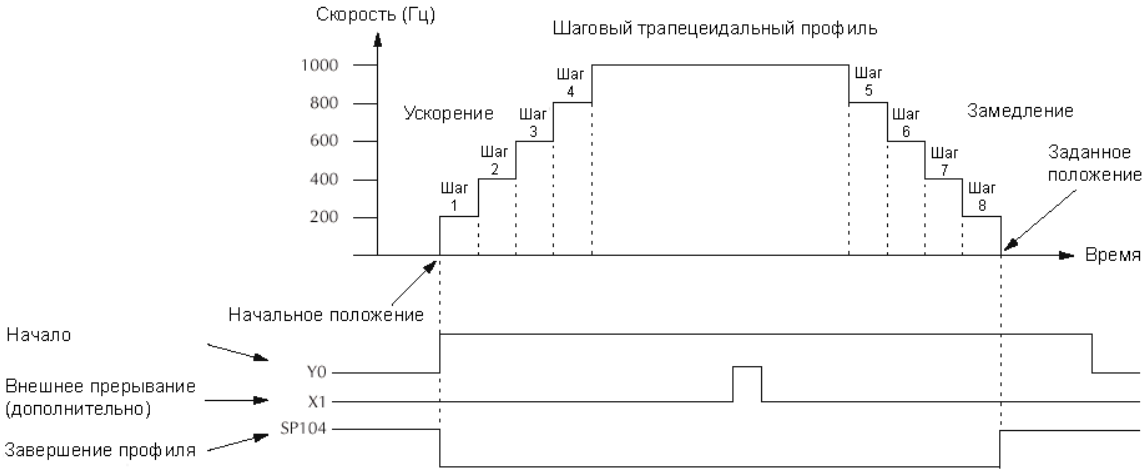




Профиль поиска исходного положения выполняется специфической частью программы, основанной на порядке обнаружения концевых выключателей. Релейная логика устанавливает C0 для запуска поиска в направлении движения по часовой стрелке (CW). Если встретится концевой выключатель CW, то программа продолжает поиск исходного положения в направлении против часовой стрелки (CCW), немного проходит его и выполняет последний поиск этого исходного положения по часовой стрелке (CW). После его достижения последняя программная цепочка загружает значение «0» в текущее положение.

Работа с ступенчатым трапецеидальным профилем

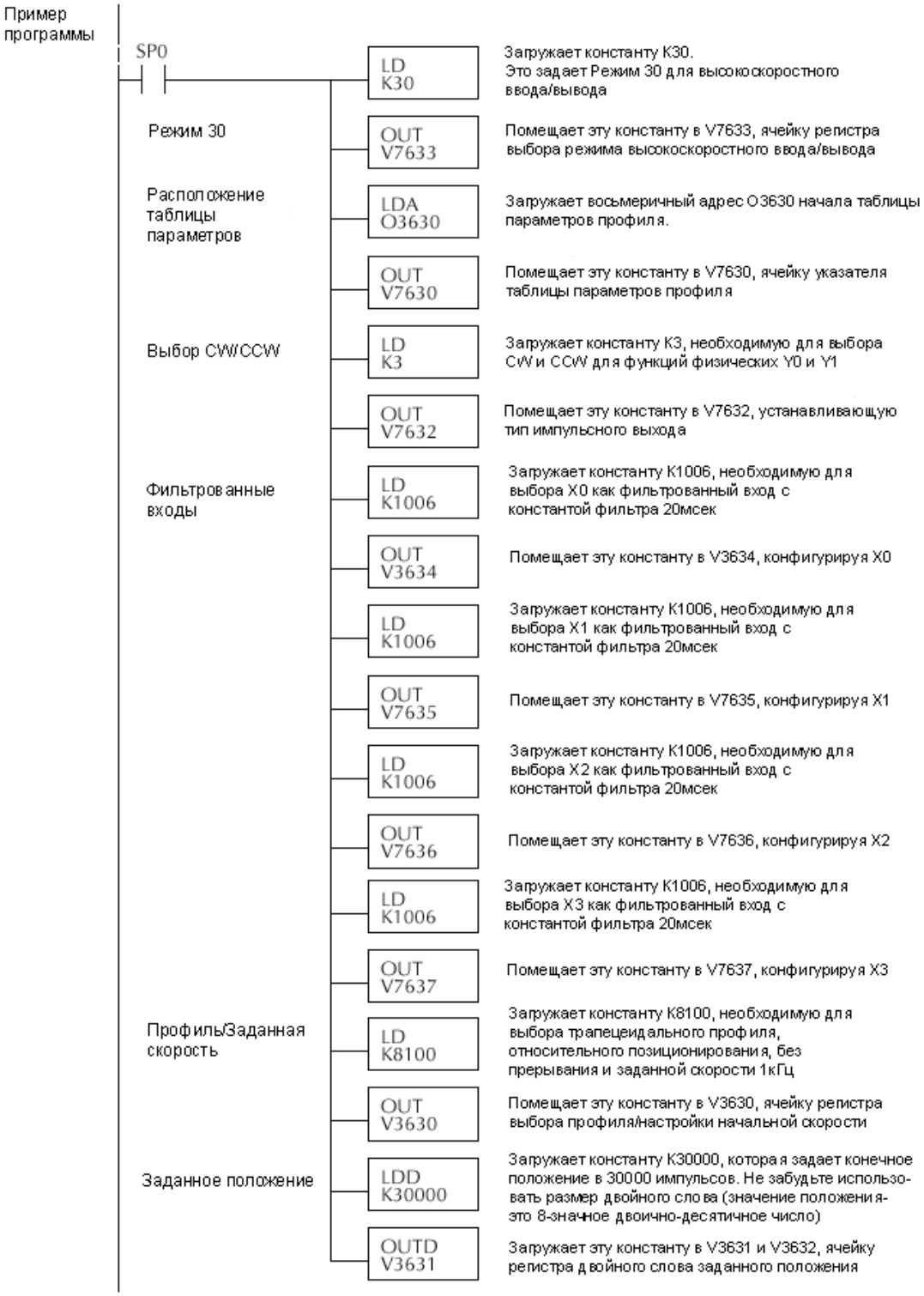
Ступенчатый трапецеидальный профиль позволит управлять наклоном ускорения и замедления по Вашему желанию.



Ниже графика профиля на временной оси представлен график состояния сигналов, который указывает порядок событий. В HSIO используется логический выход Y0 как начальный вход в HSIO, с которого начинается профиль. Немедленно HSIO включает сигнал Завершения профиля (SP104), таким образом, программа релейной логики может отслеживать процесс движения. Обычно программа релейной логики отслеживает этот бит, поэтому она знает, когда инициировать следующий шаг движения. Вы также можете использовать внешнее прерывание (X1). Если выбрано внешнее прерывание для профиля, то DL06 выполняет вывод импульсов только после включения X1. Затем DL06 выводит импульсы, определенные как заданное положение.

Каждый наклон ускорения и торможения состоит из 4 шагов. Вы можете установить скорость и расстояние (число импульсов) для каждого шага. Вы не должны применять все 4 шага каждого наклона. Например, если Вы хотите использовать только 2 шага, установите скорость и расстояние равными нулю для 3-го и 4–го шага. Если наклон ускорения и наклон торможения идентичны, установите все скорости и параметры расстояния для торможения, равными нулю.

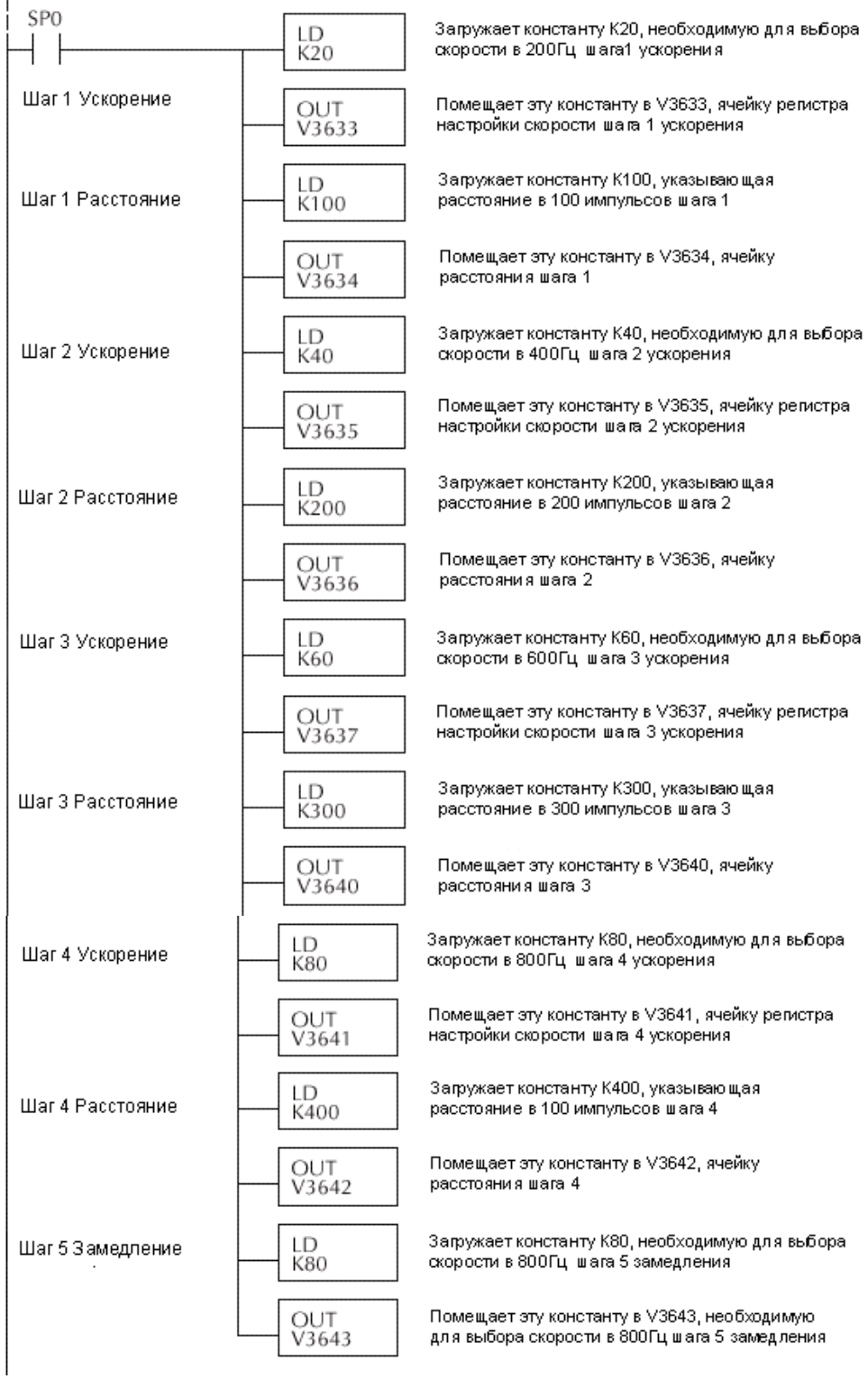
Пример 4: Ступенчатый трапецеидальный профиль



Продолжение на следующей странице.

Начало на предыдущей странице

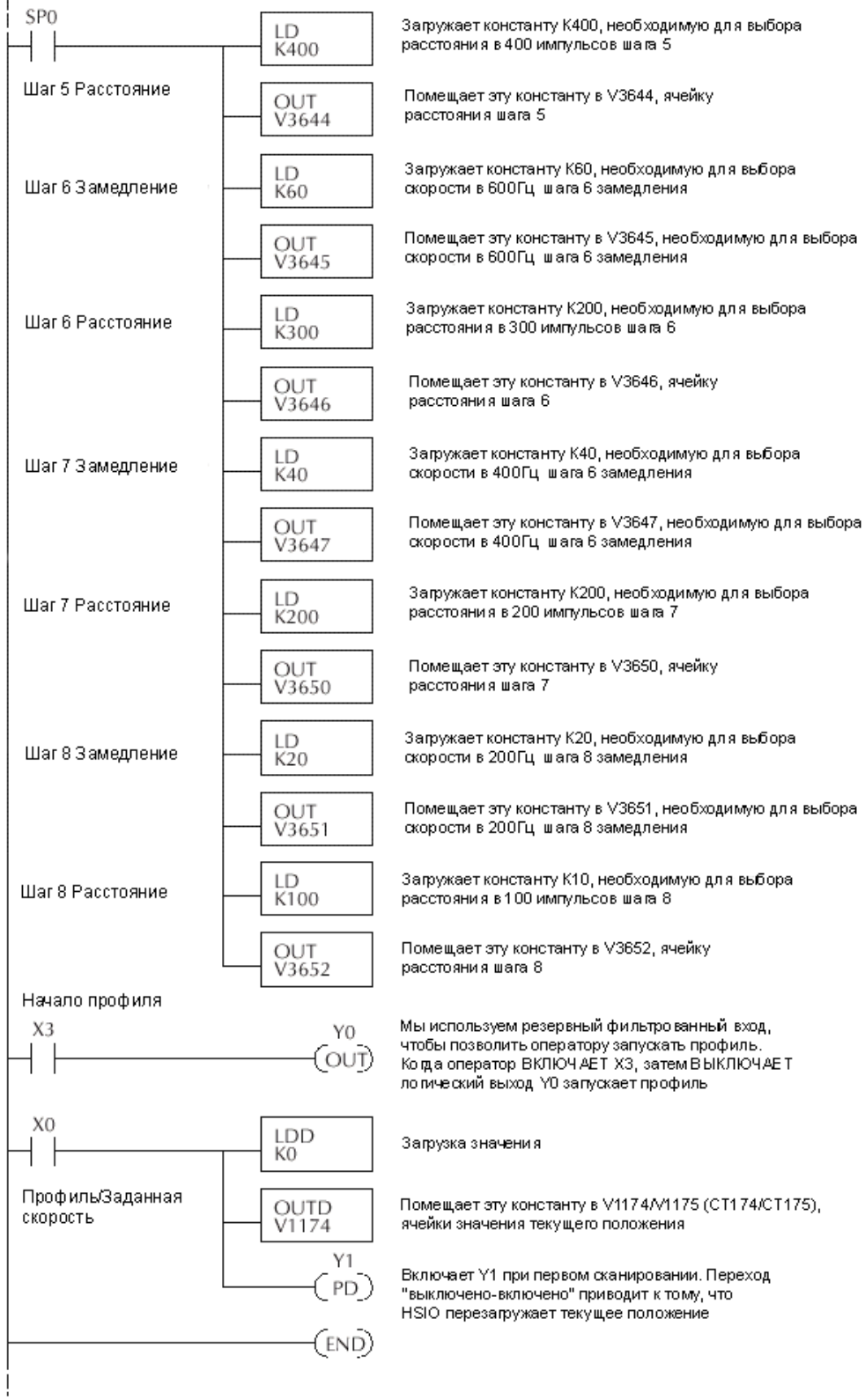
Пример программы



Продолжение на следующей странице.

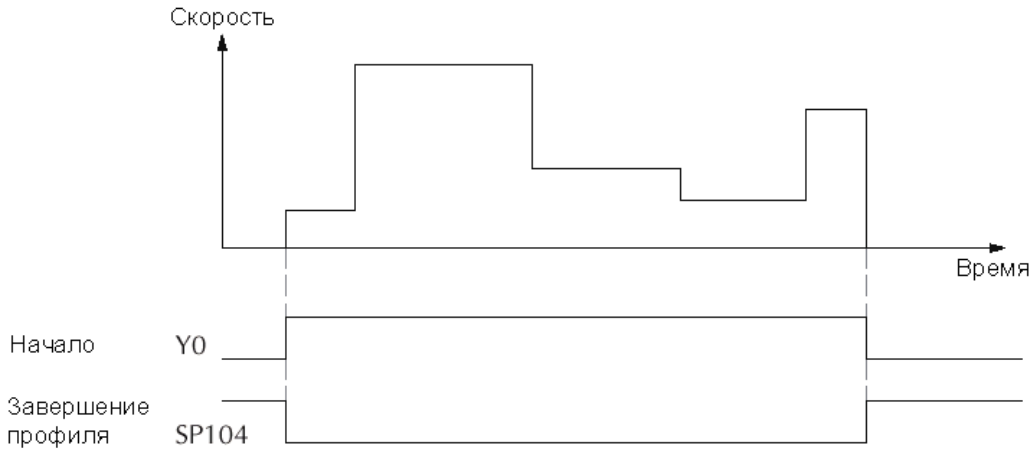
Начало на предыдущей странице

Пример программы



Работа с профилем скорости

Профиль скорости наиболее удобен для приложений, которые включают движение, не требующее перемещения к конкретной точке. Типичным примером такого движения является управление конвейером.



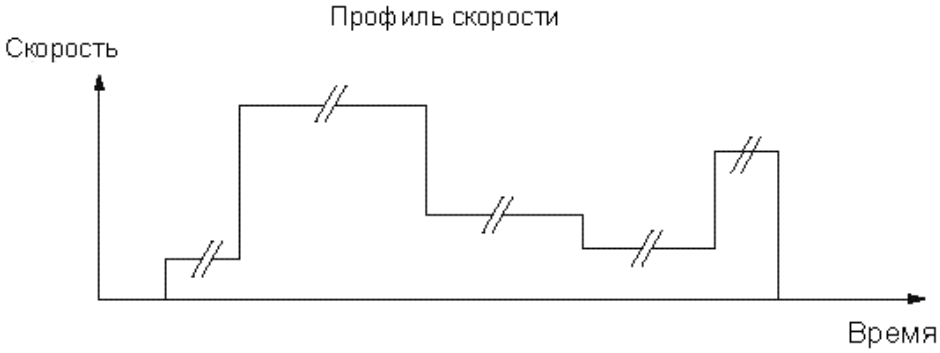
Ниже графика профиля на временной оси представлен график состояния сигналов, который указывает порядок событий. Движение начинается после включения входа Начало (Y0), при этом предполагается, что скорость больше нуля. Поскольку конечная заданная точка отсутствует, профиль рассматривается в процессе выполнения пока вход Начало остается активным. При использовании профилей скорости логический вход (X0) в программу релейной логики прямо связан с состоянием входа Начало, он и определяет завершение профиля.

Пока вход Начало активен программа может выдавать команды на изменение скорости, записывая новое значение в регистр скорости (по умолчанию V3633). Полный диапазон скоростей — от 40Гц до 10КГц. Из рисунка видно, что между изменением скорости нет наклонов с ускорением или замедлением. Так профилирование скорости работает в HSIO. Однако программа может задавать постепенное изменение скорости, более медленно увеличивая или уменьшая значение скорости. При создании ваших собственных участков ускорения или замедления могут быть полезны счетчик или таймер. Если только рабочий орган не должен делать очень сложные движения, то легче вырабатывать участки ускорения/замедления с помощью функции HSIO, чем использовать трапецеидальный профиль или профиль совмещения.

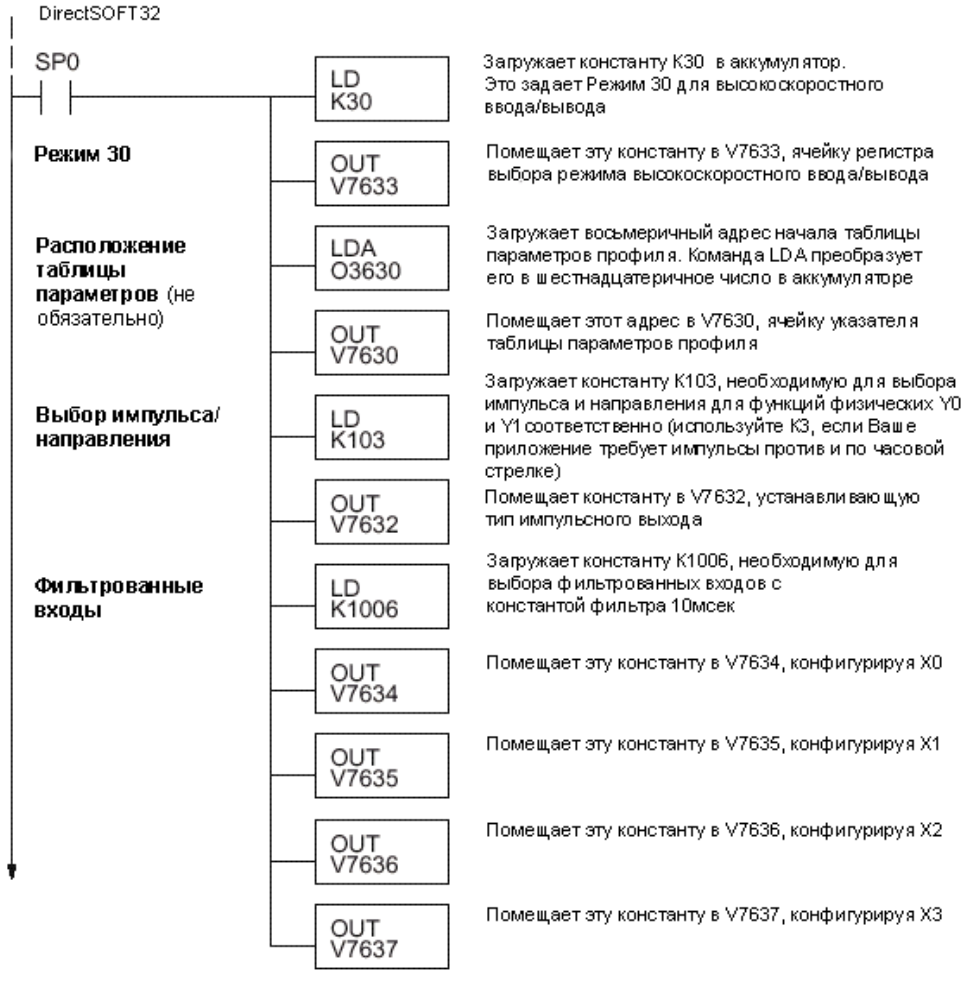
В отличие от трапецеидального профиля или профиля совмещения для профилей скорости Вы должны определять желательное направления передвижения. Загрузите в регистр выбора направления (по умолчанию V3631/ V3632) шестнадцатеричное число 8000 0000 для направления против часовой стрелки (CCW) и 0 для направления по часовой стрелке (CW).

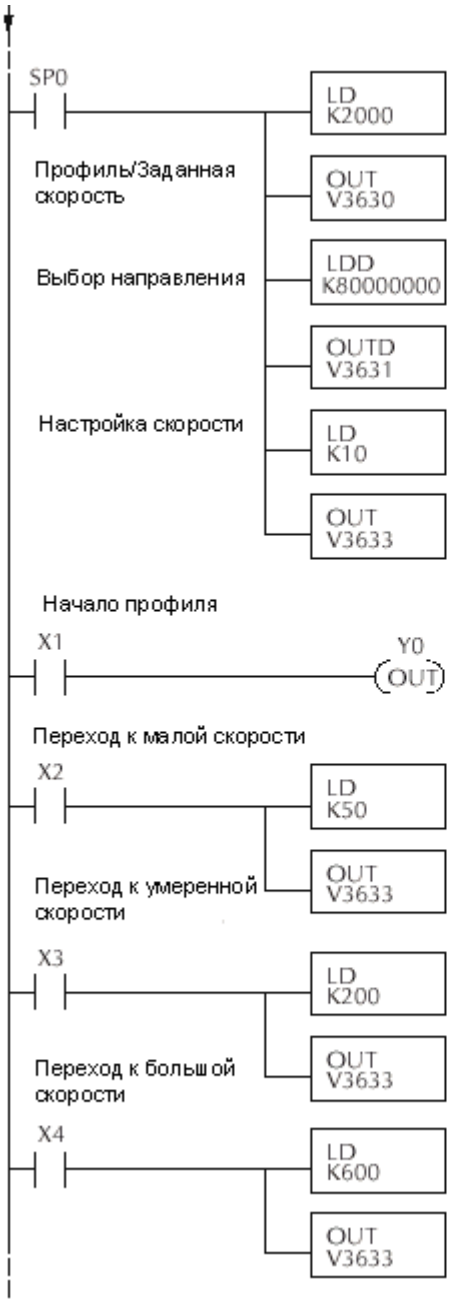
Пример 5: Профиль скорости

Профиль скорости, который мы хотим реализовать, представлен и размечен на следующем рисунке. Каждый сегмент скорости имеет неопределенную длину. Скорость изменяется, только когда программа релейной логики (или другое устройство записи в V-память) обновляет параметр скорости.



В следующей программе используются выделенные дискретные входы для загрузки новых значений скорости. Это — один из вариантов программы, так как Вы можете создать неограниченное число профилей с двумя или тремя выключателями входов. Цель состоит в том, чтобы включить одновременно только один из входов X2, X3 или X4. В начале программы вводятся все необходимые параметры настройки для режима импульсного выхода 30. Мы должны сделать это в программе один раз, поэтому мы используем контакт первого сканирования SP0 для запуска настройки.





Загружает константу K2000, необходимую для выбора профиля скорости. Это слово данных в режиме скорости не содержит информацию о скорости

Помещает эту константу в V3630, ячейку регистра выбора профиля

Загружает константу K80000000, которая задает направление CCW для профилей скорости. Не забудьте использовать размер двойного слова (значение положение-это 8-значное двоично-десятичное число)

Помещает эту константу в V3631и V3632, ячейку регистра двойного слова для заданного положения

Загружает константу K10, необходимую для выбора начальной скорости в 100 импульсов/сек (используется x на множитель 10)

Помещает эту константу в V3633, ячейку регистра параметра скорости.После выполнения программы сюда может быть записано новое значение с использованием переключателей дискретного входа

Мы используем резервный фильтрованный вход, чтобы позволить оператору запускать профиль. Когда оператор ВКЛЮЧАЕТ X1 и оставляет его включенным, логический выход Y0 запускает профиль

Загружает константу K50, необходимую для выбора скорости в 500импульсов/сек , когда оператор замыкает X2

Помещает эту константу в V3633, ячейку регистра параметра скорости. Скорость изменится немедленно.

Загружает константу K200, необходимую для выбора скорости в 2000импульсов/сек, когда оператор замыкает X3.

Помещает эту константу в V3633, ячейку регистра параметра скорости. Скорость изменится немедленно.

Загружает константу K600, необходимую для выбора скорости в 6000импульсов/сек, когда оператор замыкает X4.

Помещает эту константу в V3633, ячейку регистра параметра скорости. Скорость изменится немедленно.

Коды ошибок автоматического трапецеидального профиля

Таблица параметров профиля, начинающаяся в ячейке V3630 (по умолчанию) определяет профиль. Некоторые значения в ней могут привести к ошибке, когда HSIO попытается использовать их при выполнении профиля движения. При обнаружении ошибки HSIO записывает код этой ошибки в ячейку V3636. Большинство ошибок можно исправить при повторной проверке значений в таблице параметров профиля. Эти ошибки автоматически сотрутся при включении питания или при переходе из программного режима в рабочий.

Код ошибки	Описание ошибки
0000	Нет ошибок
0010	Неправильный код типа запрошенного профиля (должен быть 4 – 6 или C-F)
0020	Заданная скорость не в двоично-десятичном формате
0021	Заданная скорость меньше 40 импульсов/сек
0022	Заданная скорость больше 10000 импульсов/сек
0030	Заданное конечное положение не в двоично-десятичном формате
0032	Выбранное направление не 0 или 80000000
0040	Начальная скорость не в двоично-десятичном формате
0041	Начальная скорость меньше 40 импульсов/сек
0042	Начальная скорость больше 1000 импульсов/сек
0050	Время ускорения не в двоично-десятичном (BCD) формате
0051	Время ускорения равно нулю
0052	Время ускорения больше 10 секунд
0060	Время замедления не в двоично-десятичном формате
0061	Время замедления равно нулю
0062	Время замедления больше 10 секунд

Руководство по поиску неисправностей в режиме 30

Если при работе в режиме 30 у вас возникли неисправности, просмотрите следующие их признаки и возможные причины. Наиболее часто встречающие неисправности перечислены ниже.

Признак: шаговый электродвигатель не вращается.

Возможные причины:

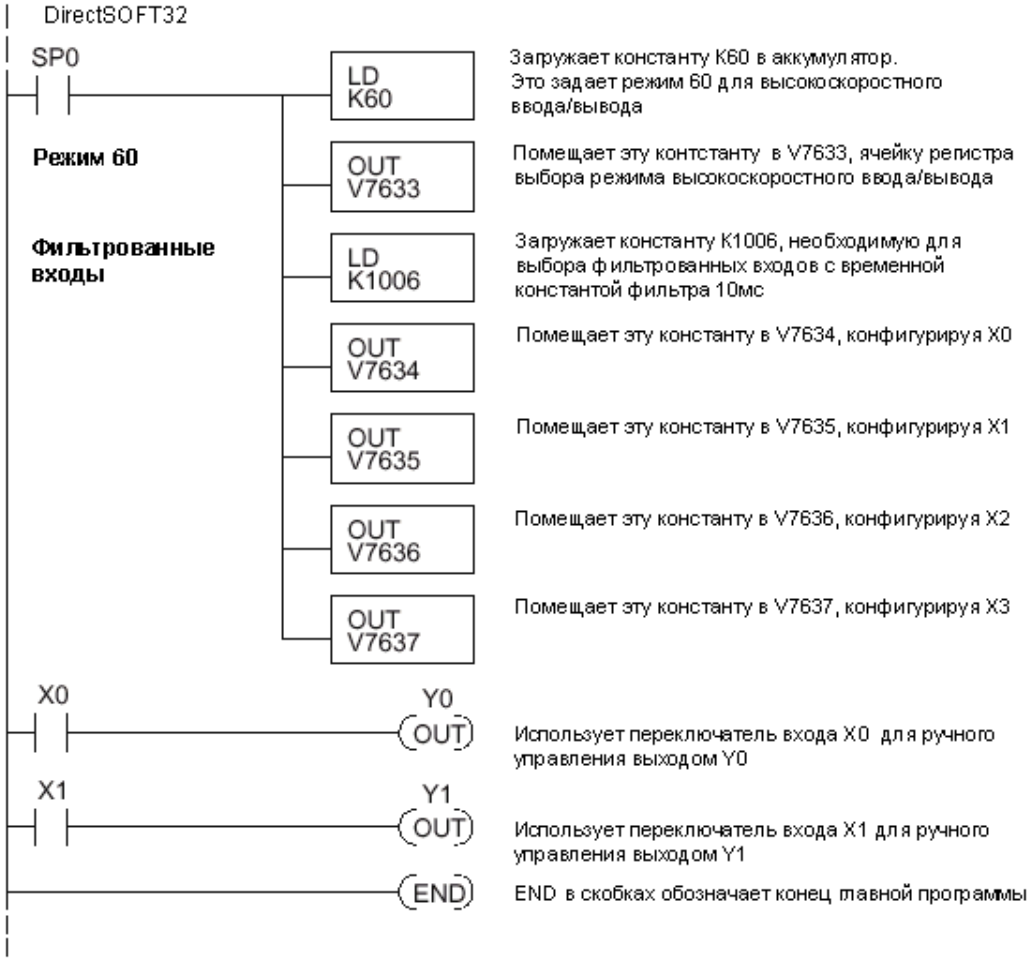
1. Конфигурация. Убедитесь в том, что HSIO реально вырабатывает импульсы выходов Y0 и Y1. Понаблюдайте за светодиодами Y0 и Y1 при включении профиля движения. Если светодиоды мигают или постоянно включены, то, скорее всего, конфигурация правильная.

2. Программная ошибка. Если на Y0 и Y1 нет импульсов, то это может быть программная ошибка. Проверьте содержимое ячейки V2326 с кодом ошибки, которая может быть выработана, когда ПЛК пытается выполнить профиль движения. Описание кодов ошибок приведено выше.

3. Проверьте заданное значение. Для профиля не вырабатываются импульсы, если значение счета равно заданному значению (например, текущий отсчет = 0, заданное значение = 0).

4. Схема соединений. Проверьте правильность схемы подключения шагового электродвигателя. Напоминаем, что сигнал заземления ПЛК должен быть соединен с системой управления движением.

5. Система управления движением. Проверьте подвод питания к приводу и его разблокировку. Чтобы проверить, как система управления движением работает, Вы можете воспользоваться операциями режима 60 (с нормальными входами/выходами ПЛК), как показано в приводимой ниже тестовой программе. В этой программе Вы можете вручную управлять Y0 и Y1 с помощью соответственно X0 и X1. Идеальным для такого типа ручной отладки является применение имитатора входа. С помощью переключателей Вы можете сделать один шаг двигателя в любом направлении. Если при таком простом управлении электродвигатель не заработает, то работа в режиме 30 невозможна, пока неисправность в системе привода электродвигателя или в схеме соединений не будет устранена.



6. Ошибка памяти. Параметры конфигурации HSIO хранятся в системной памяти процессора. Разрушенные данные в этой области памяти могут иногда препятствовать надлежащей работе HSIO. Если все действия по корректировке оказались неудачными, инициализация системной оперативной памяти может решить эту проблему. В DirectSOFT выберите меню ПЛК, затем Настройка (Setup), далее Инициировать оперативную системную память (Initialize Scratchpad).

Признак: электродвигатель вращается в неправильном направлении.

Возможные причины:

1. Схема подсоединения. Если Вы выбрали тип работы CW (по часовой стрелке) и CCW (против часовой стрелки), то вам нужно только переставить провода выходов Y0 и Y1.

2. Управление направлением. Если Вы выбрали тип работы Импульсы и Направление, то вам нужно только изменить бит направления на противоположное состояние.

Режим 40: высокоскоростные прерывания

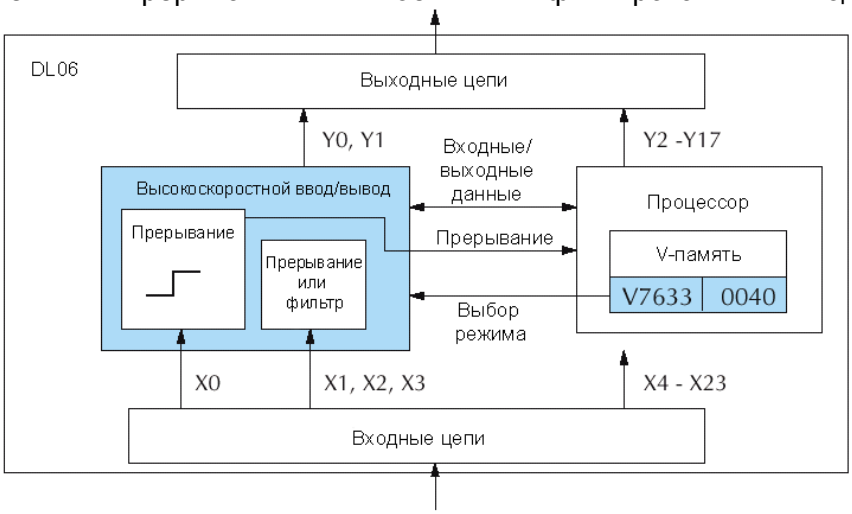
Назначение

Режим 40 высокоскоростного ввода/вывода обеспечивает высокоскоростное прерывание в программе релейной логики. Эта функция предоставляет следующие возможные сценарии вашего приложения:

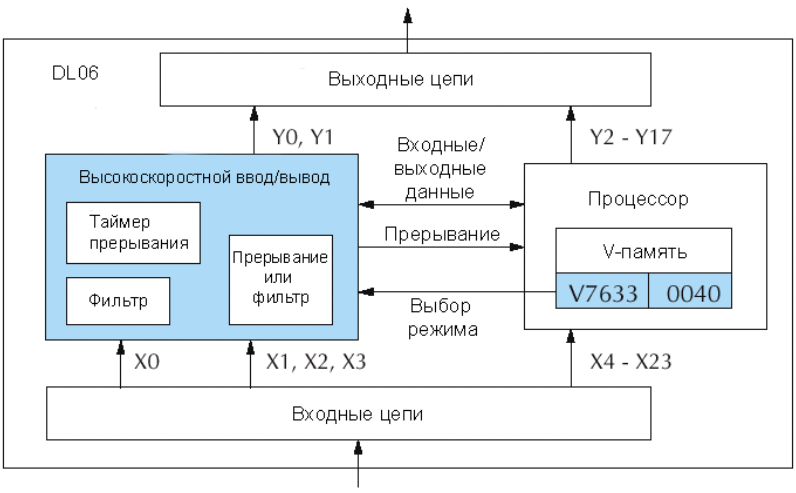
- Внешнее события требует запуска подпрограммы прерывания процессора. Обычно — это использование команд немедленного действия в подпрограмме.
- Программу прерывания необходимо выполнять через определенные интервалы времени, которые отличаются от времени цикла сканирования (они быстрее или медленнее). Прерывание по времени программируется от 5 до 999 мс.

Функциональная блок-схема

Схема высокоскоростного ввода/вывода вырабатывает в процессоре высокоскоростное прерывание. На следующей блок-схеме показан вариант внешнего прерывания, который использует X0. В этой конфигурации X1, X2 и X3 являются внешними прерываниями или обычными фильтрованными входами.



С другой стороны Вы можете сконфигурировать схему высокоскоростного ввода/вывода для генерации прерываний, основанных на таймере, как показано ниже. В этой конфигурации входы X0 являются обычными фильтрованными входами.



Настройка на режим 40

Напоминаем, что ячейка V7633 является регистром выбора режимов HSIO. Используйте двоично-десятичное число «40» в младшем байте V7633 для выбора режима высокоскоростного счетчика.

Выберите из следующего списка наиболее удобный способ программирования ячейки V7633:

- Включить в свою программу команды LOAD и OUT.
- Использовать редактор памяти DirectSOFT.
- Использовать ручной программатор D2-HPP.

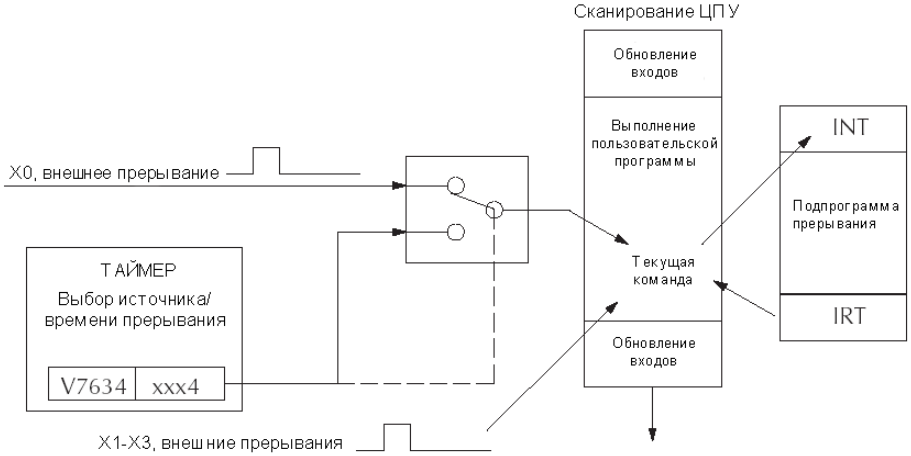
Мы рекомендуем использовать первый способ, при котором настройка высокоскоростного ввода/вывода становится неотъемлемой частью вашей прикладной программы. В примере программы, приведенном далее в данном разделе, показывается, как это делается.

Прерывания и программа релейной логики

Смотрите рисунок ниже. Источником прерывания может быть внешний источник (X0-X3). Внутренний таймер может использоваться вместо X0 как источник прерывания. Параметры настройки в ячейке V7634 предназначены для двух целей:

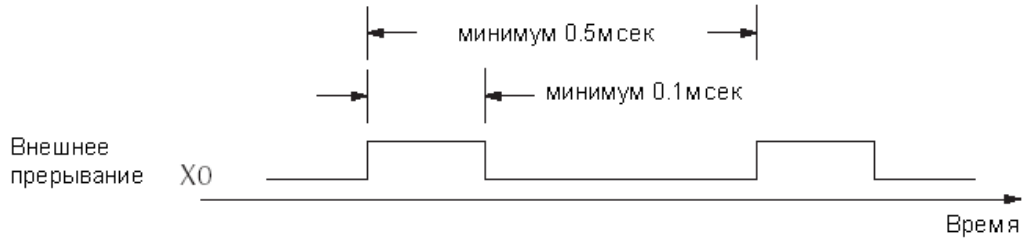
- Они задают выбор между двумя источниками прерывания (внешний или внутренний таймер). Прерывание по времени может использоваться только с X0.
- В случае прерывания по таймеру, они задают временной масштаб прерывания между 5 и 999 мс.

Результирующее прерывание использует в программе метку INT 0, 1, 2 или 3. Убедитесь, что включили команду Разрешение прерывания (ENI) в свою программу. В противном случае программа прерывания не будет выполняться.



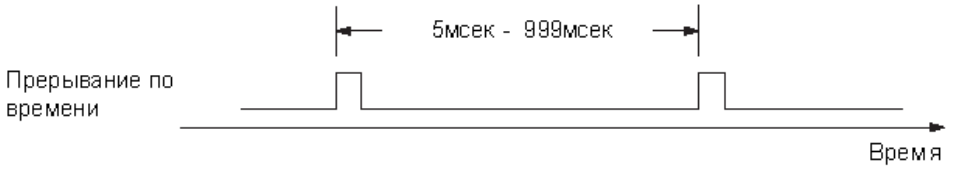
Временные параметры внешнего прерывания

Внешние прерывания должны удовлетворять определенным временным критериям, гарантирующим, что прерывание будет реализовано. Смотрите приведенную ниже временную диаграмму. Минимальная длительность импульса равна 0.1 мс. Должна быть некоторая задержка перед поступлением следующего импульса прерывания, так что период прерывания не может быть меньше 0.5 мс.



Параметры прерывания по времени

При выборе прерывания по времени высокоскоростной ввод/вывод генерирует прерывание в программе релейной логики. В этом случае отсутствует «длительность импульса», а период прерывания может регулироваться от 5 до 999 мс.



Конфигурирование входа X/ прерывания по времени

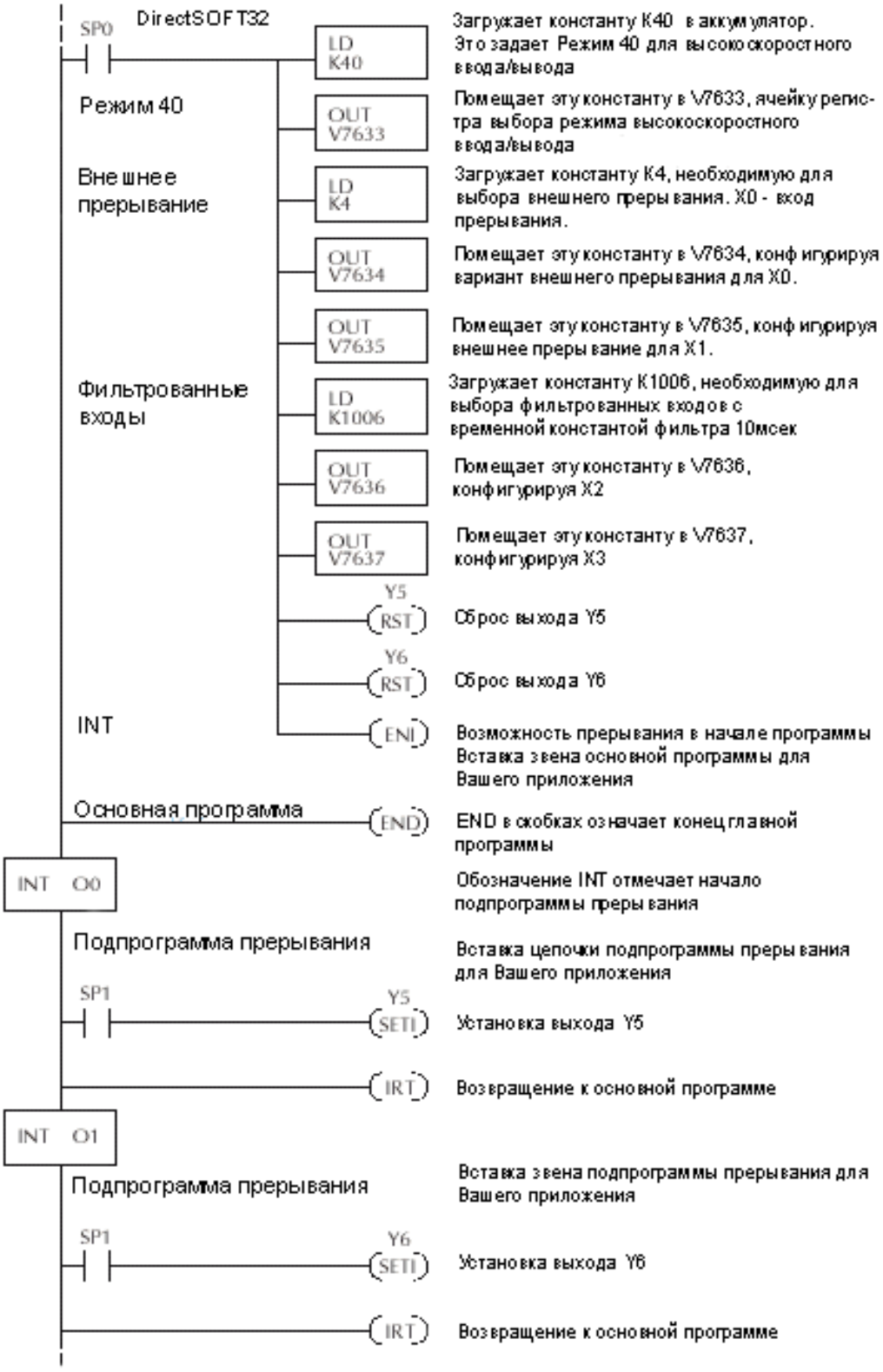
Варианты конфигурируемого дискретного входа для режима высокоскоростного прерывания приведены в таблице ниже. Вход X0 становится внешним прерыванием, когда в ячейке V7634 находится «0004». Если вам требуется прерывание по времени, то ячейка V7634 должна содержать временной период прерывания, а вход X0 должен стать фильтрованным входом (для X1 используется установленная по умолчанию временная константа фильтра). Входы X0, X1, X2 и X3 могут быть фильтрованными входами, имеющими отдельные регистры конфигурации и временные константы фильтра, входы прерывания или счетные входы.

Вход	Регистр конфигурации	Функция	Требуемый шестнадцатеричный код
X0	V7634	Внешнее прерывание	0004 (по умолчанию)
		Прерывание по времени	xxx4, xxx = временному масштабу прерывания 5 - 999 мс (двоично-десятичных)
X1	V7635	Прерывание	0004 (по умолчанию)
		Импульсный вход	0005
		Фильтрованный вход	xx06 (xx = время фильтра) 0 - 99 мс (двоично-десятичных)
X2	V7636	Прерывание	0004 (по умолчанию)
		Импульсный вход	0005
		Фильтрованный вход	xx06 (xx = время фильтра) 0 - 99 мс (двоично-десятичных)
X3	V7637	Прерывание	0004 (по умолчанию)
		Импульсный вход	0005
		Фильтрованный вход	xx06 (xx = время фильтра) 0 - 99 мс (двоично-десятичных)

Если Вы используете только *одну* из точек для прерывания, то можно выбрать различный режим (т.е. 10, 20, 30, 50 или 60); и затем только сконфигурировать один из выводов не принятых как прерывание. Например, Вы можете законфигурировать ЦПУ для счетного режима (Режим 10) и использовать точку 03 для высокоскоростного прерывания. Вы должны отдельно прочитать разделы для любого альтернативного режима, который Вы могли бы выбрать. Там Вы найдете инструкции о том, как выбрать высокоскоростное прерывание для второго входа.

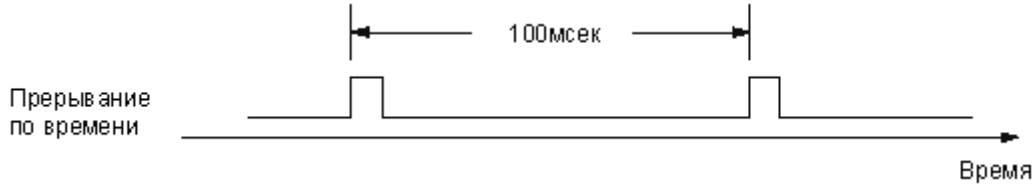
Пример 1: Внешние прерывания

В следующей программе сначало выбирается режим 40, затем вариант внешнего прерывания для входов X0 и X1. Входы X2 и X3 конфигурируются как фильтрованные входы с константой 10мс. Эта программа носит общий характер, она может быть адаптирована к вашему приложению.

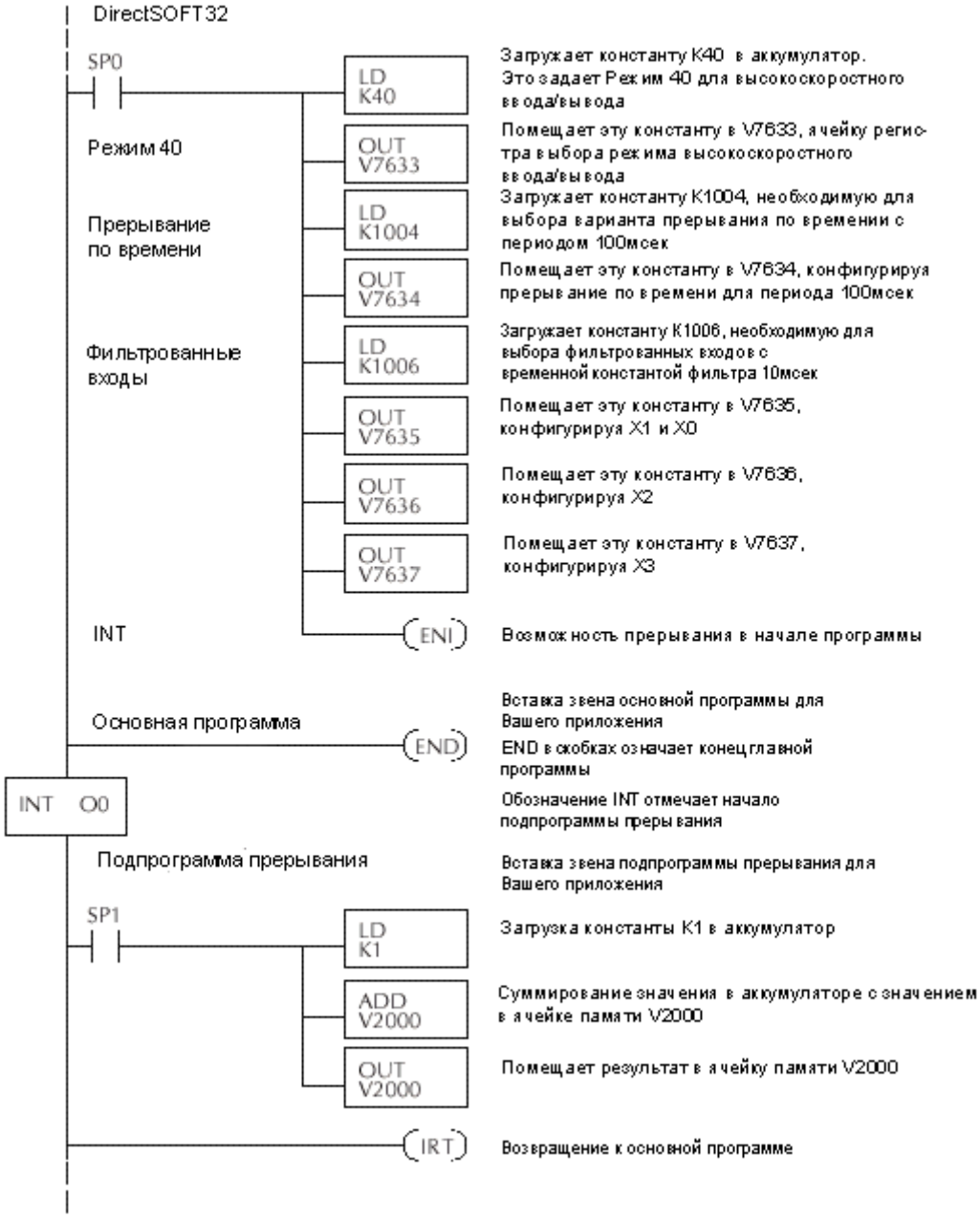


Пример 2: Прерывание по времени

В следующей программе выбран режим 40, затем прерывание по времени с периодом прерывания - 100мс.



Входы X0, X1, X2 и X3 конфигурируются как фильтрованные входы с константой 10мс. Обратите внимание, что X0 использует константу X1. Эта программа носит общий характер, она может быть адаптирована к вашему приложению.



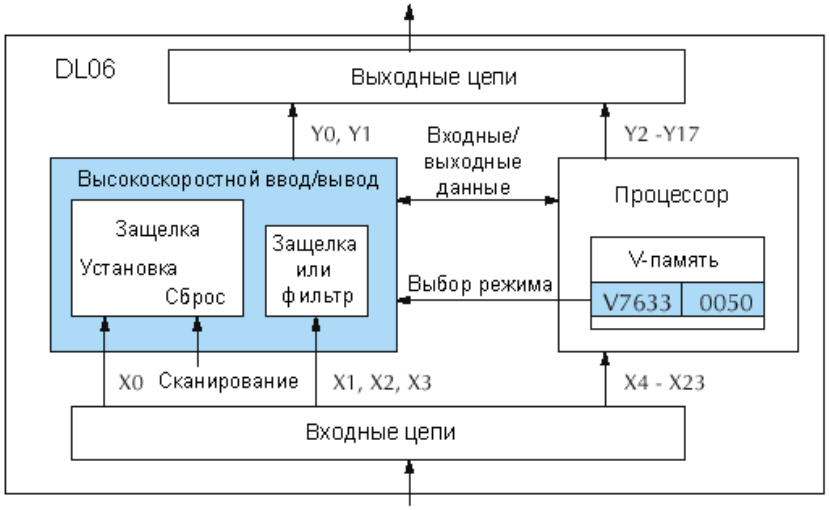
Режим 50: вход с защелкой импульсов

Назначение

Схема HSIO имеет режим защелки входных импульсов. Она отслеживает сигнал на входе X0 - X3, сохраняя случаи появления коротких импульсов. Назначение режима импульсной защелки состоит в том, чтобы дать возможность программе релейной логики «увидеть» входной импульс, который короче по продолжительности, чем время цикла сканирования. Схема HSIO защелкивает входное событие по входу X0 – X3 на одно сканирование и представляет его релейной логике через контакты специального реле SP100. Этот контакт автоматически отключается после одного сканирования. Обратите внимание: программа не может непосредственно считать состояние X0.

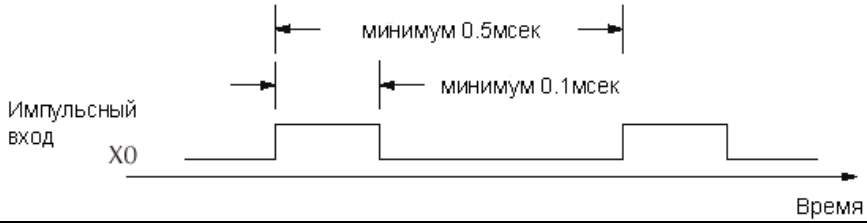
Функциональная блок-схема

Функциональная блок-схема приведена на рисунке ниже. Когда младший байт регистра режимов HSIO V7633 содержит двоично-десятичное число «50», схема HSIO переходит в режим импульсной защелки. Вход X0 – X3 автоматически становится входом с импульсной защелкой, которая устанавливается на каждый нарастающий фронт импульса. HSIO сбрасывает регистр — защелку в конце следующего цикла сканирования процессора. Входы X1, X2 и X3 могут быть также фильтрованными дискретными входами.



Временные параметры защелки импульсов

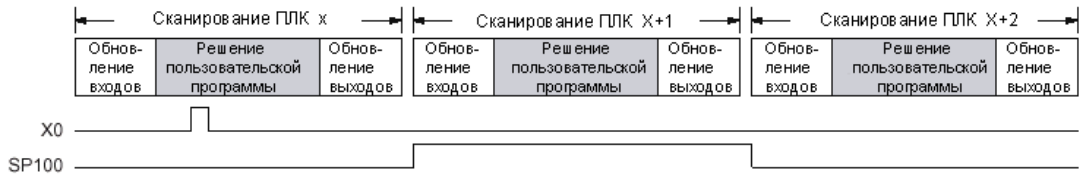
Импульсы сигнала X0 – X3 должны удовлетворять определенным временным критериям, гарантирующим, что захват импульса будет результативным. Смотрите приведенную ниже временную диаграмму. Характеристики входа X0 фиксированы (они не устанавливаются для фильтрованного входа). Минимальная длительность импульса равна 0.1мс. Должна быть некоторая задержка перед поступлением следующего импульса, так что период между ними не может быть меньше 0.5мс. Если период между импульсами меньше 0.5мс, то последующий импульс будет рассматриваться как часть текущего импульса.



ПРИМЕЧАНИЕ. *Следует отметить, что функции защелки входных импульсов и фильтрованный вход противоположны по своей природе. Функция защелки импульсов состоит в том, чтобы попытаться захватить короткий импульс, в то время как функция фильтрованного входа состоит в том, чтобы отбросить короткий импульс.*

Когда используется режим защелки импульсов

Используйте режим защелки импульсов для приложений, в которых вход (например, X0) не может использоваться в пользовательской программе из-за недостаточной длины импульса. Используйте специальное реле SP100 вместо X0. Контакты SP100 остаются замкнутыми в течение следующего цикла сканирования, как показано выше. Даже если вход X0 находится во включенном состоянии более чем один цикл сканирования, SP100 остается включенным только один цикл сканирования.

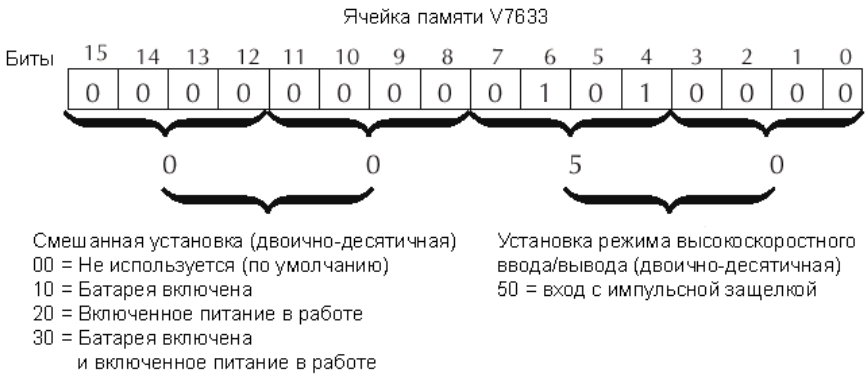


Состояние реле для X0 это SP100. Состояние других реле показаны в таблице ниже.

Вход	Состояние реле
X0	SP100
X1	SP101
X2	SP102
X3	SP103

Настройка на режим 50

Напоминаем, что ячейка V7633 является регистром выбора режимов HSIO. Используйте двоично-десятичное число «50» в младшем байте V7633 для выбора режима высокоскоростного счетчика.



Выберите из следующего списка наиболее удобный способ программирования ячейки V7633:

- Включить в свою программу команды LOAD и OUT.
- Использовать редактор памяти DirectSOFT.
- Использовать ручной программатор D2-HPP.

Мы рекомендуем использовать первый способ, при котором настройка HSIO становится неотъемлемой частью вашей прикладной программы. В примере программы, приведенном далее в данном разделе, показывается, как это делается.

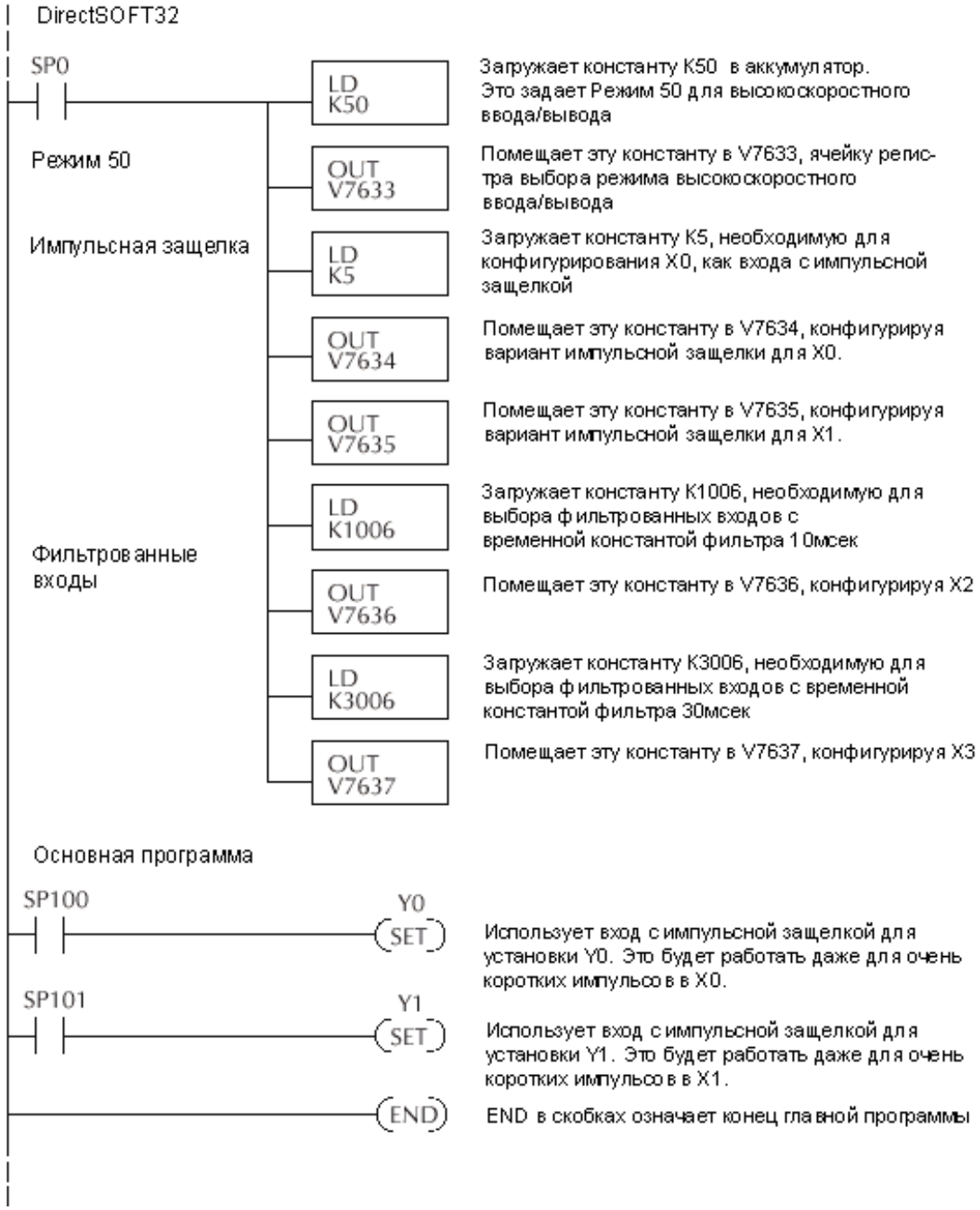
Конфигурирование входа X

Варианты конфигурируемого дискретного входа для режима защелки импульсов приведены в таблице ниже. Каждый вход имеет свой собственный регистр конфигурации и временную константу фильтра.

Вход	Регистр конфигурации	Функция	Требуемый шестнадцатеричный код
X0	V7634	Вход с защелкой импульсов	0005 (по умолчанию)
X1	V7635	Прерывание	0004
		Вход с защелкой импульсов	0005 (по умолчанию)
		Фильтрованный вход	xx06 (xx = время фильтрации) от 0 до 99 мс (двоично-десятичных)
X2	V7636	Прерывание	0004
		Вход с защелкой импульсов	0005 (по умолчанию)
		Фильтрованный вход	xx06, xx = время фильтрации от 0 до 99 мс (двоично-десятичных)
X3	V7637	Прерывание	0004
		Вход с защелкой импульсов	0005 (по умолчанию)
		Фильтрованный вход	xx06, xx = время фильтрации от 0 до 99 мс (двоично-десятичных)

Пример: импульсная защелка

В следующей программе сначала выбирается режим 50, затем программируется защелка импульсов для X0 и X1. Входы X2 и X3 конфигурируются как фильтрованные входы с временными константами фильтра соответственно 10 и 30мс. Программа носит общий характер и может быть адаптирована к Вашему приложению.



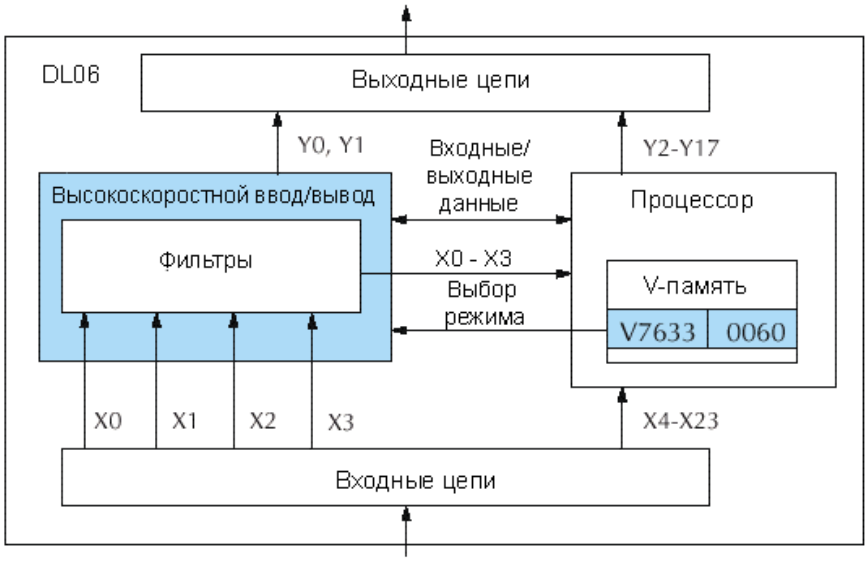
Режим 60: дискретные входы с фильтром

Назначение

Последним режимом схемы HSIO является режим 60, дискретные входы с фильтром. Назначение этого режима состоит в том, чтобы дать возможность входной схеме отбрасывать короткие и принимать длинные с точки зрения программы релейной логики импульсы. Это особенно полезная функция в среде с помехами или в приложениях, где длительность импульса имеет значение. Во все других режимах входы с X0 по X3 обычно поддерживали функции режима как специальные входы. Только резервные входы по умолчанию были фильтрованными входами. В режиме 60 все четыре входа с X0 по X3 функционируют как дискретные фильтрованные входы.

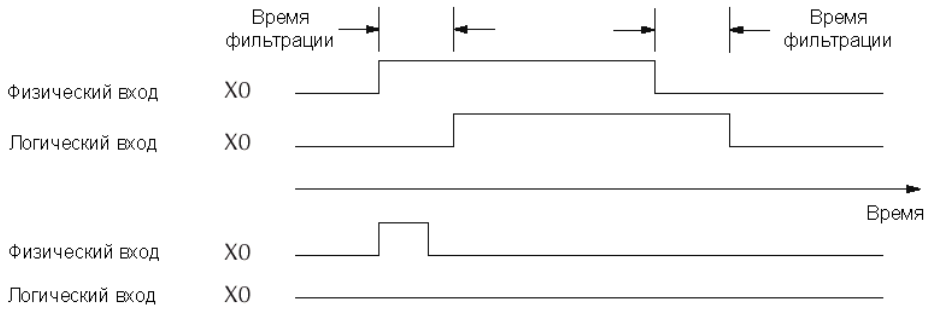
Функциональная блок-схема

Функциональная блок-схема приведена на рисунке ниже. Когда младший байт регистра режимов HSIO V7633 содержит двоично-десятичное число «60», входной фильтр схемы HSIO становится доступным. Каждый вход с X0 по X3 имеет свою собственную временную константу фильтра. Выходы из схемы фильтров имеют логические ссылки на входы X0 – X3.



Временные параметры входных фильтров

Импульсы сигналов на входах X0 – X3 фильтруются с использованием времени задержки. На рисунке ниже входной импульс, изображенный на верхней линии, имеет большую длительность, чем время фильтрации. Результирующий логический вход в программу будет сдвинут по фазе (задержан) на время фильтрации как по переднему, так и по заднему фронту импульса. На нижней временной диаграмме длительность физического входного импульса меньше, чем время фильтра. В этом случае логический вход в программу релейной логики имеет состояние ОТКЛЮЧЕН (входной импульс отфильтровывается).



Настройка на режим 60

Напоминаем, что ячейка V7633 является регистром выбора режимов HSIO. Используйте двоично-десятичное число «60» в младшем байте V7633 для выбора режима высокоскоростного счетчика.

Выберите из следующего списка наиболее удобный способ программирования ячейки V7633:

- Включить в свою программу команды LOAD и OUT.
- Использовать редактор памяти DirectSOFT.
- Использовать ручной программатор D2-HPP.

Мы рекомендуем использовать первый способ, при котором настройка HSIO становится неотъемлемой частью вашей прикладной программы. В примере программы, приведенном далее в данном разделе, показывается, как это делается.

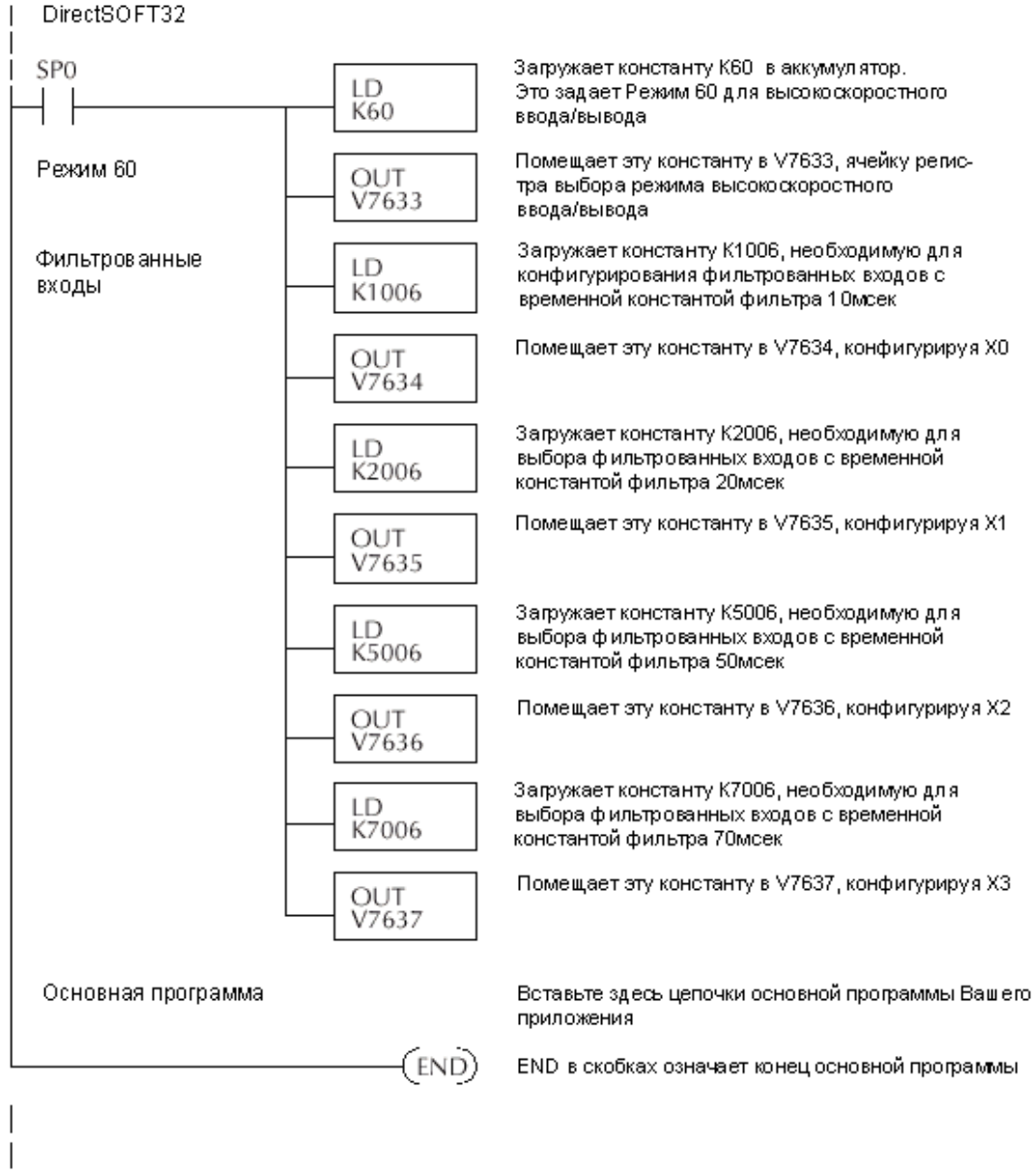
Конфигурирование входа X

Варианты конфигурируемого дискретного входа для режима дискретных фильтрованных входов приведены в таблице ниже. Временные константы фильтров (задержки) устанавливаются от 0 до 99 мс (когда константа времени равна 0, вход действует как обычный дискретный вход). Код этого выбора занимает верхний байт регистра конфигурации в двоично-десятичном формате. Мы комбинируем это число с требуемым значением «06» в нижнем байте, чтобы получить «xx06», где xx = от 0 до 99. Входы X0, X1, X2 и X3 могут быть только фильтрованными входами. Каждый вход имеет свой собственный регистр конфигурации и временную константу фильтра.

Вход	Регистр конфигурации	Функция	Требуемый шестнадцатеричный код
X0	V7634	Фильтрованный вход	xx06 (xx = время фильтрации) от 0 до 99 мс (двоично-десятичных) (по умолчанию)
X1	V7635	Фильтрованный вход	xx06 (xx = время фильтрации) от 0 до 99 мс (двоично-десятичных) (по умолчанию)
X2	V7636	Фильтрованный вход	xx06 (xx = время фильтрации) от 0 до 99 мс (двоично-десятичных) (по умолчанию)
X3	V7637	Фильтрованный вход	xx06 (xx = время фильтрации) от 0 до 99 мс (двоично-десятичных) (по умолчанию)

Пример: фильтрованные входы

В следующей программе сначала выбирается режим 60, затем программируются временные константы задержки фильтра для входов X0, X1, X2 и X3. В иллюстративных целях все временные константы фильтра различны. Программа носит общий характер и может быть адаптирована к вашему приложению.



Глава 4

4. Характеристики и работа процессора.

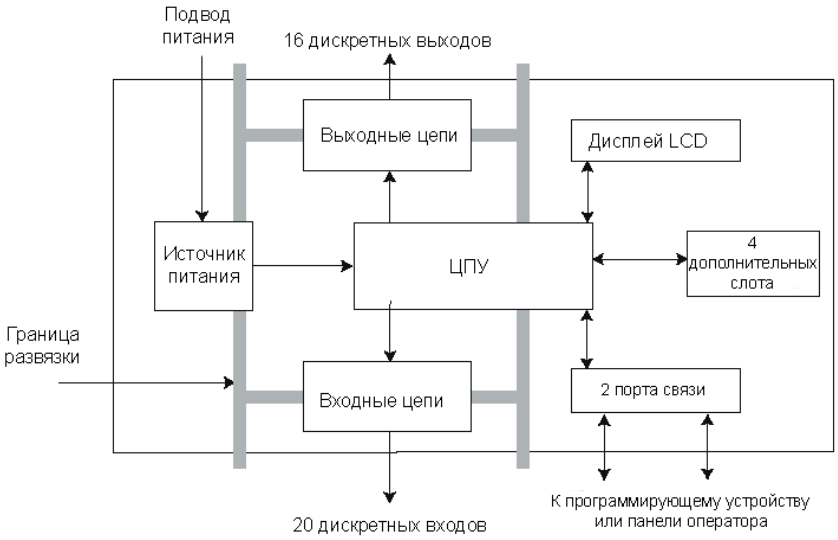
В этой главе…

Введение ..... 4—2
Общие характеристики процессора ..... 4—3
Технические характеристики аппаратных средств процессора ..... 4—4
Использование батарейного питания ..... 4—8
Работа процессора ..... 4—12
Время опроса ввода/вывода ..... 4—17
Анализ времени сканирования процессора ..... 4—20
Карта памяти ..... 4—25
Системная V-память DL06 ..... 4—29
Карта памяти управляющих реле ..... 4—35
Карта бит состояния таймера ..... 4—37
Карта бит состояния счетчика ..... 4—37
Карта бит удаленного ввода/вывода ..... 4—38
Размещение модулей ..... 4—42
Расчет мощности ..... 4—44
Конфигурация портов DL06 ..... 4—46
Подключения к сетям MODBUS и DirectNET ..... 4—48
Нециклический протокол (ASCII ввод/вывод и Печать) ..... 4—50
Функционирование сети в режиме ведомого устройства ..... 4—51
Функционирование сети в режиме ведущего устройства ..... 4—577
Функционирование сети в режиме ведущего устройства (применение команд MRX и MWX) ..... 4—61

Введение

Процессор является сердцем микроконтроллера. Почти все действия ПЛК контролируются процессором, поэтому важно, чтобы он был правильно настроен. В данной главе приводится информация, необходимая для понимания:

- Требуемых шагов по настройке процессора.
- Работы программы релейной логики
- Организации памяти переменных



ПРИМЕЧАНИЕ. *Функция высокоскоростного ввода/вывода (HSIO) является конфигурируемой частью технических средств DL06, но в этой части руководства не рассматривается так, как она не реализуется программой релейной логики. Описание функции высокоскоростного ввода/вывода приведено в главе 3.*

Особенности процессора DL06

DL06 имеет 14.8Кслов памяти, включающей 7.6К программной памяти и примерно 7.6К слов V-памяти (регистры данных). Программы хранятся в ЭППЗУ(Flash), которая является частью платы ЦПУ. Кроме ЭППЗУ на плате процессора имеется также оперативная память (RAM), в которой могут храниться параметры системы, V-память и другие данные, не относящиеся к прикладной программе. Оперативная память поддерживается суперконденсатором, сохраняющим данные несколько часов в случае потери питания. Конденсатор автоматически подзаряжается при работе ПЛК.

DL06 поддерживает 20 дискретных входов и 16 дискретных выходов.

Есть более 220 команд, для разработки программ и интенсивная внутренняя диагностика доступная для прикладных программ. В главах 5, 6 и 7 подробно описаны все команды.

DL06 имеет два встроенных порта, так что Вы можете легко подсоединить ручной программатор, операторскую панель или персональный компьютер.

Общие характеристики процессора

Техническая характеристика	DL06
Общая память пользователя (слов)	14.8K
Программная память (слов)	7680
Вся V-память (слов)	7616
V-память пользователя (слов)	7488
Неразрушаемая V-память (слов)	128
Выполнение 1 булевой операции	2.0мкс
Типичный цикл работы (1К булевых команд)	3-4мс
Программирование на языках RLL и RLLplus	Да
Ручной программатор	Да
Редактирование в рабочем режиме	Да
Цикл работы (сканирование)	Переменный/ фиксированный
Программирование DirectSOFT32 для Windows	Да
Встроенные коммуникационные порты (RS232C)	Да
ЭППЗУ(Flash)	Стандартная на процессоре
Доступные локальные каналы дискретного ввода/вывода	36
Макс. локальных каналов аналогового ввода/вывода	Нет
Высокоскоростной ввод/вывод (квадр., импульсный выход, прерывание, импульсный вход с защелкой и др.)	Да,2
Плотность каналов ввода/вывода	20 входов/16 выходов
Число доступных команд (детально см. главу 5)	229
Управляющие реле	1024
Специальные реле (системные)	512
Стадии в RLLplus	1024
Таймеры	256
Счетчики	128
Немедленный ввод/вывод	Да
Вход прерывания (внешнее/по времени)	Да
Подпрограммы	Да
Циклы FOR/NEXT	Да
Математика (целочисленная и плавающая точка)	Да
Барабанные командоаппараты	Да
Часы истинного времени/календарь	Да
Внутрисистемная диагностика	Да
Защита паролем	Да
Регистрация системных ошибок	Да
Регистрация ошибок пользователя	Да
Батарейное питание	Дополнительно D2-BAT-1 (поставлется отдельно)

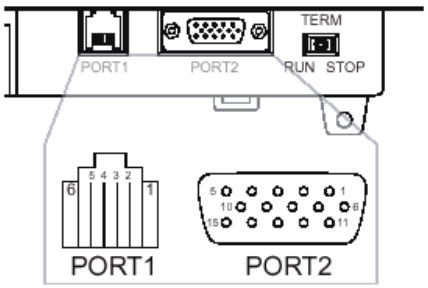
Технические характеристики аппаратных средств процессора

Назначение контактов коммуникационных портов

Поставляемые кабели позволяют быстро и легко подсоединить ручной программатор или персональный компьютер к DL06. Однако, если Вы хотите делать свои кабели, используйте схему контактов портов приведенную далее. Для подключения к портам DL06 необходима вилка телефонного разъема RJ-12 для порта 1 и 15-контактный SVGA D-sub разъем для порта 2.

Процессор DL06 имеет два коммуникационных порта. Порт 1(только RS232C) обычно используется для подключения ручного программатора, DirectSOFT32, операторских панелей, или связи по MODBUS / DirectNET (только ведомое устройство). У порта 1 фиксированная скорость -9600 бод. Порт 2 (RS232C/RS422/RS485) может быть использован для подключения ручного программатора, DirectSOFT32, операторских панелей. Настройки порта: от 300бод до 38.4Кбод, DirectNET ведущий/ведомый, MODBUS ведущий/ведомый и ASCII ввод/вывод.

Порт 1	Контакт	Описание
1	0V	Контакт питания (-) (земля)
2	5V	Контакт питания (+)
3	RXD	Прием данных (RS232S)
4	TXD	Передача данных (RS232S)
5	5V	Контакт питания (+)
6	0V	Контакт питания (-) (земля)



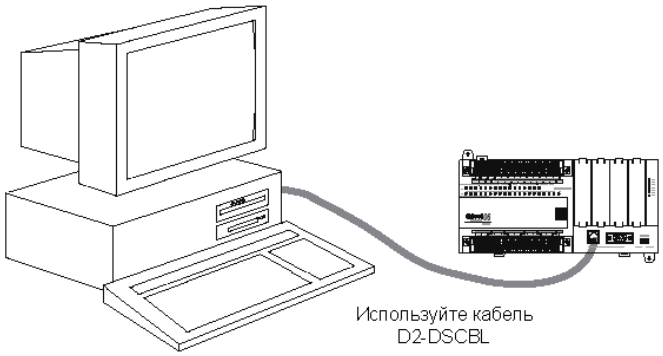
Порт 2	Контакт	Описание
1	5V	Контакт питания (+)
2	TXD	Передача данных (RS232C)
3	RXD	Прием данных (RS232C)
4	RTS	Запрос передачи
5	CTS	Готов к передаче
6	RXD-	Прием данных (-) (RS422/485)
7	0V	Контакт питания (-) (земля)
8	0V	Контакт питания (-) (земля)
9	TXD+	Передача данных (+) (RS422/485)
10	TXD-	Передача данных (-) (RS422/485)
11	RTS+	Запрос передачи (+) (RS422/485)
12	RTS-	Запрос передачи (-) (RS422/485)
13	RXD+	Прием данных (+) (RS422/485)
14	CTS+	Готов к передаче (+) (RS422/485)
15	CTS-	Готов к передаче (-) (RS422/485)

Порт 1
Подключение ручного программатора, DirectSOFT32, операторских панелей и т.д. 6-контактный, RS232C Фиксированная скорость (бод) - 9600 Контроль по четности: нечетный (по умолчанию) Адрес станции: 1 (фиксированный) 8 бит данных 1 стартовый, 1 стоповый бит Асинхронный, полудуплекс, DTE Протоколы: (автовыбор) K-sequence (только ведомый) DirectNET (только ведомый) MODBUS (только ведомый)

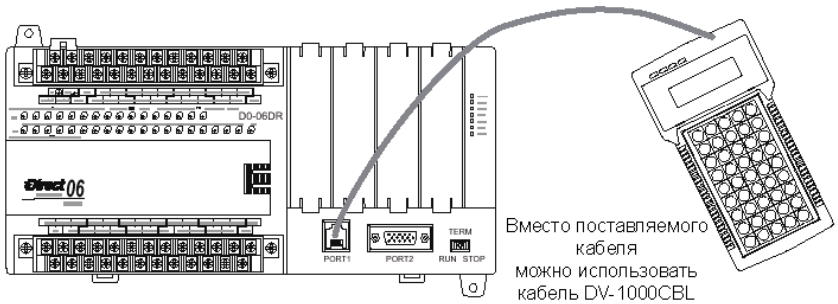
Порт 2
Подключение ручного программатора, DirectSOFT32, операторских панелей и т.д. 15-контактный, многофункциональный порт, RS232C, RS422, RS485 Скорость (бод): 300, 600, 1200, 2400, 4800, 9600, 19200, 38400 Контроль по четности: нечетный (по умолчанию), четный, нет Адрес станции: 1 (по умолчанию) 8 бит данных 1 стартовый, 1 стоповый бит Асинхронный, полудуплекс, DTE Протоколы: (автовыбор) K-sequence (только ведомый) DirectNET (ведущий/ведомый) MODBUS (ведущий/ведомый) non- sequence/печать/ASCII ввод/вывод

Подсоединение устройств для программирования

Если Вы используете персональный компьютер с пакетом программирования DirectSOFT32, то Вы можете использовать любой порт. Для разработки программ этот способ программирования предпочтителен.



Ручной программатор подключается к процессору с помощью кабеля ручного программатора. Ручной программатор поставляется вместе с кабелем длиной примерно 2 м. Это устройство идеально для корректировки существующих программ и наладки систем контроля и управления.

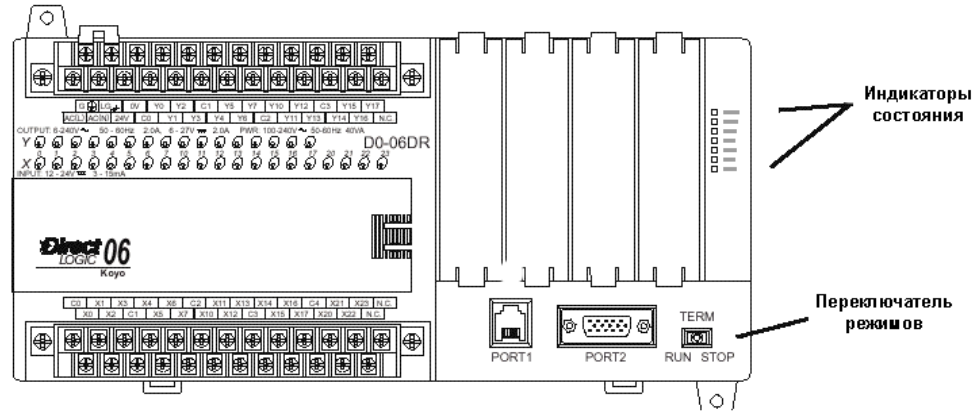


Информация о настройках процессора

Даже если Вы имеете опыт использования ПЛК, просмотрите некоторые моменты перед началом программирования. В этой части описаны некоторые известные понятия, такие как изменение режима работы процессора, но есть и новые. Далее приведен краткий список обсуждаемых тем:

- Использование вспомогательных функций
- Очистка программной и другой памяти
- Инициализация системной памяти
- Установка областей сохранения памяти (Retentive memory range)

В последующих параграфах приведена вся необходимая информация для подготовки процессора к программированию с помощью любого устройства программирования. Для самого программирования необходимо руководство на ручной программатор D2-HPP или руководство на DirectSOFT32.



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
	Мигание	Процессор находится в режиме обновления встроенного программного обеспечения
CPU	ON	Процессор диагностировал ошибку
	OFF	Процессор диагностировал исправную работу
	Мигание	Разрядка батареи или обновление встроенного программного обеспечения
TX1	ON	ЦПУ передает данные. Порт 1
	OFF	ЦПУ не передает данные. Порт 1.
RX1	ON	ЦПУ принимает данные. Порт 1
	OFF	ЦПУ не принимает данные. Порт 1.
TX2	ON	ЦПУ передает данные. Порт 2.
	OFF	ЦПУ не передает данные. Порт 2
RX2	ON	ЦПУ принимает данные. Порт 2.
	OFF	ЦПУ не принимает данные. Порт 2.

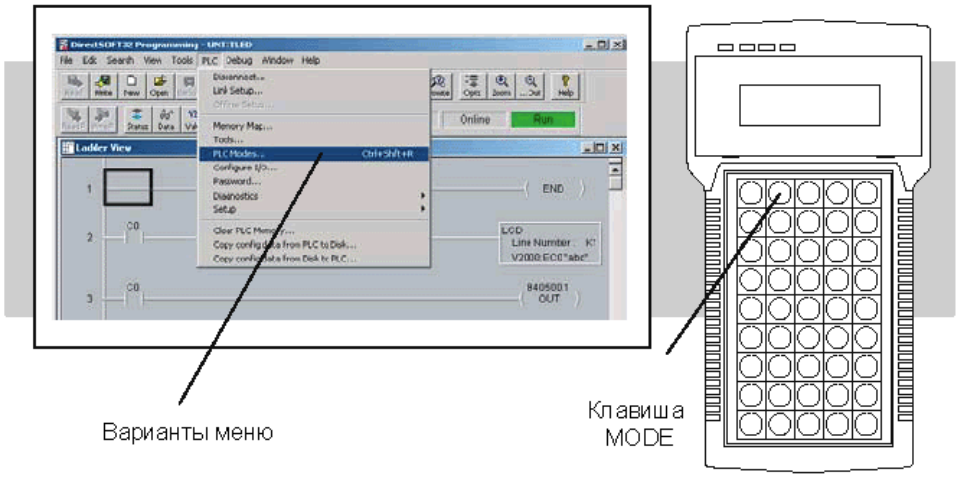
Функции переключателя режимов

Переключатель режимов в DL06 имеет положения для разрешения и блокировки изменения состояния работы процессора. Если переключатель режимов не находится в положении TERM, то изменения в режиме RUN и STOP не разрешаются никакому интерфейсному устройству (ручному программатору, программному пакету DirectSOFT32 или интерфейсу оператора). Программы могут просматриваться, но никакие изменения не могут вноситься. Если переключатель находится в положении TERM и программы не защищены паролем, то доступны все рабочие режимы, также как и сами 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) RUN	Допустимы режимы PROGRAM и TEST. Разрешены изменения режимов и программ с помощью подсоединенных устройств программирования/контроля.
STOP	Процессор переводится в режим STOP. Никакие изменения с помощью подсоединенных устройств программирования/ контроля не разрешены.

Изменение режима работы ПЛК DL06

Существует два способа изменения режима процессора. Вы можете использовать переключатель режимов процессора для выбора рабочего режима. Или Вы можете 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). В ручном программаторе используйте клавишу MODE.



Режим работы при включении питания

Обычно ПЛК DL06 при включении питания начинает работать в режиме, в котором он работал до отключения питания. Например, если до отключения питания DL06 работал в режиме программирования, то после включения питания будет в режиме программирования (см.предупреждение).

ПРЕДУПРЕЖДЕНИЕ. Если конденсатор был разряжен, системная память не сохранит предыдущий режим работы. Когда это произойдет, то при включении питания контроллер может оказаться в режиме программирования или работы, если переключатель режимов был в положении TERM. Нет способа установить нужный режим в этом случае. Если не учитывать это предупреждение, возможен неожиданный пуск оборудования.

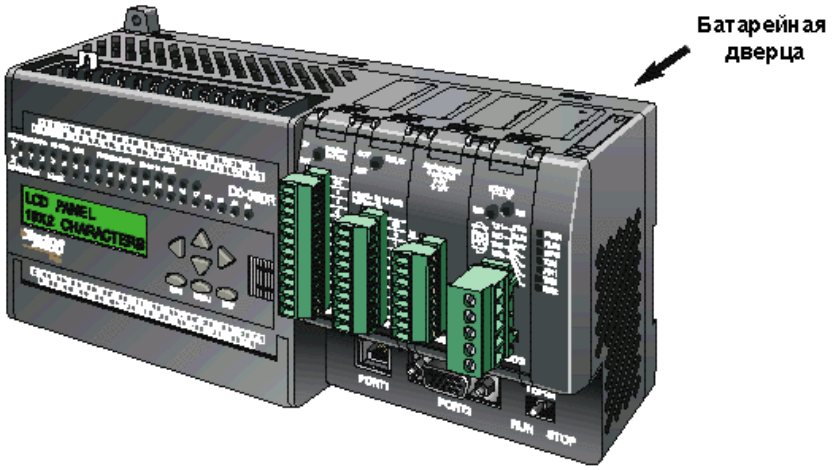
Использование батарейного питания

DL06 может быть снабжен литиевой батареей, которая позволяет сохранять данные в оперативной памяти (Retentive memory) 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.

***ПРИМЕЧАНИЕ**: Перед установкой или заменой батареи, скопируете Вашу V-память и системные параметры. Вы можете сохранить программу, V-память и системные параметры используя DirectSOFT32 на жестком/гибком диске или на персональном компьютере.*

Установка батареи D2-BAT-1 в процессор DL06:

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



ПРЕДУПРЕЖДЕНИЕ. Не пытайтесь перезарядить батарею или класть старую батарею вблизи огня. Батарея может взорваться или выделить опасные компоненты.

Батарейное питание

Батарейная поддержка начинает действовать немедленно после установки батареи. Пользователь может задать диапазон энергонезависимой памяти для хранения данных в C, S, T, CT и V-памяти, если питание отключено. Индикация разрядки батареи может быть включена, если установить 12 бит в регистре V7633. Индикатор на процессоре будет мигать, если напряжение у батареи низкое (смотрите таблицу на странице 4-6). Если батарея разряжена, то специальное реле 43 (SP43) будет в состоянии «включено». Если в индикации разрядки батареи нет необходимости, не устанавливайте 12 бит в V7633.

Вспомогательные функции

Многие задачи настройки процессора предполагают использование вспомогательных (AUX) функций. AUX функции выполняют много различных операций, включая очистку программной памяти, отображение времени сканирования, копирование программ в ЭППЗУ и в ручной программатор и др. Они разделены на категории, которые предназначены для различных системных параметров. В приложении A приводится описание этих AUX функций.

Вы можете получить доступ к AUX функциям из DirectSOFT32 или с помощью ручного программатора. Руководства по этим продуктам включают пошаговые процедуры для получения доступа к AUX функциям. Некоторые из этих AUX функций разработаны специально для ручного программатора, поэтому они не нужны (или не доступны) для пакета DirectSOFT32. В таблице ниже приводится перечень вспомогательных (AUX) функций для ручного программатора (HPP).

AUX 2* — Операции RLL	
21	Проверка программы
22	Изменение ссылки
23	Очистка памяти в заданном диапазоне цепей
24	Очистка памяти по всем цепям
AUX 3* — Операции с V-Памятью	
31	Очистка V-Памяти
AUX 4* — Конфигурация ввода/вывода	
41	Показать конфигурацию ввода/вывода
42	Диагностика ввода/вывода
44	Включите проверку конфигурации ввода/вывода
45	Выбор конфигурации
46	Конфигурация ввода/вывода
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	Установить коммуникационныйпорт 2
57	Установить области сохранения
58	Операции тестирования
59	Перезапись Setup
5B	Конфигурировать высокоскоростной ввод/вывод
5C	Показать историю ошибки
5D	Настройка скан-цикла
AUX 6* — Конфигурация ручного программатора	
61	Показать номер версии
62	Включить/отключить звуковой сигнал
65	Запустить самодиагностику
AUX 7* — Операции ЭППЗУ	
71	Копировать память процессора в ЭППЗУ программатора
72	Записать ЭППЗУ программатора в процессор
73	Сравнить процессор с ЭППЗУ программатора
74	Проверить очистку памяти (ЭППЗУ программатора)
75	Стереть ЭППЗУ программатора
76	Показать тип ЭППЗУ (Процессор и программатора)
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 программную память. 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 RLL
- AUX 24 — очистка всей памяти программ
- AUX 31 — очистка V-Памяти

Инициализация системной памяти

Процессор DL06 поддерживает системные параметры в области оперативной памяти, которую часто называют «электронный блокнот» — Scratch Pad. 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

Помните, что данная AUX функция восстанавливает всю системную память. Если Вы установили специальные параметры, например, сохраняемые области и т. д., то при использовании AUX 54 они будут стерты. Перед выбором этой операции убедитесь, что рассмотрели все ее последствия.

Установка областей сохранения памяти

Процессор DL06 обеспечивает по умолчанию определенные области сохранения в оперативной памяти (Retentive Range). Эти области сохранения по умолчанию пригодны для многих приложений, но Вы можете изменить их, если Ваше приложение требует дополнительных областей сохранения или вообще никаких областей сохранения. Областями по умолчанию являются:

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

Для установки областей сохранения Вы можете использовать AUX 57 или меню DirectSOFT32. В приложении A содержится подробное описание вспомогательных функций.

ПРЕДУПРЕЖДЕНИЕ. Процессоры DL06 не поставляются вместе с батарейкой. Суперконденсатор поддержит значения при потере питания, но только в течение короткого периода времени, зависящего от окружающих условий. Если время сохранения важно для Вашего приложения, удостоверьтесь, что Вы заказали дополнительную батарею.

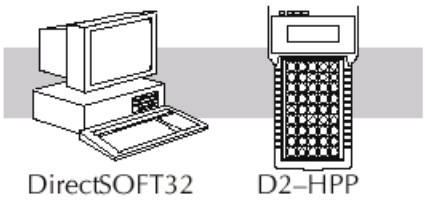
Защита с помощью пароля

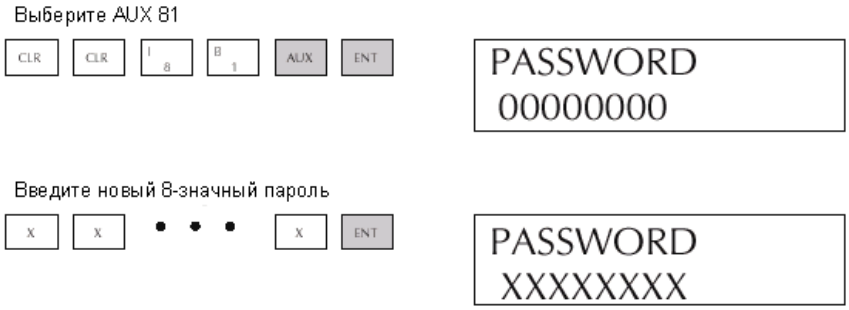
Процессоры DL06 позволяют использовать пароли с целью минимизации риска несанкционированных изменений программ и/или данных. Как только Вы вводите пароль, вы «блокируете» доступ в процессор. Если процессор заблокирован, то Вам необходимо вводить пароль перед тем, как использовать программирующее устройство для изменения каких-либо системных параметров.

Вы можете выбрать 8-значный цифровой пароль. Процессор поставляется с заводским паролем 00000000. Все нули заменяются при установлении защиты с помощью пароля. Если пароль введен, Вы не можете заменить его всеми нулями. Однако после ввода правильного пароля Вы можете изменять пароль, заменить его всеми нулями, то есть снять защиту паролем.

ПРЕДУПРЕЖДЕНИЕ. Постарайтесь не забыть свой пароль. Если Вы забыли свой пароль, то потеряли доступ к процессору. Этот процессор должен быть возвращен изготовителю для снятия пароля (вместе с удалением программы).

Вы можете использовать D2-HPP ручной программатор или DirectSOFT32 для ввода пароля. На следующих схемах показано, как вводить пароль с ручного программатора.





Существует три способа заблокировать процессор после ввода:

1. Если питание отключено, процессор будет автоматически блокирован при последующем обращении.
2. Если Вы ввели пароль из DirectSOFT32, процессор будет автоматически блокирован при последующем обращении, после выхода из DirectSOFT32.
3. Использовать команду AUX81 для блокирования процессора.

Когда Вы захотите использовать DirectSOFT32, Вам надо будет ввести пароль при обращении к блокированному процессору. При вводе AUX82, Вам также будет необходимо ввести пароль.

***ПРИМЕЧАНИЕ**: Процессор DL06 поддерживает многоуровневую защиту паролем. Это позволяет установить пароль, не блокируя коммуникационный порт связанный с операторской панелью. Многоуровневый пароль может быть активизирован, создавав пароль с верхним регистром «А» и семью цифрами (например, A1234567).*

Работа процессора

Для того, чтобы добиться надежного управления аппаратурой или процессом, необходимо хорошо понимать, как работает процессор DL06.

. В данном разделе изучаются следующие четыре сферы работы процессора:

- Операционная система процессора — процессор управляет всеми аспектами системного контроля.
- Режимы работы процессора — основными режимами работы являются: программный режим (Program) и рабочий режим (Run).
- Временные характеристики процессора — обсуждаются две важные временные характеристики: время отклика ввода/вывода и время сканирования процессора.
- Карта распределения памяти процессора — карта распределения памяти процессора отображает адреса различных ресурсов, например, таймеров, счетчиков, входов и выходов.

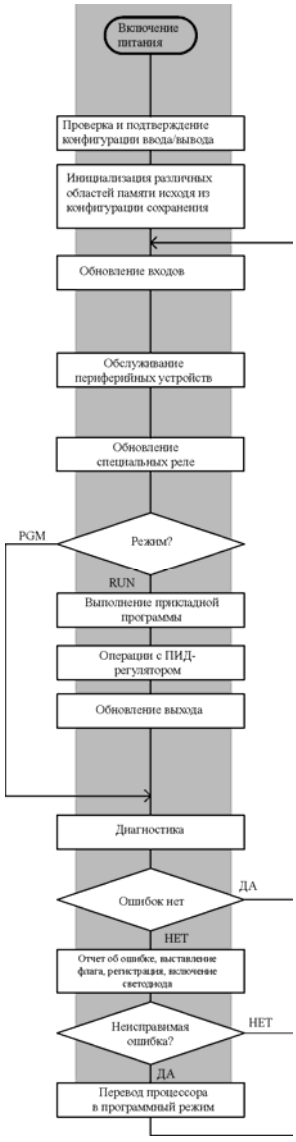
Операционная система процессора DL06

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 процессор 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 пользовательскую программу релейной логики. Сразу после этого выполняется каждый сконфигурированный контур ПИД -регулирования. Затем процессор записывает результаты этих двух задач в соответствующие выходные регистры.

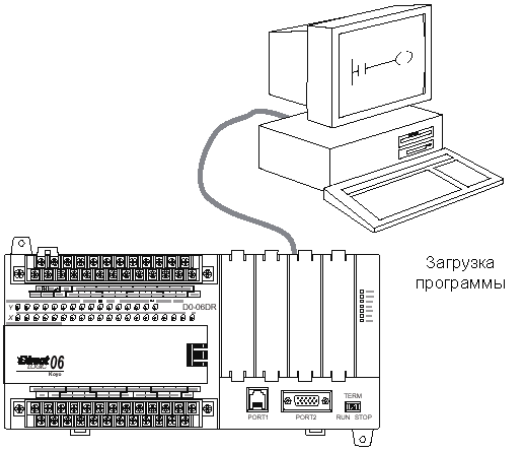
Обнаруженные ошибки имеют два уровня. По исправимым ошибкам формируется сообщение, а процессор остается в текущем режиме. Если обнаруживаются неисправимые ошибки, то процессор переводится в программный режим, а выходы сбрасываются.



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.

Вы можете использовать переключатель режимов в процессоре DL06 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 контура ПИД-регулятора и обновляет входы/выходы системы. В рабочем режиме Вы можете выполнить многие операции. Некоторые из них:

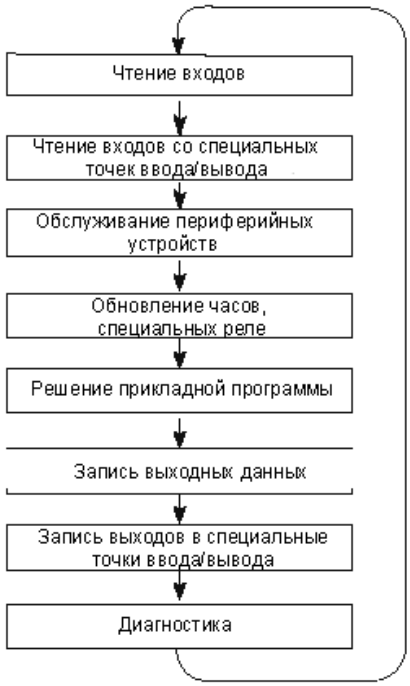
- Контроль и изменение состояния точек ввода/вывода.
- Обновление параметров настройки таймеров/счетчиков.
- Обновление ячеек памяти с переменными.

Работу в рабочем режиме можно разделить на несколько ключевых процессов. Важно понять, как эти зоны влияют на результаты решений Вашей прикладной программы. Для конкретного приложения важными могут быть разные процессы, например, обновление входов/выходов и форсирование переменных.

Вы можете использовать переключатель режимов для выбора работы в рабочем режиме. Или при переключателе, установленном в положении TERM, Вы можете использовать устройство для программирования 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. Подробности смотрите в главе 9.

Режим нормального сканирования



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.

Чтение входов

Процессор считывает состояние всех входов и записывает их в регистры 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 входных данных. Команды немедленного действия считывают состояние входов прямо с модулей ввода/вывода. Однако применение команд немедленного действия удлиняет время сканирования, поскольку процессор должен повторно считывать состояние точек ввода/вывода. Полный список команд немедленного действия приведен в главе 5.

Обслуживание периферийных устройств и форсирование ввода/вывода

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

Существует два базовых типа воздействия, доступных для процессора DL06:

- Нормальное воздействие со стороны периферийного устройства — оно не является постоянным воздействием, действует только в одном цикле сканирования.
- Форсирование бита (Bit override) - поддерживает состояние входа/выхода (или другой бит) постоянно. Применимыми битами являются X, Y, C, T, CT и S. (Эти типы памяти описываются более детально далее в этой главе).

Нормальное воздействие. С помощью этого типа воздействия можно изменять состояние бита дискретного сигнала. Например, Вы желаете включить вход, хотя реально его нет. Это даст Вам возможность изменить состояние точки, которое хранится в регистре отображения. Это значение 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.

Форсирование бита. Форсирование бита можно включать для каждой точки с помощью функции AUX 59 с ручного программатора или с помощью опций меню DirectSOFT32. По существу при форсировании бита процессор не может делать какие-либо изменения в точке дискретного сигнала. Например, если Вы включили форсирование бита для X1, а X1 был временно отключен, то процессор не сможет изменить состояние X1. Это означает, что даже когда X1 включается, процессор не допустит его изменения. Поэтому в данном случае, если X1 используется в программе, то его всегда следует рассматривать как «отключенный». Конечно, если X1 был включен при форсировании бита, то его всегда следует рассматривать как «включенный».

Использование функции форсирования бита принесет Вам некоторую пользу. Нормальное воздействие не блокируется при включении форсирования бита. Например, если Вы включили форсирование бита для Y0, а он был в это время отключен, то процессор не сможет изменить состояние Y0. Однако Вы можете использовать программирующее устройство для изменения этого состояния. Теперь, если вы использовали программирующее устройство для принудительного включения Y0, он останется включенным, а процессор не сможет изменить его состояние. Далее, если вы принудительно отключите Y0, то процессор будет поддерживать Y0 как «отключенный». Процессор никогда не обновит эту точку по результатам работы прикладной программы или по обновлению ввода/вывода, пока форсирование бита не будет снято.

На следующей схеме дается общее представление функции форсирования бита. Заметим, что процессор не обновляет регистр отображения при включенной функции форсирования бита.

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.

Шина связи процессора

Возможна передача данных в процессор и из процессора по системной шине контроллера. Состав этих данных шире, чем стандартное состояние точек ввода/вывода. Этот тип связи может быть использован только в рамках (локального) блока процессора. Это часть исполнительного цикла, используемая для взаимодействия с этими модулями. Процессор выполняет как чтение, так и запись запросов в данном сегменте.

Обновление часов, специальных реле и специальных регистров

DL06 имеет внутренние часы реального времени и календарь, доступные для прикладной программы. Специальные ячейки V-памяти отображают 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.

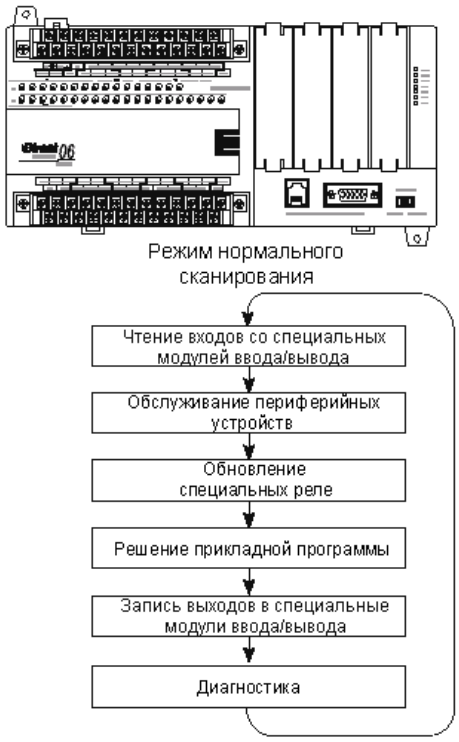
Выполнение прикладных программ

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, пока не встретится команда END. Затем формируется новое отображение выходных данных. Процессор использует регистры отображения выходов обозначаемые, как Y для хранения желаемых воздействий на физические выходы. Обновление состояния физических выходов происходит каждый скан. Кроме того есть команды немедленного вывода, которые позволяют обновлять выходы не дожидаясь соответствующего сегмента цикла. Полный список команд приведен в главе 5.

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 форсировании 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.



Решение уравнений контура ПИД-регулятора

Процессор DL06 может обрабатывать до 8-х 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. Смотрите главу 8 «Организация контуров ПИД-регулятора» для более подробной информации 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 выходы. 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 выходов в специальные модули ввода/вывода

После обновления выходов в каркасе процессор посылает информацию, требуемую всеми установленными специальными модулями.

Диагностика

В данной части цикла сканирования процессор выполняет всю системную диагностику и решает другие задачи, такие как, вычисление времени сканирования, сброс сторожевого таймера. Процессор DL06 автоматически обнаруживает и регистрирует многие ошибочные состояния. В приложении В приводится список кодов различных ошибок, имеющих место в DL06.

Одной их наиболее важных задач диагностики является расчет времени сканирования и управление сторожевым таймером. Процессоры DL06 имеют таймер — «сторож», в котором хранится максимально допустимое время, в течение которого процессор должен закончить прикладной сегмент цикла сканирования. Его значение по умолчанию равно 200 миллисекунд. Если это время будет превышено, то процессор перейдет 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.

Время отклика ввода/вывода

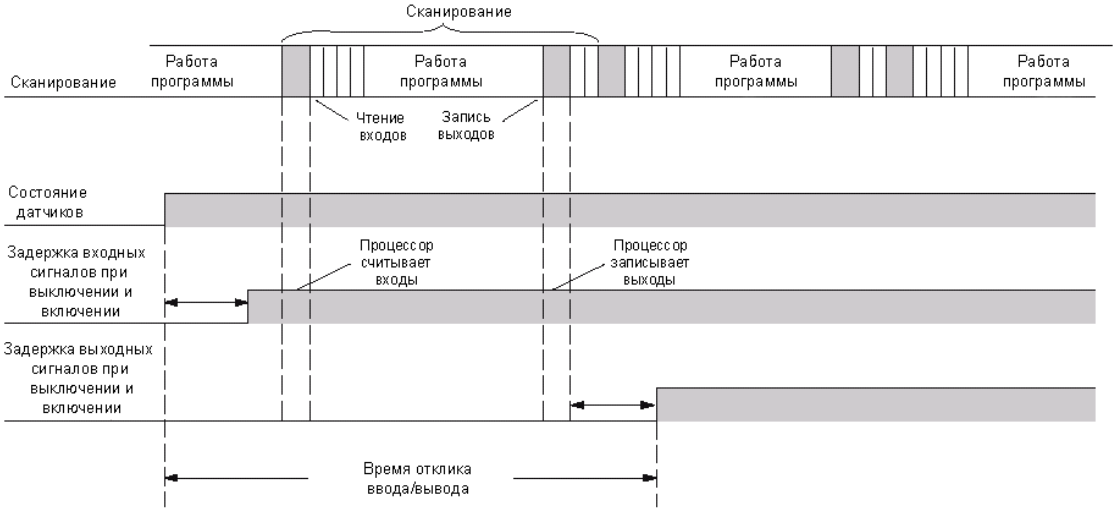
Важно ли быстродействие для Вашего приложения?

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

- Момент (относительно периода сканирования), когда входы меняют свое состояние;
- Время задержки перехода входных схем из состояния выключено в состояние включено;
- Время сканирования процессора;
- Время задержки перехода выходных схем из состояния выключено в состояние включено.

Нормальное минимальное время отклика ввода/вывода

Время отклика ввода/вывода будет наименьшим, когда значения входа изменяется до начала периода опроса входов исполнительного цикла. В этом случае считываются состояния входов, решается прикладная программа и обновляются значения выходов. На следующей диаграмме показан пример распределения времени для этого случая.

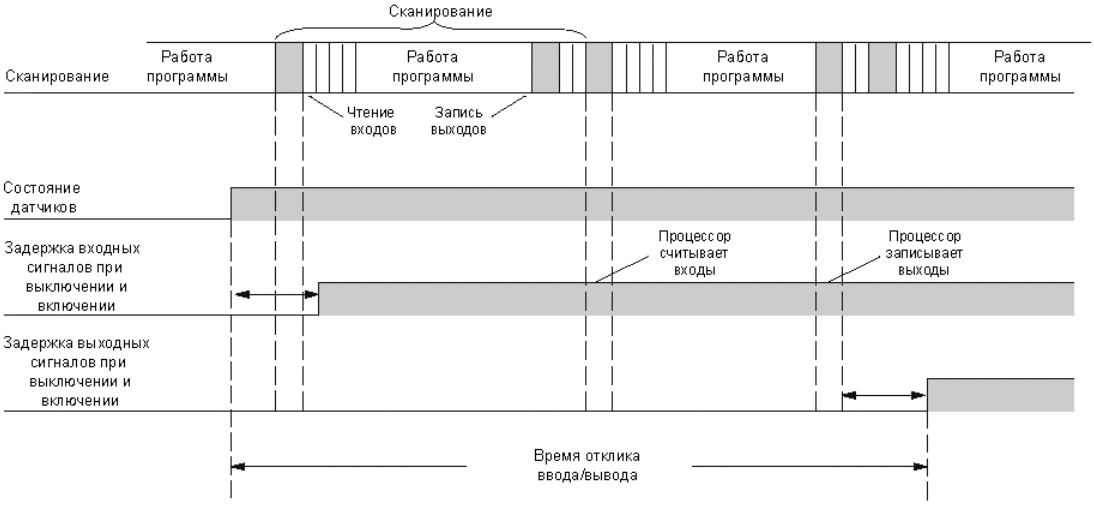


В этом случае Вы можете вычислить время отклика простым суммированием следующих элементов.

задержка входов + время сканирования + задержка выходов = время отклика

Нормальное максимальное время отклика ввода/вывода

Время отклика ввода/вывода будет наибольшим, когда когда значения входа изменяется после начала периода 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.

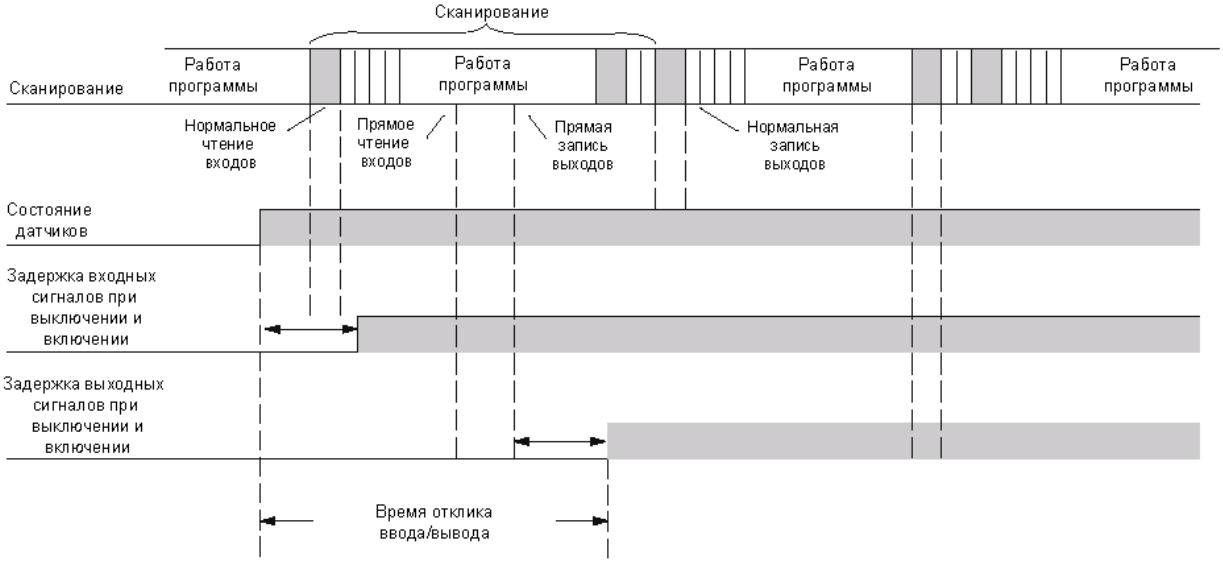
задержка входов + (2 x время сканирования) + задержка выходов = время отклика

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 состояние точек ввода/вывода во время выполнения программы).
- Выбрать режим 50 высокоростного ввода/вывода разработанного специально для высокоскоростных задач. Подробнее в главе 3.
- В режиме 60 изменить временную константу фильтра на 0 мсек для X0, X1, X2 и X3.

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 входов + время выполнения команд + задержка выходов = 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 со входа она использует полученный результат для решения задач, выполняя это в рамках одной команды без обновления значений регистра 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 DirectSOFT32 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 следующие наиболее важные:

- Обновление входов
- Обслуживание периферийных устройств
- Выполнение программы
- Обновление выходов
- Выполнение прерывание по времени

Из всех сегментов на длительность одного Вы можете 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 типа ввода/вывода и периферийных устройств, также влияют на время сканирования. Однако этот выбор, как правило, диктуется приложением.

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

Чтение входов

Время, необходимое для считывания состояния всех встроенных входов - 52.6мкс. Не путайте это время со временем отклика.

Запись выходов

Время, необходимое для записи во встроенные выходы – 41.1 мкс. Не путайте это время со временем отклика.

Обслуживание периферийных устройств

Запросы на связь 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 нет присоединеных периферийных устройствами.

Регистрация запроса (в любое время)		DL06
Отсутствие соединения	мин и макс	0 мкс
Порт 1	Отправление мин /макс	5.8/11.8 мкс
	Получение мин /макс	12.5/25.2 мкс
Порт 2	Отправление мин /макс	6.2/14.3 мкс
	Получение мин /макс	14.2/31.9 мкс
LCD	мин и макс	4.8/49.2 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.

Запрос на обслуживание	DL06
Минимальный	9 мкс
Максимальный в режиме RUN	412 мкс
Максимальный в режиме PROGRAM	2.5 сек

Связь по шине процессора

Некоторые специальные модули могут взаимодействовать с процессором непосредственно через шину процессора. В данной части цикла процессор полностью отрабатывает все коммуникации через шину. Реально необходимое время зависит от типов установленных модулей и типов обрабатываемых запросов.

Обновление часов/календаря, специальных реле, специальных регистров

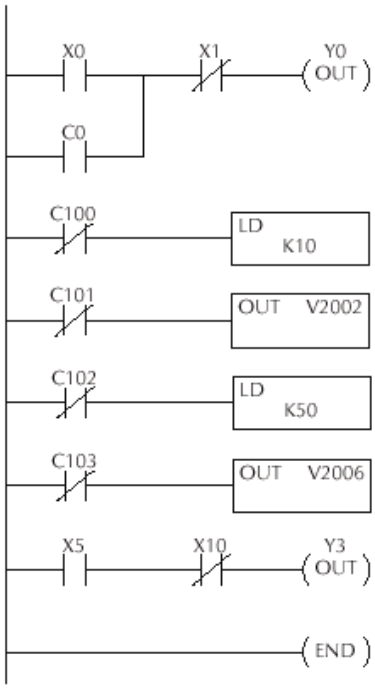
На этом этапе обновляются часы, календарь и специальные реле, результаты загружаются в специальные ячейки V-памяти. Такое обновление выполняется как в рабочем режиме, так и в программном режиме.

Режимы		DL06
Программный режим	Минимум	12.0 мкс
	Максимум	12.0 мкс
Рабочий режим	Минимум	20.0мкс
	Максимум	27.0мкс

Выполнение прикладной программы

Процессор обрабатывает программу сверху (с адреса 0) до команды END. Процессор выполняет программу слева направо и сверху вниз. В каждой цепи вычисляется соответствующий регистр отображения или обновляется ячейка памяти.
Время, необходимое для выполнения прикладной программы, зависит от типа и числа используемых команд и от количества служебных затрат времени.
Вы можете сложить времена выполнения всех команд в Вашей программе, чтобы найти общее время выполнения программы.
Например, время выполнения на DL06 программы, представленной на рисунке, рассчитывается следующим образом.

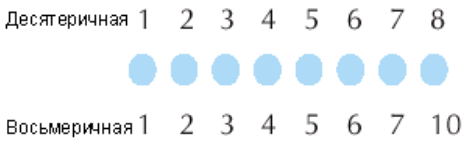
Команда	Время
STR X0	0.67мкс
OR C0	0.51мкс
ANDN X1	0.51мкс
OUT Y0	1.82мкс
STRN C100	0.67мкс
LD K10	9.00мкс
STRN C101	0.67мкс
OUT V2002	9.3мкс
STRN C102	0.67мкс
LD K50	9.00мкс
STRN C103	0.67мкс
OUT V2006	1.82мкс
STR X5	0.67мкс
ANDN X10	0.51мкс
OUT Y3	1.82мкс
END	12.80мкс
Сумма	51.11мкс



Системные издержки DL06

МИНИМУМ 746.2МКС

Максимум 4352.4мкс

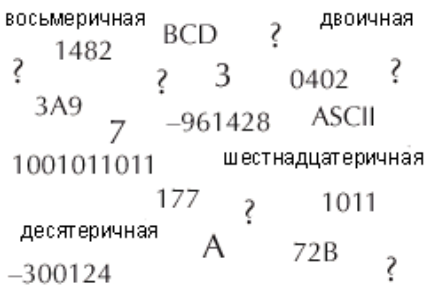


ОБЩЕЕ ВРЕМЯ = (ВРЕМЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ + СИСТЕМНЫЕ ИЗДЕРЖКИ) X 1.18

Сама программа выполняется за 51.11мкс во время каждого скан-цикла. DL06 тратит 0.18мс на обработку внутреннего прерывания от таймера каждую миллисекунду. Общее время сканирования вычисляется добавлением к времени выполнения программы времени на системные издержки и умножения результата на 1.18. «Системные издержки» включают обновление входов/выходов и диагностические задачи. Из-за флуктуации времени на «системные издержки» общее время сканирования немного изменяется от цикла к циклу.

Команды управления программой. Процессор DL06 имеет дополнительные команды, которые могут изменить ход выполнения программы. В эти команды входят циклы FOR/NEXT, подпрограммы, программы прерывания. Эти команды прерывает нормальный процесс выполнения программы и могут влиять на время выполнения программы. В главе 5 приводится детальная информация о том, как работают команды этого типа.

Системы счисления в ПЛК



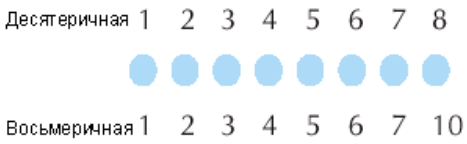
Если Вы — новый пользователь ПЛК, выделите время на изучение того, как ПЛК работает с числами. Вы узнаете, что каждый изготовитель ПЛК имеет собственное соглашение об использовании чисел в ПЛК. Выделите время, чтобы познакомиться с тем, как используются числа в ПЛК DirectLogic. Эта информация относится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

Как и каждый 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.

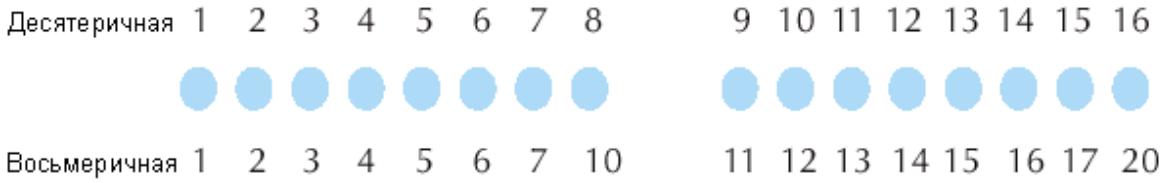
Ресурсы ПЛК

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 по 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).

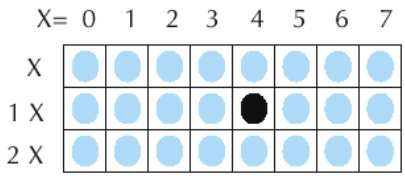


На рисунке ниже приведены две группы по восемь кружков. В восьмеричной системе мы имеем «20» предметов, что означает 2 группы по восемь плюс 0 единичных. Нельзя говорить «двадцать», надо говорить «два - ноль восьмеричных». В этом четкое различие между системами счисления.

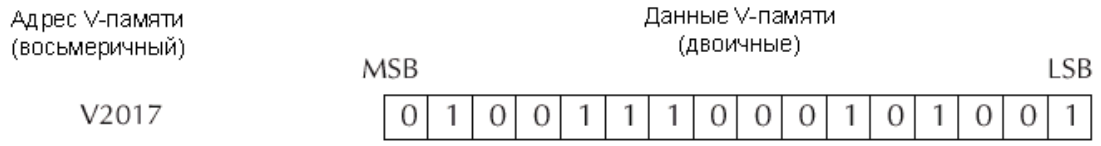


После определения того, как считаются ресурсы, следует перейти к тому, как осуществляется доступ ПЛК к ресурсам (это не одно и то же). Команды процессора при доступе к ресурсам ПЛК используют восьмеричные адреса. Восьмеричные адреса тоже имеют восьмеричное значение, за исключением того, что счет начинается 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.

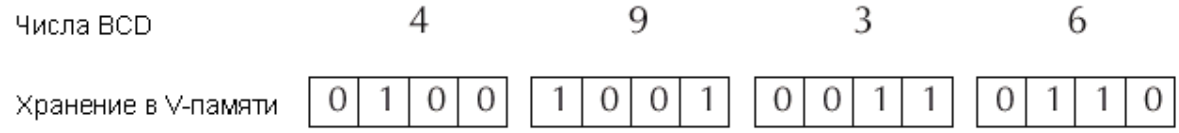


V-память



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 V-памяти есть 16-битовай двоичный код, 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 преобразовывает, ни переносит данные.

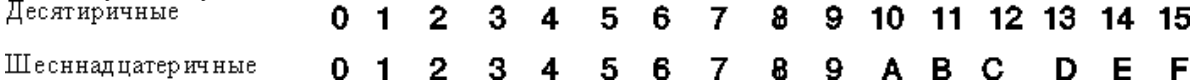
Двоично-десятичные числа



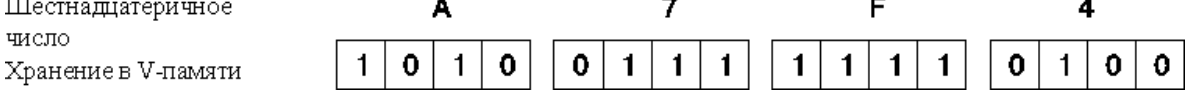
Поскольку люди считают в десятичной системе, ввод и представление данных ПЛК также делает в десятичной форме (через интерфейсы операторов). Но компьютеры более эффективны при использовании строго двоичных чисел. Компромиссом между этими двумя системами является двоично-десятичное (BCD-Binary Coded Decimal) 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 DirectSOFT32 и ручной программатор позволяют вводить и просматривать данные в BCD.

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 необходимы 16 цифр. Для дополнения десятичных цифр от 0 до 9 используются буквы от A до F, как показано ниже.



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 данных чувствительных элементов и др. Шестнадцатеричная система является удобным для человека

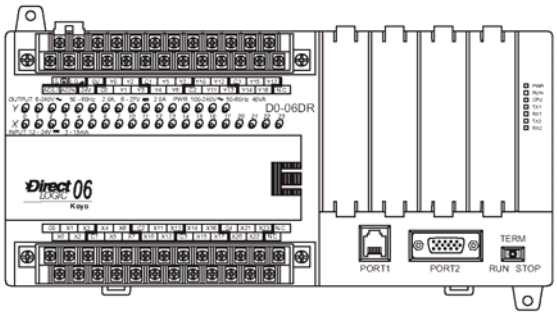


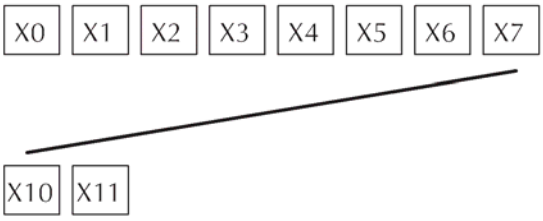
Карта памяти

Работая с любой системой на базе ПЛК, обычно требуется обрабатывать множество различных видов информации. Сюда включаются состояние входных устройств, состояние выходных устройств, различные элементы таймеров, счетчиков, и т.д. Важно понять, как система представляет и сохраняет различные типы данных. Например, необходимо знать, как система идентифицирует входные точки и точки вывода, слова данных, и т.д. Следующие абзацы обсуждают различные типы памяти, используемые в микроконтроллере DL06. Краткий обзор карты памяти процессора следует за описаниями типов памяти.

Восьмиричная система исчисления.

Все ячейки памяти и ресурсы контроллера пронумерованы в восьмеричной системе исчисления. Например, диаграмма показывает, как восьмеричная система исчисления работает для входных дискретных каналов. Общеизвестно, что восьмеричная система не содержит чисел с цифрами 8 или 9.



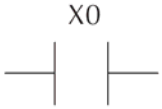


Дискретные и двухбайтные ячейки памяти.

Поскольку рассматриваются различные типы памяти, то вы должны заметить, что в DL06 существуют два типа памяти, дискретная и двухбайтная (слово) память. Дискретная память – это один бит, который может принимать значение 1 или 0. Двухбайтную память называют V-памятью, она является 16-разрядной ячейкой памяти обычно используемой, для манипуляций с данными/числами, запоминания данных / чисел, и т.д.

Некоторая информация автоматически сохраняется в V-памяти. Например, текущее значение таймера.

Дискретная ячейка - ВКЛ или ВЫКЛ, 1 бит

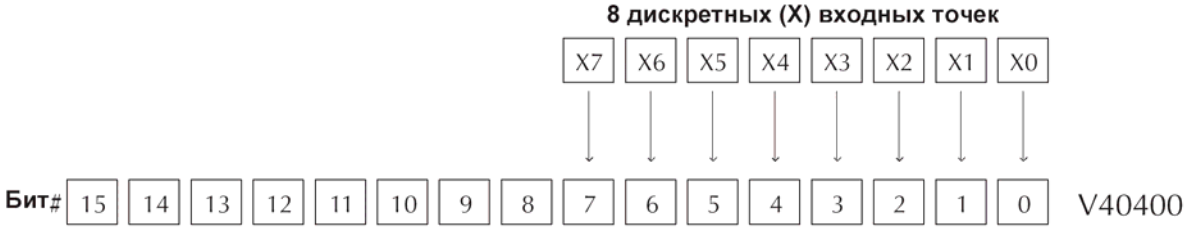


Двухбайтовая ячейка - 16бит

0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	1

Ячейки V-памяти для дискретной области памяти

Дискретная область памяти необходима для отображения входов, выходов, управляющих реле, специальных реле, стадий, битов состояния таймера и битов состояния счетчика. Однако, Вы можете также обращаться к битовым типам данных как к слову V-памяти. Каждая ячейка V-памяти содержит 16 последовательных дискретных бит. Например, следующий рисунок показывает, как точки ввода X отображаются в ячейках V-памяти.



Эти дискретные области памяти и соответствующие им диапазоны V-памяти перечислены в таблице распределения памяти для микроконтроллера DL06 на следующих страницах.

Входные точки (Тип данных X)

Точки дискретных входных сигналов являются типом данных X. Имеются 20 точек дискретных входных сигналов и 512 адресов дискретных входных сигналов, доступных ЦПУ DL06. В этом примере, точка вывода Y0 будет включена, когда подано питание на точку ввода X0.

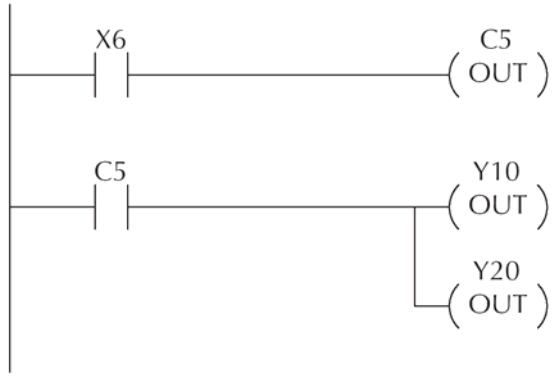
Выходные точки (Тип данных Y)

Дискретные точки вывода и являются типом данных Y. Имеются 16 дискретных выходов и 512 дискретных адресов вывода, доступные процессору DL06. В этом примере, выход Y1 будет включен, когда на вход X1 будет подано питание.

Управляющие реле (Тип данных C)

Управляющие реле – это дискретные биты, обычно используемые, для управления программой. Управляющие реле не представляют собой реальное устройство, то есть они не могут быть физически привязаны к переключателям, выходам обмоток, и т.д. Это внутренние переменные процессора. Из-за этого, управляющие реле могут программироваться как дискретные входные сигналы или как дискретные выходы. Они расположены в дискретной области памяти (C) или в соответствующим слове, которое содержит 16 последовательных дискретных бит.

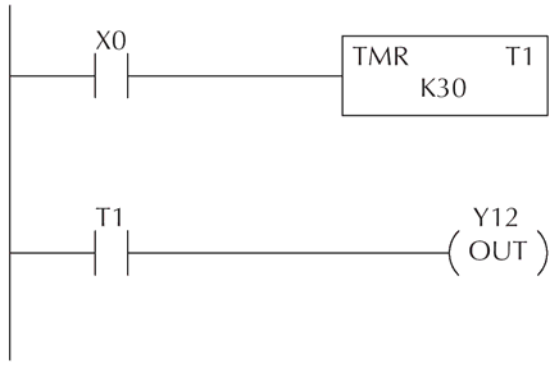
В этом примере, ячейка памяти C5 будет включена, при включении входа X6. Вторая ветвь программы показывает простой пример того, как использовать реле управления как входную переменную.



Таймеры и биты состояния таймеров (Тип данных T)

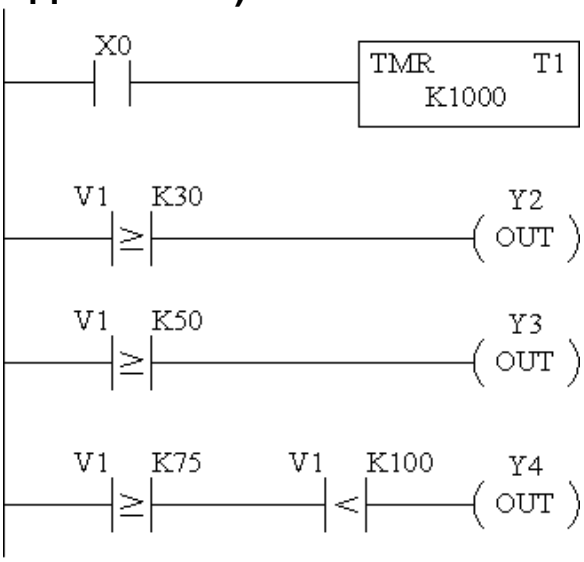
Биты состояния таймера отражают связь между текущим значением и предустановленным значением определенного таймера. Бит состояния таймера будет включен, когда текущее значение тайиера является равным или большим чем предустановленное значение соответствующего таймера.

Когда вход X0 включается, таймер T1 начнет отсчет. Когда таймер достигает предустановленного значения в 3 секунды (K30) бит состояние таймера T1 включается. Затем T1 включает вывод Y12. Выключение X0 сбрасывает таймер.



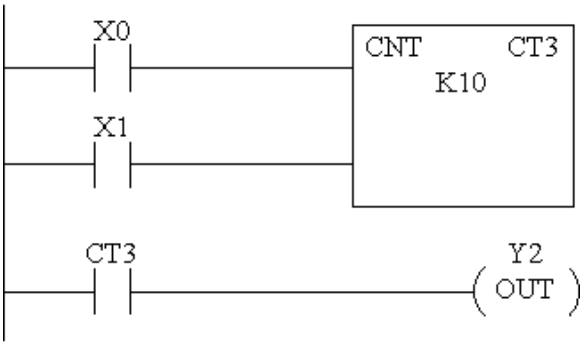
Текущее значение таймера (Тип данных V)

Как было упомянуто ранее, некоторая информация автоматически сохраняется в V-памяти. Это верно для текущих значений таймеров. Например, V0 сохраняет текущее значение для Таймера 0, V1 сохраняет текущее значение для Таймера 1, и т.д. Они могут также быть обозначены как TA0 (Накопленное значение таймера) для Таймера 0, и TA1 для Таймера 1.
Сделано так для большей гибкости программирования.
Пример показывает, как Вы можете использовать контакты сравнения, чтобы контролировать различные временные интервалы одного таймера.



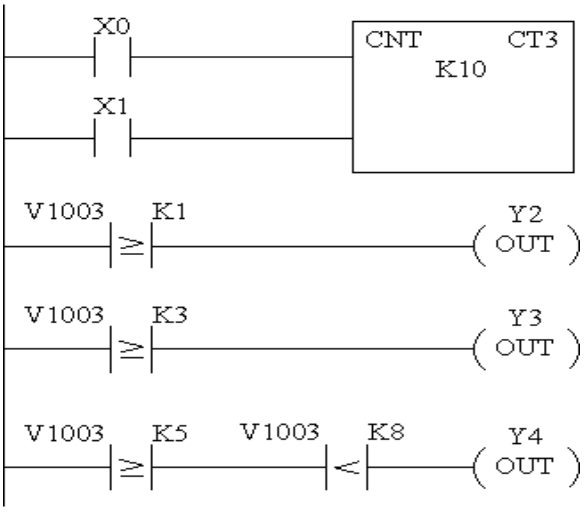
Счетчики и биты состояния счетчика (тип данных CT)

Биты состояния счетчиков отражают отношение между текущим и предварительно установленным значением определенного счетчика. Бит состояния счетчика будет «включен», когда текущее его значение станет равным или большим, чем предварительно установленное значение соответствующего счетчика.
Каждый раз, когда контакт X0 переходит из состояния «выключен» в состояние «включен», счетчик увеличивается на единицу. Если X1 включается, то счетчик возвращается в состояние «ноль». Когда счетчик достигает установленных 10 подсчетов (K = 10), то контакт состояния счетчика CT3 «включается». Когда «включается» CT3, включается и выход Y2.



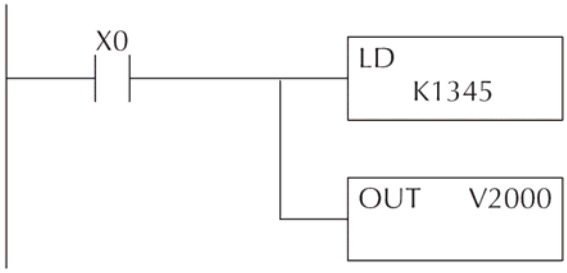
Текущее значение счетчика (тип данных V)

Точно так же как и таймеры, текущие значения счетчиков также автоматически запоминаются в V-памяти. Например, V1000 сохраняет текущее значение Счетчика CT0, V1001 сохраняет текущее значение Счетчика CT1 и т. д. Они могут также быть обозначены как CTA0 (Суммирующий Счетчик) для Счетчика 0 и CTA01 для Таймера1.
Пример показывает, как вы можете использовать относительные контакты для контроля значений счетчика



Двухбайтная (пословная) память (тип данных V)

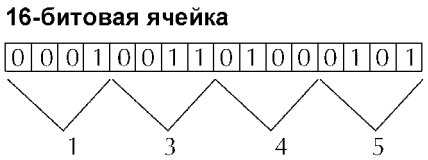
Двухбайтная память относится к V-памяти (память переменных) и представляет собой 16-битовую(слово) ячейку, обычно используемую для манипулирования и хранения данных/чисел и др. Некоторая информация автоматически сохраняется в V-памяти. Например, текущие значения таймера запоминаются в V-памяти. В приведенном примере показано, как четырехзначная двоично-десятичная константа загружается в накапливающий регистр и затем запоминается в ячейке V-памяти

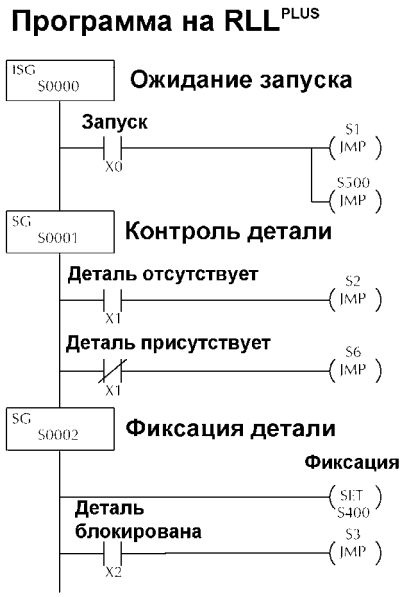


Стадии (тип данных S)

Стадии используются в программах RLLPLUS для создания структурированных программ, подобных блок-схемам. Каждый этап программы обозначает программный сегмент. Когда программный сегмент, или этап, активны, выполняется логическая схема данного сегмента. Если этап «отключен», или не активен, логическая схема сегмента не выполняется, а ЦПУ переходит к следующему активному этапу. (См. Главу 7 для более детального описания программирования на RLLPLUS).

Каждый этап также имеет дискретный разряд индикации состояния, который можно использовать как вход для указания того, активен ли этап или нет. Если этап активен, то разряд индикации состояния «включен». Если этап не активен, то разряд индикации состояния «выключен». Этот разряд индикации состояния может «включаться» и «выключаться» также другими командами, такими как команды SET или RESET. Это дает вам возможность легко управлять этапами 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 мс. и отключаться на 50 мс., так как SP5 является специальным реле, которое находится в состоянии «включено» в течение 50 мс. и в состоянии «выключено» также в течение 50 мс.

SP4:Таймер 1 секунда
SP5:Таймер 100 милисекунд
SP6:Таймер 50 милисекунд

Системная V-память DL06

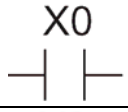
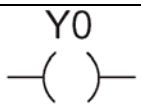
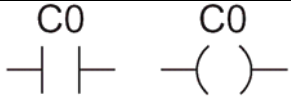
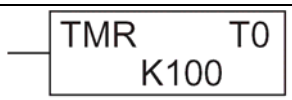
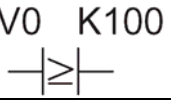
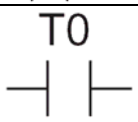
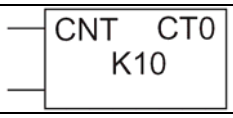
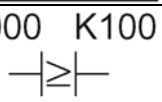
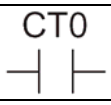
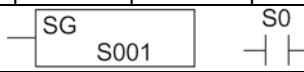
Системные параметры и размещение данных по умолчанию (тип данных V)

В ПЛК DL06 резервируются отдельные адреса V-памяти для сохранения системных параметров или некоторых типов данных системы. Эти ячейки памяти всегда сохраняют свое назначение, например, коды ошибок, данные высокоскоростного ввода/вывода, и другие типы информации по инизиализации системы.

системная V-память	Описание содержимого	Значение по умолчанию/диапазон
V700-V707	Указывает ячейку V-памяти для модуля в слоте 1	Отсутствует
V710-V717	Указывает ячейку V-памяти для модуля в слоте 2	Отсутствует
V720-V727	Указывает ячейку V-памяти для модуля в слоте 3	Отсутствует
V730-V737	Указывает ячейку V-памяти для модуля в слоте 4	Отсутствует
V3630–V3707	Ячейки (по умолчанию) для предустановленных значений счетчика 1 или импульсной защелки высокоскоростного ввода/вывода	Отсутствует
V3710-V3767	Ячейки (по умолчанию) для предустанов. значений счетчика 2	Отсутствует
V7620–V7627	Ячейки для параметров панели оператора DV-1000.	
V7620	Устанавливает адрес ячейки V-памяти, которая содержит значение.	V0 – V3760
V7621	Устанавливает адрес ячейки V-памяти, которая содержит сообщение.	V0 – V37601 - 32
V7622	Устанавливает общее число (1-32) ячеек V-памяти, которые нужно отобразить.	V0 – V3760
V7623	Устанавливает ячейку V-памяти, которое содержит числа, которые нужно отобразить.	V0 – V3760
V7624	Устанавливает ячейку V-памяти, которое содержит символьный код, который нужно отобразить.	V-память для X, Y, или C0,
V7625	Содержит номер функции, который может быть присвоен каждой клавише.	1, 2, 3,
V7626	Режим работы после включения питания.	12,
V7627	Изменение предустановленных значений.	знач. по умолч. = 0000
V7630	Начальная ячейка для многошаговых предварительных установок канала 1. Значение "по умолчанию" – 3630 указывает, что первое значение должно быть получено из V3630. Так как имеются 24 доступные предварительно заданные установки, диапазон по умолчанию - V3630 - V3707. Вы можете изменять начальную точку при необходимости.	Значение по умолчанию: V3630 Диапазон: V0- V3710
V7631	Начальная ячейка для многошаговых предварительных установок канала 2. Значение "по умолчанию" – 3710 указывает, что первое значение должно быть получено из V3710. Так как имеются 24 доступные предварительно заданные установки, диапазон по умолчанию - V3710 - V3767. Вы можете изменять начальную точку при необходимости.	Значение по умолчанию: V3710 Диапазон: V0- V3710
V7632	Регистр настройки для импульсного вывода	Отсутствует
V7633	Устанавливает желательный код функции высокоскоростного счетчика, прерывания, импульсной защелки, последовательности импульсов, и входной фильтрации. Ячейка может использоваться, чтобы установить режим RUN при включении питания.	По умолчанию: 0060 Младший байт: 10 –счетчик 20 –квадратурный счетчик 30 – импульсный вывод 40 –Прерывание 50 – импульсная защелка 60 –фильтрованное значение Старших байт: Биты 8–11, 14, 15 - не используется, Бит 13: режим RUN при включении питания, только если перключатель режима в положении TERM. Бит 12: батарея подключена
V7634	Регистры настройки X0 для высокоскоростного Ввода/Вывода.	По умолчанию: 1006
V7635	Регистры настройки X1 для высокоскоростного Ввода/Вывода	По умолчанию: 1006
V7636	Регистры настройки X2 для высокоскоростного Ввода/Вывода	По умолчанию: 1006
V7637	Регистры настройки X3 для высокоскоростного Ввода/Вывода.	По умолчанию: 1006
V7640	Начальные адреса Таблиц ПИД-регуляторов (PID Loop table)	V1200-V7377,V10000-V17777

системная V-память	Описание содержимого	Значение по умолчанию/диапазон
V7641	Число ПИД- контуров	1-8
V7642	Код ошибки – ошибка размещения ячейки V-памяти таблицы контура	
V7643-V7647	Резерв	
V7650	Порт 2: адрес V-памяти для ASCII протокола	V1200 – V7377 V10000 - V17777
V7653	Порт 2: символы кода завершения передачи ASCII протокола	
V7655	Порт 2: Настройки протокола, тайм-аут ответа, и время задержки ответа	
V7656	Порт 2: Установка номера станции, скорости передачи, количество стоповых бит, и вид контроля четности	
V7657	Порт 2: Код завершения, извещающий о завершении настройки параметров	
V7660	Установка управления сканированием : сохраняет режим управления сканированием.	
V7661	Установка таймера превышения: Вычисляет время, на которое фактическое время сканирования превышает установленное время.	
V7662–V7717	Резерв	
V7720–V7722	Ячейки для параметров интерфейса оператора DV- 1000.	
V7720	Указатель Именованной заданной величины таймера	
V7721	Указатель Именованной заданной величины счетчика	
V7722	Размер блока для Именованных уставок таймера в старших байтах, и размер блока для Именованных уставок счетчика в младших байтах	
V7723–V7737	Резерв	
V7740	Порт 1 и порт 2: настройка таймера перезагрузки связи	По умолчанию: 3030
V7741–V7746	Резерв	
V7747	Ячейка содержит 10мс счетчик (0-99) . Значение ячейки увеличиваются на единицу каждые 10 мс.	
V7750	Резерв	
V7751	Код ошибки в сообщении об ошибке - хранится 4-значный код, используемый с командой FAULT при выполнении этой команды	
V7752	Ошибка конфигурации ввода – вывода: Текущий идентификационный код слота с ошибкой	
V7753	Ошибка конфигурации ввода – вывода: Прежний идентификационный код слота с ошибкой	
V7754	Ошибка конфигурации ввода – вывода: номер слота с ошибкой	
V7755	Код ошибки-хранится код фатальной ошибки.	
V7756	Код ошибки-хранится код основной ошибки.	
V7757	Код ошибки-хранится код второстепенной ошибки.	
V7760–V7762	Резерв	
V7763	Адрес программы с синтаксической ошибкой	
V7764	Код Синтаксической ошибки	
V7765	Счетчик циклов — сохраняет общее число циклов программы, которые произошли после последнего перехода из програмного режима в раюочий режима.	
V7766	Содержит текущее число секунд (00-59)	
V7767	Содержит текущее число минут (00-59)	
V7770	Содержит текущее число часов (00-23)	
V7771	Содержит текущий день недели (Mon., Tues.,. Wed, и т.д.)	
V7772	Содержит текущее день месяца (01, 02, и т.д.)	
V7773	Содержит текущий месяц (от 01 до 12)	
V7774	Содержит текущий год (от 00 до 99)	
V7775	Скан-контроль: - сохраняет текущее время сканирования (миллисекунды).	
V7776	Скан-контроль - сохраняет минимальное время сканирования, которое произошло начиная с последнего перехода из программного режима в рабочий	
V7777	Переход от режима программирования к рабочему (мс).	
V37700-V37737	Для удаленного ввода – вывода (Remote I/O)	

Карта памяти DL06

Тип памяти	Указатель дискретной памяти (восьмиричный)	Указатель двухбайтной памяти (восьмиричный)	Количество Десятичное	Символ
Входные точки	X0 – X777	V40400 - V40437	512	
Выходные точки	Y0 – Y777	V40500 – V40537	512	
Управляющие реле	C0 – C1777	V40600 - V40677	1024	
Специальные реле	SP0 – SP777	V41200 – V41237	512	
Таймеры	T0 – T377	V41100 – V41117	256	
Текущие значения таймеров	-	V0 – V377	256	
Биты состояния таймеров	T0 – T377	V41100 – V41117	256	
Счетчики	CT0 – CT177	V41140 – V41147	128	
Текущие значения счетчиков	-	V1000 – V1177	128	
Биты состояния счетчиков	CT0 – CT177	V41140 – V41147	128	
Слова данных (см. приложение F)	-	V400-V677 V1200 – V7377 V10000 - V17777	192 3200 4096	Нет обозначения, используется со многими командами
Слова данных EEPROM (см. приложение F)	-	V7400 – V7577	128	Нет обозначения, используется со многими командами. Данные могут перезапи-сываться в EEPROM по крайней мере 100000 раз.
Стадии	S0 – S1777	V41000 – V41017	1024	
Удаленный ввод/вывод	GX0-GX3777 GY0-GY3777	V40000-V40177 V40200-V40377	2048 2048	
Системные параметры	-	V700-V777 V7600 – V7777 V36000-V37777	64 128 1024	Нет обозначения, используется для различных целей

Системы на базе DL06 ограничены 20-ю дискретными входами и 16 дискретными выходами, представленными встроенными аппаратными средствами контроллера, однако существуют по 512 адресов для отображения дискретных входов и выходов.

Карта битового отображения входов X/выходов Y

Эта таблица показывает связь каждого дискретного входа и выхода, с адресами бит V-памяти для двадцати входов и 16 выходов самого DL06 и дополнительных 64 входов и 64 выходов, которые Вы можете установить при помощи модулей ввода/вывода. Фактически, доступны точки – от X0 до X777 (V40400 - V40437) и от Y0 до Y777 (V40500 -V40537).

Старший бит			Входные (X) и выходные (Y) точки DL06													Младший бит	Адрес	Адрес
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0	входа X	выхода Y
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
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

Карта отображения битов Управления/Состояния стадий.

Эта таблица показывает связь каждого бита управления стадией с адресами битов V-памяти.

Старший бит								Биты управления стадиями(S) DL06				Младший бит				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

Старший бит				Биты управления стадиями(S) DL06									Младший бит			Адрес
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	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
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
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	1232	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

Карта памяти управляющих реле

Эта таблица показывает связь каждого управляющего реле с адресами битов V-памяти.

Старший бит								Управляющие реле(C) DL06							Младший бит	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
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

Старший бит					Управляющие реле(C) DL06							Младший бит				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	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

Карта битов состояния таймеров

Эта таблица показывает связь каждого контакта таймера с адресами битов V-памяти.

Старший бит					Контакты таймера (T) DL06								Младший бит				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	V41100
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	V41101
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	V41102
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	V41103
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	V41104
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	V41105
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	V41106
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	V41107
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

Карта битов состояния счетчика

Эта таблица показывает связь каждого контакта счетчика с адресами битов V-памяти.

Старший бит					Контакты таймера (T) DL06								Младший бит				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	V41140
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	V41141
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	V41142
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	V41143
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	V41144
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	V41145
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	V41146
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	V41147

Карта битов удаленного ввода/вывода

Эта таблица показывает связь каждой точки удаленного ввода/вывода с адресами битов V-памяти.

Старший бит			Точки удаленного ввода(GX)/вывода(GY)										Младший бит				Адрес	Адрес
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0		GX	GY
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		V40004	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		V40007	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

Старший бит								Точки удаленного ввода(GX)/вывода(GY)							Младший бит	Адрес	Адрес
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0	GX	GY
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

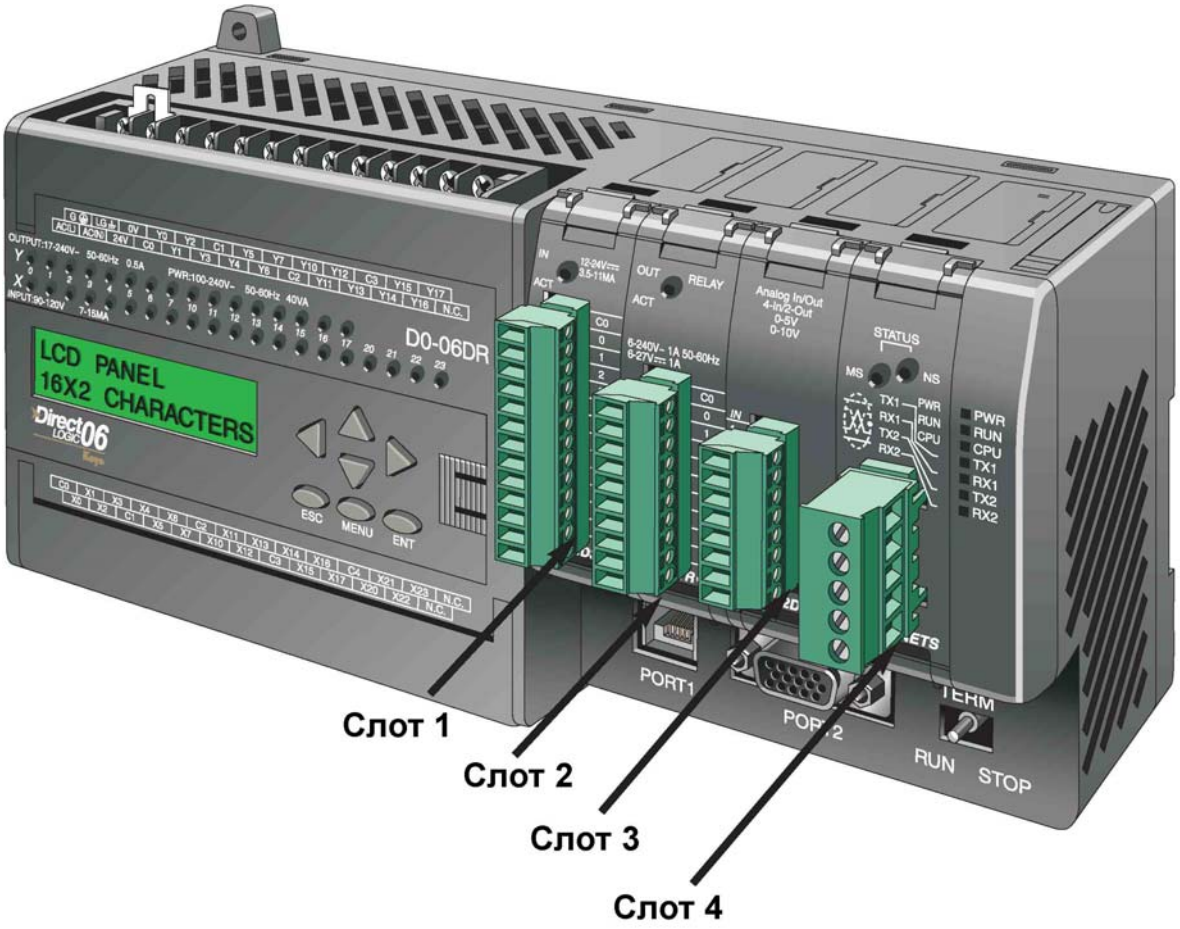
Старший бит								Точки удаленного ввода(GX)/вывода(GY)							Младший бит	Адрес	Адрес
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0	GX	GY
2017	2016	2015	2014	2013	2012	2011	2010	2007	2006	2005	2004	2003	2002	2001	2000	V40100	V40300
2037	2036	2035	2034	2033	2032	2031	2030	2027	2026	2025	2024	2023	2022	2021	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	2736	2735	2734	2733	2732	2731	2730	V40136	V40336
2777	2776	2775	2774	2773	2772	2771	2770	2767	2766	2765	2764	2763	2762	2761	2760	V40137	V40337

Старший бит								Точки удаленного ввода(GX)/вывода(GY)							Младший бит	Адрес	Адрес
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0	GX	GY
3017	3016	3015	3014	3013	3012	3011	3010	3007	3006	3005	3004	3003	3002	3001	3000	V40140	V40340
3037	3036	3035	3034	3033	3032	3031	3030	3027	3026	3025	3024	3023	3022	3021	3020	V40141	V40341
3057	3056	3055	3054	3053	3052	3051	3050	3047	3046	3045	3044	3043	3042	3041	3040	V40142	V40342
3077	3076	3075	3074	3073	3072	3071	3070	3067	3066	3065	3064	3063	3062	3061	3060	V40143	V40343
3117	3116	3115	3114	3113	3112	3111	3110	3107	3106	3105	3104	3103	3102	3101	3100	V40144	V40344
3137	3136	3135	3134	3133	3132	3131	3130	3127	3126	3125	3124	3123	3122	3121	3120	V40145	V40345
3157	3156	3155	3154	3153	3152	3151	3150	3147	3146	3145	3144	3143	3142	3141	3140	V40146	V40346
3177	3176	3175	3174	3173	3172	3171	3170	3167	3166	3165	3164	3163	3162	3161	3160	V40147	V40347
3217	3216	3215	3214	3213	3212	3211	3210	3207	3206	3205	3204	3203	3202	3201	3200	V40150	V40350
3237	3236	3235	3234	3233	3232	3231	3230	3227	3226	3225	3224	3223	3222	3221	3220	V40151	V40351
3257	3256	3255	3254	3253	3252	3251	3250	3247	3246	3245	3244	3243	3242	3241	3240	V40152	V40352
3277	3276	3275	3274	3273	3272	3271	3270	3267	3266	3265	3264	3263	3262	3261	3260	V40153	V40353
3317	3316	3315	3314	3313	3312	3311	3310	3307	3306	3305	3304	3303	3302	3301	3300	V40154	V40354
3337	3336	3335	3334	3333	3332	3331	3330	3327	3326	3325	3324	3323	3322	3321	3320	V40155	V40355
3357	3356	3355	3354	3353	3352	3351	3350	3347	3346	3345	3344	3343	3342	3341	3340	V40156	V40356
3377	3376	3375	3374	3373	3372	3371	3370	3367	3366	3365	3364	3363	3362	3361	3360	V40157	V40357
3417	3416	3415	3414	3413	3412	3411	3410	3407	3406	3405	3404	3403	3402	3401	3400	V40160	V40360
3437	3436	3435	3434	3433	3432	3431	3430	3427	3426	3425	3424	3423	3422	3421	3420	V40161	V40361
3457	3456	3455	3454	3453	3452	3451	3450	3447	3446	3445	3444	3443	3442	3441	3440	V40162	V40362
3477	3476	3475	3474	3473	3472	3471	3470	3467	3466	3465	3464	3463	3462	3461	3460	V40163	V40363
3517	3516	3515	3514	3513	3512	3511	3510	3507	3506	3505	3504	3503	3502	3501	3500	V40164	V40364
3537	3536	3535	3534	3533	3532	3531	3530	3527	3526	3525	3524	3523	3522	3521	3520	V40165	V40365
3557	3556	3555	3554	3553	3552	3551	3550	3547	3546	3545	3544	3543	3542	3541	3540	V40166	V40366
3577	3576	3575	3574	3573	3572	3571	3570	3567	3566	3565	3564	3563	3562	3561	3560	V40167	V40367
3617	3616	3615	3614	3613	3612	3611	3610	3607	3606	3605	3604	3603	3602	3601	3600	V40170	V40370
3637	3636	3635	3634	3633	3632	3631	3630	3627	3626	3625	3624	3623	3622	3621	3620	V40171	V40371
3657	3656	3655	3654	3653	3652	3651	3650	3647	3646	3645	3644	3643	3642	3641	3640	V40172	V40372
3677	3676	3675	3674	3673	3672	3671	3670	3667	3666	3665	3664	3663	3662	3661	3660	V40173	V40373
3717	3716	3715	3714	3713	3712	3711	3710	3707	3706	3705	3704	3703	3702	3701	3700	V40174	V40374
3737	3736	3735	3734	3733	3732	3731	3730	3727	3726	3725	3724	3723	3722	3721	3720	V40175	V40375
3757	3756	3755	3754	3753	3752	3751	3750	3747	3746	3745	3744	3743	3742	3741	3740	V40176	V40376
3777	3776	3775	3774	3773	3772	3771	3770	3767	3766	3765	3764	3763	3762	3761	3760	V40177	V40377

Размещение модулей.

Нумерация слотов.

DL06 имеет четыре слота, которые пронумерованы следующим образом:



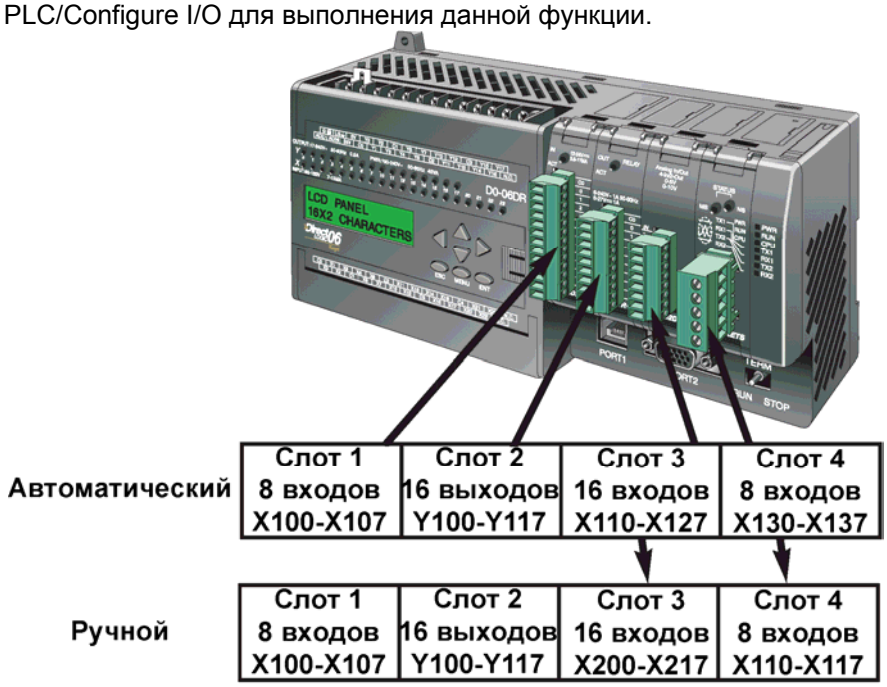
Автоматическая конфигурация Ввода / вывода

Процесор микроконтроллера DL06 автоматически обнаруживает любые установленные модули Ввода - вывода (включая специальные модули) при включении питания, и устанавливает правильную конфигурацию ввода/вывода и адресацию. Это относится к локальному вводу/выводу, а также к дополнительным модулям. В большинстве случаев, изменение конфигурации никогда не понадобится.

Адреса ввода/вывода используют восьмеричную нумерацию, начинающуюся с X100 и Y100 для слота, следующего за процессорным блоком.

Адреса назначаются группами по 8 или 16 точек в зависимости от числа каналов ввода/вывода для модуля. Входные и выходные дискретные модули могут размещаться в любом порядке, однако могут иметься ряд ограничений на размещение специальных модулей. Следующая диаграмма показывает соглашение о нумерации каналов ввода/вывода, для данного примера. И Ручной Программатор и DirectSOFT32 обеспечивают работу с функцией AUX, которая разрешает автоматическое конфигурирование ввода/вывода.

Например, с Ручного Программатора можно выполнить функцию автоматической конфигурации AUX 46, которая разрешает, чтобы процессор исследовал установленные модули и определил конфигурацию и адресацию ввода - вывода. В DirectSOFT32, выберите пункт меню PLC/Configure I/O для выполнения данной функции.



Ручная конфигурация ввода/вывода

Возможно, это Вам никогда не понадобится, но процессоры DL06, допускают ручное назначение адресов для любого слота ввода - вывода. Вы можете вручную изменить автоматическую конфигурацию, и установить произвольные номера ввода/вывода. Например, два смежных входных модуля могут иметь начальные адреса X100 и X200. Используйте пункт меню DirectSOFT32 PLC/Configure I/O, чтобы вручную указать адрес ввода - вывода. В автоматической конфигурации адреса назначаются группами по 8 точек.

Ручная конфигурация, однако, предполагает, что все модули имеют, по крайней мере, 16 точек, так что Вы можете указывать адреса только кратные восьмеричному числу 20. Вы можете,естественно, использовать и модули с 8-ю точками ввода или вывода, но из 16 адресов будут использованы только старшая часть, а восемь адресов отстанутся неиспользуемыми.

ПРЕДУПРЕЖДЕНИЕ: Если Вы вручную конфигурируете адреса для слота ввода - вывода, то адресация для других модулей может изменяться. Это происходит потому, что процессор DL06 не разрешает использовать двойные адреса ввода/вывода. Вы должны всегда исправлять любые ошибки конфигурации перед запуском процессора в рабочий режим. Неисправленные ошибки могут вызывать непредсказуемые действия, которые могут привести к риску получения травмы персоналом или повреждению оборудования.

Расчет мощности

DL06 имеет четыре дополнительных слота. Чтобы определить ли, имеете ли Вы достаточную мощность для выбираенной комбинации модулей, Вам необходимо будет выполнить расчет потребляемой мощности.

Электропитание

Электропитание осуществляется из двух источников, первый – встроенный источник питания каркаса и, если требуется большая мощность, дополнительный внешний источник питания (приобретается пользователем отдельно). Модели D0-06xx (с питающим напряжением переменного тока) имеют встроенный источник питания на 24В постоянного тока ограниченной мощности. Вывод 24VDC можно использовать для подачи питания на внешние устройства. При расчете требуемой мощности, сначала, рассматривают мощность, потребляемая модулями от блока питания 5В контроллера. Все контроллеры DL06 имеют источник питания 5В постоянного тока одинаковой мощности. Только модели с питанием от сети переменного тока имеют дополнительно встроенный источник питания на 24В постоянного тока.

Общая потребляемая мощность для блоков питания на 5В и 24В остается постоянная. Максимальная мощность выдаваемая блоком питания на 5В зависит от нагрузки на блок питания 24В, и, наоборот, максимальная мощность выдаваемая блоком питания на 24В зависит от нагрузки на блок питания. Определите внутреннюю потребляемую мощность из таблицы на следующей странице.

Мощность, потребляемая основным модулем

Из-за различных конфигураций ввода - вывода, доступных в семействе микроконтроллеров DL06, мощность, потребляемая непосредственно основным модулем изменяется от модели к модели. Вычтите мощность, потребляемой основным модулем из мощности встроенного блока питания. Убедитесь, что Вы вычли обе потребляемые мощности и по 5В, и по 24В.

Мощность, потребляемая дополнительными модулями

Затем, вычтите мощность, потребляемую дополнительными модулями, которые Вы планируете использовать.

Снова, не забудьте вычитать обе потребляемые мощности и по 5В, и по 24В. Если ваш расчет мощности показывает, что источника питания хватит для питания Вашей конфигурации, то у Вас получилась рабочая конфигурация.

Мощность встроенного блока питания каркаса DL06

№ каталога	5В (мА)	24В (мА)
D0-06xx	<1500мА	300мА
	<2000мА	200мА
D0-06xx-D	1500мА	-

Если нагрузка по 5В менее 2000мА, но более 1500мА, то доступная мощность блока питания по 24В – 200мА
Если нагрузка по 5В менее 1500мА, то доступная мощность блока питания по 24В – 300мА

О модулях смотрите в руководстве на дополнительные модули DL05/DL06.

Потребляемая мощность питания каркаса DL06

№ каталога	5В (мА)	24В (мА)
D0-06AA	800мА	-
D0-06AR	900мА	-
D0-06DA	800мА	-
D0-06DD1	600мА	280mA*
D0-06DD2	600мА	-
D0-06DR	950мА	-
D0-06DD1-D	600мА	280mA*
D0-06DD2-D	600мА	-
D0-06DR-D	950мА	-

* Дополнительный источник питания 24В использует клемму V+ для питания выходов D0-06DD1/-D типа потребитель.

Потребляемая мощность других устройств DL06

№ каталога	5В (мА)	24В (мА)
D0-06LCD	50мА	-
D2-HPP	200мА	-
DV1000	150мА	-

Пример расчета баланса мощности

Источник питания		**5В (мА)**	**24В (мА)**
D0-06DD1 (выберите строку А или В)	A	1500мА	300мА
	B	2000мА	200мА
Требуемый ток		**5В (мА)**	**24В (мА)**
D0-06DD1		600мА	280мА*
D0-16ND3		35мА	0
D0-10TD1		150мА	0
D0-08TR		280мА	0
F0-4AD2DA-2		100мА	0
D0-06LCD		50мА	0
Макс. потребляемый ток		1215мА	280мА
Резерв	A	285мА	20мА
	B	785мА	Прим. 1

Примечание 1: Если используется дополнительный источник питания контроллера на 24В для питания выходов типа потребитель используйте расчет мощности, приведенный выше, в строке А.

Настройка портов DL06

Раздел содержит описание настройки встроенных сетевых портов процессора либо с протоколом MODBUS, либо с протоколом DirectNET. Это даст вам возможность подсоединить ПЛК DL06 непосредственно к сети MODBUS через протокол RTU или к другим устройствам в сети DirectNET. Ведущий сети MODBUS должен иметь возможность выдавать команды MODBUS на чтение и запись соответствующих данных. За подробной информацией по DirectNET обратитесь к руководству по DirectNET (DA-DNET_M).

Характеристики портов

Коммуникационный порт 1

Com 1

Для связи с Ручным программатором, DirectSOFT, панелями операторов и др.

- 6-контактов,RS232C
- 9600 бод (фиксированная скорость)
- Контроль нечетности (фиксированно)
- 8 бит данных
- 1 стартовый и 1 стоповый биты
- Асинхронный, полудуплекс, DTE
- Протоколы (Avto-select):
  K-sequence (Slave)
  DirectNET(Slave)
  MODBUS (Slave)

Коммуникационный порт 2

Com 2

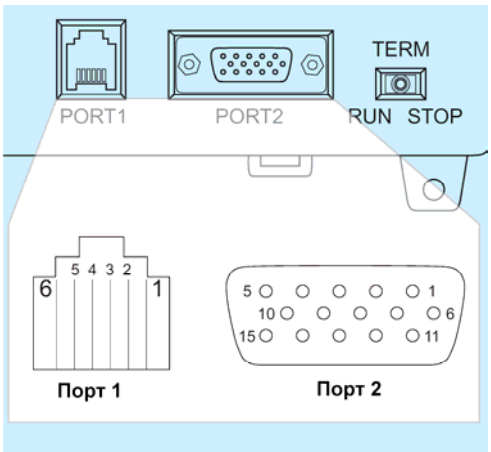
Для связи с Ручным программатором, DirectSOFT, панелями операторов и др.

- 15-контактов, многофункционакльный порт RS232C, RS422, RS485;
- Скорость передачи: 300, 600, 1200, 2400, 4800,9600,19200,38400
- Контроль нечетности (по умолчанию),четности, нет контроля
- 8 бит данных
- 1 стартовый и 1 стоповый биты
- Асинхронный, полудуплекс, DTE
- Протоколы (Avto-select):
  K-sequence (Slave)
  DirectNET(Master/Slave)
  MODBUS (Master/Slave)
  Непроцедурный/Принтер/ASCII ввод/вывод

Описание разъемов

Описание контактов порт 1

1	0V	Питание (-) (земля)
2	5V	Питание (+)
3	RXD	Прием данных (RS232S)
4	TXD	Передача данных (RS232S)
5	5V	Контакт питания (+)
6	0V	Контакт питания (-) (земля)



Описание контактов порт 2

1	5V	Питание (+)
2	TXD	Передача данных (RS-232C)
3	RXD	Прием данных (RS-232C)
4	RTS	Готовность к передаче
5	CTS	Готовность к приему
6	RXD-	Прием данных (-) (RS-422/485)
7	0V	Питание (-) (земля)
8	0V	Питание (-) (земля)
9	TXD+	Передача данных (+) (RS-422/485)
10	TXD-	Передача данных (-) (RS-422/485)
11	RTS+	Готовность к передаче (+) (RS-422/485)
12	RTS-	Готовность к передаче (-) (RS-422/485)
13	RXD+	Прием данных (+) (RS-422/485)
14	CTS+	Готовность к приему (+) (RS-422/485)
15	CTS-	Готовность к приему (-) (RS-422/485)

Выбор спецификации сети

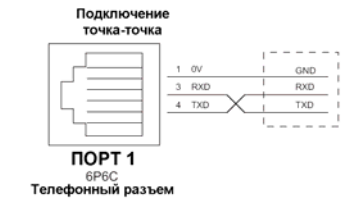
Многофункциональный порт DL06 ПЛК предоставляет Вам возможность использования спецификаций протоколов RS-232C, RS-422, или RS-485. Прежде всего определите тип сети: 2-х проводная RS-232C, 4-х проводная RS-422, или 2-х/4-х проводная RS-485.

Наиболее простой является спецификация протокола RS-232C, которая предназначена для связи только двух устройств и на расстоянии до 15 метров максимум. Сигналы сетей RS-422 и RS-485 обеспечивают более длинные расстояния (до 1000 метров максимум) и предназначены для многоточечных сетей связи (от 2 до 247 устройств).

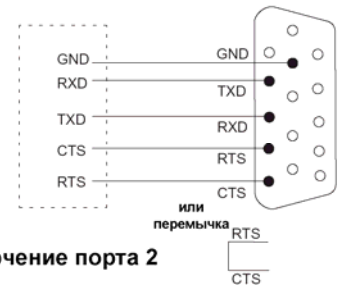
ПРИМЕЧАНИЕ: С двух концов сетей RS-422 и RS-485 требуется устанавливать терминальные (согласующие) резисторы. Необходимо выбрать резисторы, которые соответствуют оценке полного сопротивления кабеля (между 100 и 500 Омами).

Сеть RS-232

Обычно, сигналы RS-232 используются для коротких расстояний (15 метров максимум), и для связи между двумя устройствами.



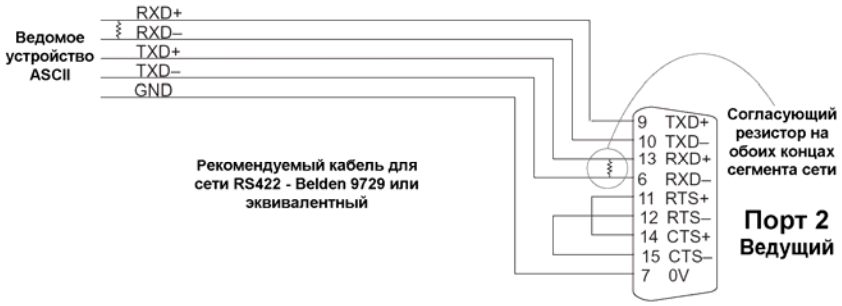
Подключение порта 1



Подключение порта 2

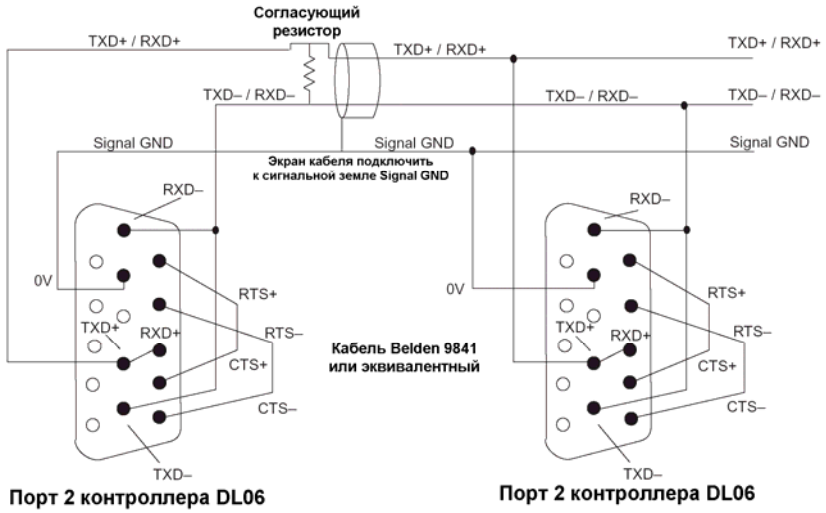
Сеть RS-422

Сигналы RS-422 предназначены для передачи на более длинныерасстояния (1000 метров максимум). Используйте согласующие резисторы в обоих концах сегмента сети RS-422, соответствующие полному сопротивлению кабеля (между 100 и 500 Омами).



Сеть RS-485

Сеть RS-485 предназначена для использования на длинных расстояний (1000 метров максимум) и при многоточечной схемы подключения. Используйте согласующие резисторы в обоих концах сегмента сети RS-485, соответствующие полному сопротивлению кабеля (между 100 и 500 Омами).

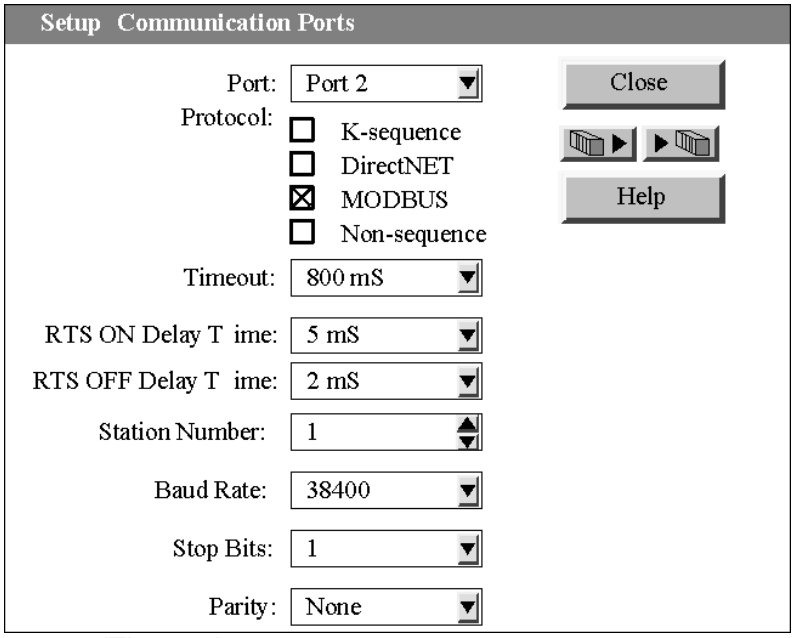


Подключения к сетям MODBUS и DirectNET

Настройка порта 2 на MODBUS

В DirectSOFT выберите меню PLC, далее Setup, затем “Secondary Comm Port”

- Порт (Port). Из списка номеров портов в верхней части окна выберите «Port2».
- Протокол (Protocol): Установите метку «MODBUS» в левой части окна (на Ручном Программаторе используйте AUX 56 и выберите «MBUS») и вы увидите следующее окно.



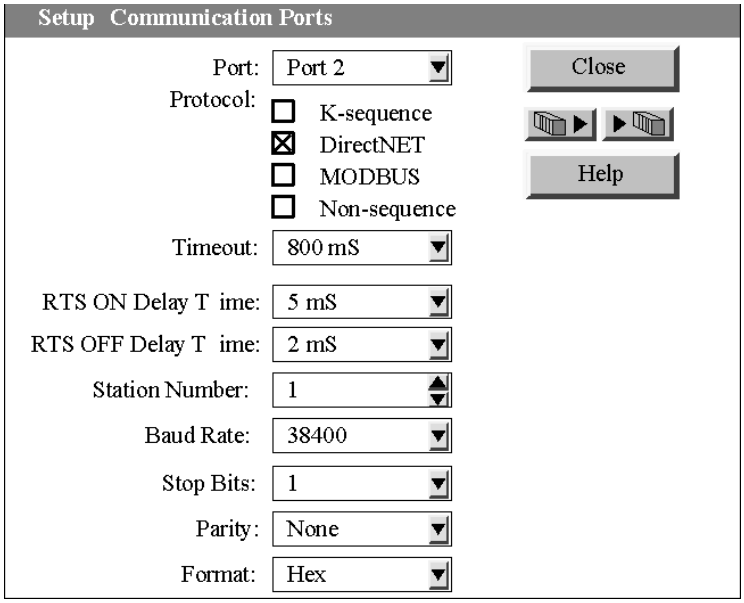
- Время ожидания (Timeout). время ожидания порта после посылки им сообщения и до получения реакции перед регистрацией ошибки.
- Время задержки RTS (Ready to Send) ON/OFF. Время задержки RTS ON — это время между установлением сигнала на линии RTS и передачей данных. Время задержки RTS OFF — это время после окончания передачи данных DL06 и снятием сигнала на линии RTS. При использовании DL06 в многоточечной сети Время задержки RTS ON должно быть не менее 5мс, Время задержки RTS OFF должно быть не менее 2мс. А если у Вас проблемы с синхронизацией, то эти времена должны быть увеличены.
- Номер станции (Station Number). Для того, чтобы сделать порт Процессора ведущим устройством MODBUS выберите в качестве номера станции «1». Допустимый диапазон номеров станций для ведомых устройством MODBUS — от 1 до 247, но сетевые команды DL06, используемые в режиме Ведущего устройства, допускают для ведомых устройств только номера с 1 по 99. Каждое ведомое устройство должно иметь уникальный номер. При включении питания порт автоматически устанавливается в режим ведомого устройства, который сохраняется до тех пор, пока DL06 не начнет выполнять сетевые команды релейной логики, использующие порт как Ведущее устройство. После их выполнения порт возвращается в режим ведомого устройства, пока релейная логика снова не использует этот порт.
- Скорость передачи (Baud Rate). Доступны скорости передачи 300, 600, 900, 2400, 4800, 9600, 19200 и 38400 бод. Сначала выберите наибольшую скорость передачи, и снижайте скорость, если при передаче возникают ошибки в данных или помехи в линии. Вы должны установить одинаковую скорость передачи данных для всех устройств сети.
- Стоповые биты (Stop Bits). Для использования в протоколе выберите 1 или 2 стоповых бита.
- Контроль четности (Parity). Для контроля ошибок выберите «нет», «четное» или «нечетное».

Затем нажмите клавишу, указывающую на запоминание конфигурации Процессором DL06, и потом нажмите «Close» - «Закрыть».

Настройка порта 2 на DirectNET

В DirectSOFT выберите меню PLC, далее Setup, затем Secondary Comm Port…

- Порт (Port). Из списка номеров портов в верхней части окна выберите «Port2».
- Протокол (Protocol). Установите 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 RTS (Ready to Send) ON/OFF. Время задержки RTS ON — это время между установлением сигнала на линии RTS DL06 и передачей данных. Время задержки RTS OFF — это время после окончания передачи данных DL06 и снятием сигнала на линии RTS. При использовании DL06 в многоточечной сети Время задержки RTS ON должно быть не менее 5мс, Время задержки RTS OFF должно быть не менее 2мс. А если у Вас проблемы с синхронизацией, то эти времена должны быть увеличены.
- 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 DL06 не начнет выполнять сетевые команды релейной логики, использующие порт как Ведущее устройство. После выполнения этих команд 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, и снижайте скоростям, если при передаче возникают ошибки в данных или помехи в линии. 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 (HEX), либо ASCII формат.

Затем нажмите клавишу, указывающую на запоминание конфигурации процессором DL06, и потом нажмите «Close».

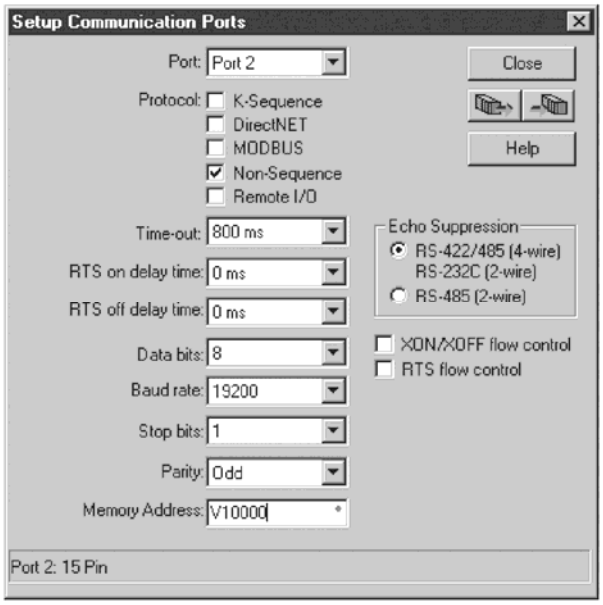
Протокол Non–Sequence (ASCII ввод/вывод и Печать)

Настройка порта на работу с протоколом Non-Sequence

Настройка порта 2 на работу с протоколом Non-Sequence разрешает процессору использовать порт 2, чтобы читать или записывать необработанные строки ASCII-символов, используя команды ASCII. Смотрите команды ASCII ввода/вывода и команды печати (PRINT) в главе 5.

В DirectSOFT выберите меню PLC, далее Setup, затем “Secondary Comm Port”

- Порт (Port). Из списка номеров портов в верхней части окна выберите «Port2».
- Протокол (Protocol): Установите на метку «Non-sequence» в левой части окна.



- Время ожидания (Timeout). время ожидания порта после посылки им сообщения и до получения реакции перед регистрацией ошибки.
- Время задержки RTS (Ready to Send) ON/OFF. Время задержки RTS ON — это время между установлением сигнала RTS и передачей данных. Время задержки RTS OFF — это время после окончания передачи данных DL06 и снятием сигнала RTS .
- Биты данных (Data Bits): Выберите 7 или 8 бит соответствующее числу информационных бит, для подключенного устройства.
- Скорость передачи (Baud Rate). Доступны скорости передачи 300, 600, 900, 2400, 4800, 9600, 19200 и 38400 бод. Сначала выберите наибольшую скорость передачи, и снижайте скоростям, если при передаче возникают ошибки в данных или помехи в линии. Вы должны установить одинаковую скорость передачи данных для всех устройств сети. Обратитесь к соответствующему руководству на подключаемое изделие за дополнительной информацией.
- Стоповые биты (Stop Bits). Выберите 1 или 2 стоповых бита так, чтобы это число соответствовало числу стоповых бит подключаемого устройства.
- Контроль четности (Parity). Для контроля ошибок выберите «нет», «четное» или «нечетное» так, чтобы это значение соответствовало настройке подключаемого устройства.
- Подавления эхо (Echo Suppression): Выберите кнопку, соответствующую Вашему виду подключения порта 2.
- Аппаратное управление потоком данных (Xon/Xoff Flow Control): Выберите этот пункт, если Вы имеете подключаемые к Порту 2 устройства с сигналами RTS и CTS.
- Управление потоком данных через RTS (RTS Flow Control): Выберите этот пункт, если Вы имеете подключаемые к Порту 2 устройства с сигналом RTS.

Затем нажмите клавишу, указывающую на запоминание конфигурации Процессором DL06, и потом нажмите «Close».

Функционирование сети в режиме ведомого устройства

В данном разделе рассматривается вопрос, каким образом другие устройства в сети могут быть подключаться к порту процессора, который сконфигурирован Вами как ведомое устройство (DL06) DirectNET Slave или MODBUS Slave. Ведущее устройство MODBUS должно использовать протокол MODBUS RTU для установления соединения с DL06 в качестве ведомого устройства. Программа этого Ведущего устройства должны посылать код функции MODBUS и адрес MODBUS для указания ячейки памяти ПЛК, входящей в состав DL06. Ведущее устройство DirectNET использует обычные адреса для доступа к процессору DL06 и к системе. Не требуется никаких команд релейной логики процессора для поддержания функционирования MODBUS или DirectNET в качестве ведомого устройства.

Поддерживаемые коды функций MODBUS.

Коды функций MODBUS определяют тип запроса: чтение или запись, а также относится ли запрос к одному данному, либо к группе данных. Ниже приводятся коды функций MODBUS, поддерживаемые DL06.

Коды функций MODBUS	Функция	Доступные типы данных DL06
01	Чтение группы обмоток	Y, CR, T, CT
02	Чтение группы входов	X, SP
05	Установка/сброс одного реле	Y, CR, T, CT
15	Установка/сброс группы реле	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

Определение адресов MODBUS

Существуют два способа, с помощью которых большинство программных средств ведущего устройства может специфицировать ячейку памяти ПЛК. Ими являются:

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

Если программное обеспечение требует тип данных и адрес

Многие программные пакеты ведущих устройств позволяют указать тип данных MODBUS и адрес MODBUS, которые соответствуют ячейке памяти ПЛК. Это наиболее простой способ, но не все пакеты имеют такую возможность.

Уравнение, применяемое для вычисления адреса, учитывает тип данных, 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 DL06	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	256	X0-X377	2048-2303	Входы
Специальные реле (SP)	512	SP0-SP777	3072-3583	Входы
Выходы (Y)	256	Y0-Y377	2048-2303	Реле
Управляющие реле (CR)	512	C0-C777	3072-4583	Реле
Контакты таймера (T)	128	T0-T177	6144-6271	Реле
Контакты счетчика (CT)	128	CT0-CT177	6400-6527	Реле
Биты состояния Stage (S)	256	S0-S377	5120-5375	Реле
Для типов данных «Слово» …. Преобразовать адрес ПЛК в десятичный + Тип данных				
Текущие значения таймера (V)	128	V0-V177	0-127	Регистр входов (Input Register)
Текущие значения счетчика (V)	128	V1000-V1177	512-639	Регистр входов (Input Register)
V-память, данные пользователя (V)	3968	V1200-V7377	640-3839	Регистр хранения (Holding Register)
V-память, неразрушаемая (V)	128	V7600-V7777	3968-4095	Регистр хранения (Holding Register)

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

Пример 1: V2100

Найти адрес MODBUS для пользовательской ячейки памяти V2100

1. Найдите ячейку V-памяти в таблице.
2. Преобразуйте V2100 в десятичный вид (1088) — Регистр хранения 1088
3. Используйте тип данных MODBUS из таблицы.

V Память, данные пользователя (V)	3200	V1200 – V7377	640 – 3839	Регистр хранения (Holding)

Пример 2: Y20

Найти адрес MODBUS для выхода Y20

1. Найдите выхода Y в таблице.
2. Преобразуйте Y20 в десятичный вид (16) — Реле 2064
3. Прибавьте стартовый адрес для диапазона (2048)
4. Используйте тип данных MODBUS из таблицы.

Выходы (Y)	256	Y0– Y377	2048 - 2303	Реле

Пример 3: текущее значение T10

Найти адрес MODBUS для получения текущего значения таймера T10

1. Найдите таймера в таблице.
2. Преобразуйте T10 в десятичный вид (8) — Регистр входа 8
3. Используйте тип данных MODBUS из таблицы.

Текущие значения таймера (V)	128	V0 – V177	0 – 127	Регистр входа (Input)

Пример 4: C54

Найти адрес MODBUS для управляющего реле C54

1. Найдите управляющие реле в таблице.
2. Преобразуйте C54 в десятичный вид (44) — Реле 3116
3. Прибавьте стартовый адрес для диапазона (3072)
4. Используйте тип данных MODBUS из таблицы.

Управляющее реле (CR)	512	C0– C77	3072 - 3583	Реле

Если программное обеспечение для MODBUS требует только адрес

Некоторые программы не позволяют указывать тип данных и адрес MODBUS. Они могут указывать только адрес. Эта ситуация требует других шагов при определении адреса, но они также достаточно просты. По существу MODBUS тоже разделяет типы данных по диапазонам адресов. Отсюда следует, что адреса достаточно для выбора типа данных и ячейки 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 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.

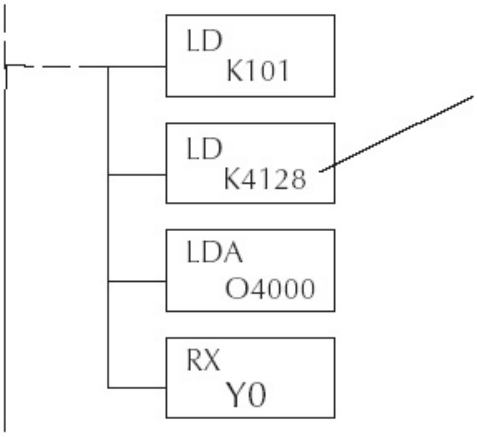
- Дискретные — X, SP, Y, CR, S, T, C (контакты)
- Слова — V, Текущее значение таймера, Текущее значение счетчика

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.

Дискретный тип данных				
Тип памяти DL06	**Диапазон ПЛК (восьмерич)**	**Адрес режима 484**	**Адрес режима 584/984**	**Тип данных MODBUS**
Глобальные входы (GX)	GX0-GX1746	1001 - 1999	10001 - 10999	Входы
	GX1747-GX3777	-	11000 - 12048	Входы
Входы (X)	X0-X1777	-	12049 - 13072	Входы
Специальные Реле (SP)	SP0-SP777	-	13073 - 13584	Входы
Глобальные выходы (GY)	GY0 – GY3777	1 - 2048	1 - 2048	Выходы
Выходы (Y)	Y0-Y1777	2049 - 3072	2049 - 3072	Выходы
Управляющие Реле (CR)	C0-C3777	3073-5120	3073-5120	Выходы
Контакты Таймера (T)	T0-T377	6145 - 6400	6145 - 6400	Выходы
Контакты Счетчика (CT)	CT0-CT377	6401 - 6656	6401 - 6656	Выходы
Биты состояния Стадий (S)	S0-S1777	5121 - 6144	5121 - 6144	Выходы

Тип данных «Слово»			
Регистры	**Диапазон ПЛК (восьмерич)**	**Вход/Хранение (режим 484)***	**Вход/Хранение (режим 584/984)***
V память (Таймеры)	V0-V377	3001/4001	30001/40001
V память (Счетчики)	V1000-V1177	3513/4513	30513/40513
V-память (данные «Слово»)	V1200-V1377	3641/4641	30641/40641
	V1400-V1746	3769/4769	30769/40769
	V1747-V1777	-	31000/41000
	V2000-V7377	-	41025
	V10000-V17777	-	44097

* MODBUS: Функция 04
Контроллеры DL05/06, DL250-1/260, DL350 и DL450 поддерживают функцию 04, регистр чтения входов (Адрес 30001). Для использования функции 04, поместите число 4 на место наиболее значащего разряда (4xxx). Чтобы работать должным образом с этим режимом, все четыре цифры должны быть введены.



Максимальное значение константы – 4128. Это связано с ограничением команд RX/WX (максимум 128 байт). Команда RX использует функцию 04 (30001 диапазон), если число 4 в наиболее значащем разряде слова.

1. О размере карты памяти смотрите руководство пользователя на ваш контроллер. Некоторые из адресов, показанных выше, могут не принадлежать вашему процессору.

Пример 1: V2100 Режим 584/984

Найти адрес MODBUS для пользовательской ячейки памяти V2100. Адрес ПЛК (Дес.)+Адрес режима

1. Найдите V-память в таблице. V2100 = 1088 десятичная
2. Преобразуйте V2100 в десятичный вид (1088) 1088 + 40001 = **41089**
3. Добавьте стартовый адрес для режима(40001).

Для типов данных «Слово». . . .	**Адрес ПЛК (десят.)**	**+**	**Соответствующий адрес режима**			
Текущее значение таймера (V)	128	V0 – V177	0 - 127	3001	30001	Регистр входа
Текущее значение счетчика (V)	128	V1200 – V7377	512 - 639	3001	30001	Регистр входа
V память, пользовательские данные (V)	1024	V2000 – V3777	1024 - 2047	4001	40001	Регистр хранения

Пример 2: Y20 Режим 584/984

Найти адрес MODBUS для выхода Y20 Адрес ПЛК (Дес.)+Стартовый адрес + Режим

1. Найдите выходы Y в таблице. Y20 = 16 десятичная
2. Преобразуйте Y20 в десятичный вид (16) 16 + 2048 + 1 = **2065**
3. Добавьте стартовый адрес для данного диапазона (2048)
4. Добавьте адрес MODBUS для режима (1).

Выходы (Y)	320	Y0-Y477	2048-2367	1	1	Реле
Управляющие реле (CR)	256	C0 – C377	3072 - 3551	1	1	Реле
Контакты таймера (T)	128	T0 – T177	6144 - 6271	1	1	Реле

Пример 3: T10 Режим 484

Найти адрес MODBUS для текущего значения таймера T10 Адрес ПЛК (Дес.)+Адрес режима

1. Найдите значения таймера в таблице. TA10 = 8 десятичная
2. Преобразуйте T10 в десятичный вид (8) 8 + 3001 = **3009**
3. Добавьте стартовый адрес MODBUS для режима (3001).

Для типов данных «Слово». . . .	**Адрес ПЛК (десят.)**	**+**	**Соответствующий адрес режима**			
Текущее значение таймера (V)	128	V0 – V177	0 - 127	3001	30001	Регистр входа
Текущее значение счетчика (V)	128	V1200 – V7377	512 - 639	3001	30001	Регистр входа
V память, пользовательские данные (V)	1024	V2000 – V3777	1024 - 2047	4001	40001	Регистр хранения

Пример 4: C54 Режим 584/984

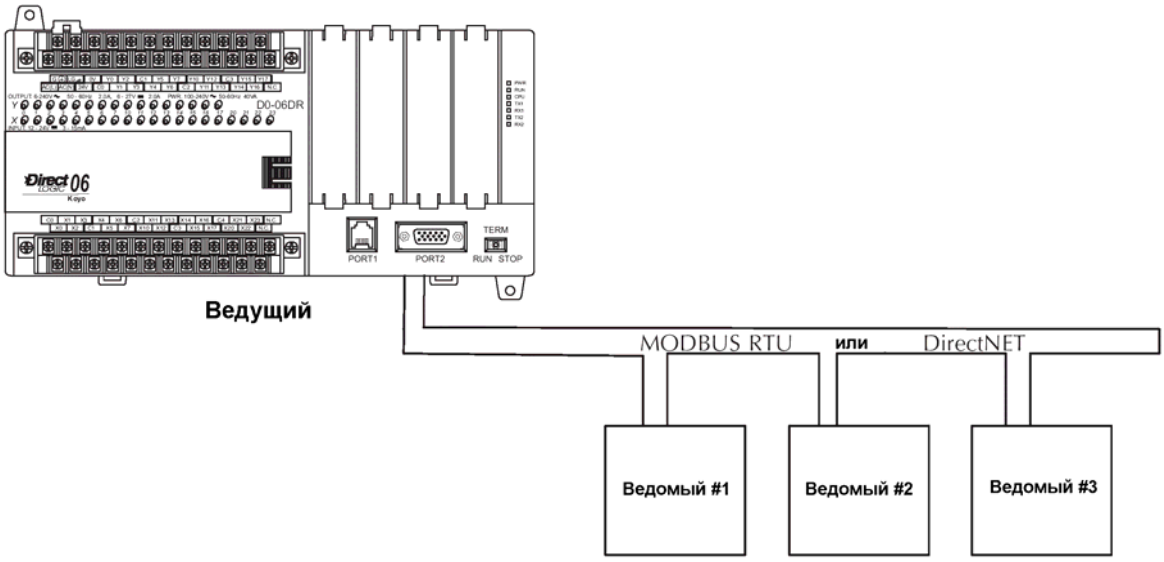
Найти адрес MODBUS для управляющего реле **C54** Адрес ПЛК (Дес.)+Стартовый адрес + Режим

1. Найдите управляющие реле в таблице. C54 = 44 десятичная
2. Преобразуйте C54 в десятичный вид (44) 44 + 3072 + 1 = **3117**
3. Добавьте стартовый адрес для данного диапазона (3072)
4. Добавьте адрес MODBUS для режима (1).

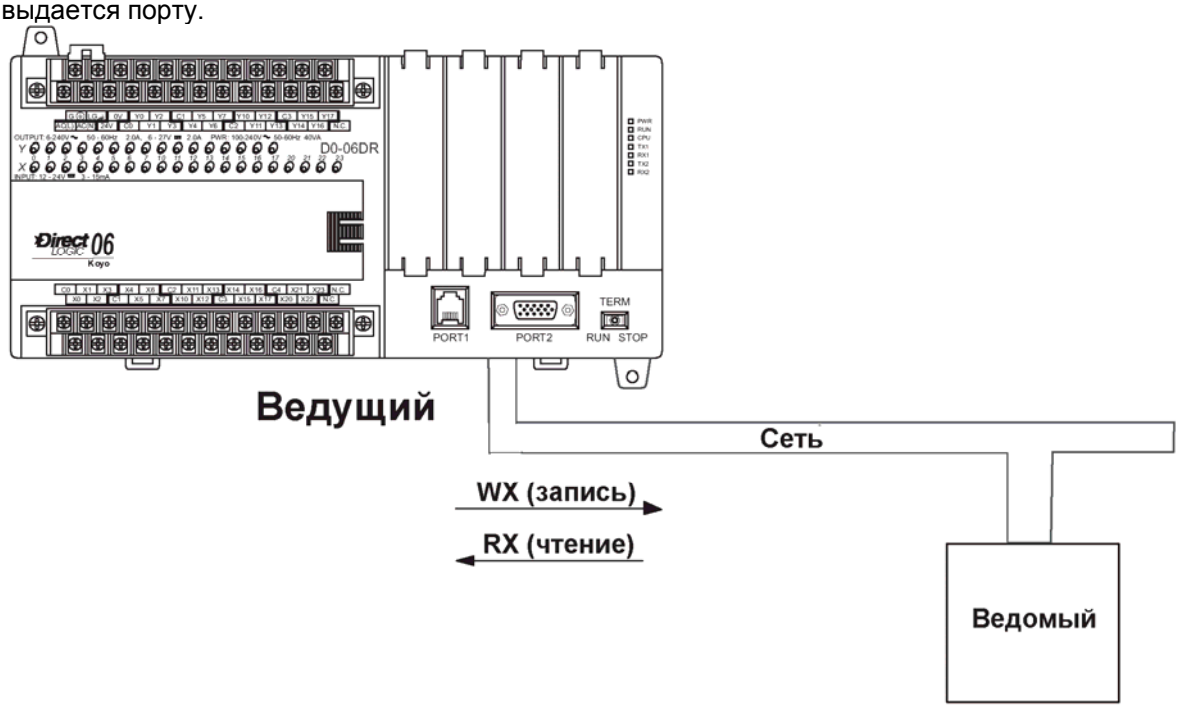
Выходы (Y)	320	Y0-Y477	2048-2367	1	1	Реле
Управляющие реле (CR)	256	C0 – C377	3072 - 3551	1	1	Реле
Контакты таймера (T)	128	T0 – T177	6144 - 6271	1	1	Реле

Функционирование в режиме ведущего устройства сети

В разделе описывается, как DL06 может взаимодействовать с сетью MODBUS или DirectNET в качестве ведущего устройства. Для сетей MODBUS используется протокол MODBUS RTU, который должен одинаково 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 показано, как можно использовать команды релейной логики при работе контроллера в качестве ведущего устройства сети.



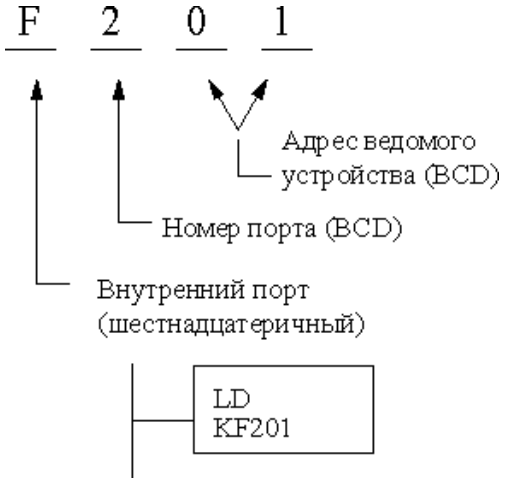
При использовании контроллера DL06 в качестве ведущей станции применяйте команды RLL для инициализации запросов. Команда WX инициирует сетевые операции записи, а команда RX инициирует сетевые операции чтения. Перед выполнением этих команд вам необходимо загрузить данные, относящиеся к операции чтения 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.



Следующая пошаговая процедура содержит информацию, необходимую вам для настройки вашей программы релейной логики на получение данных из ведомого устройства сети.`

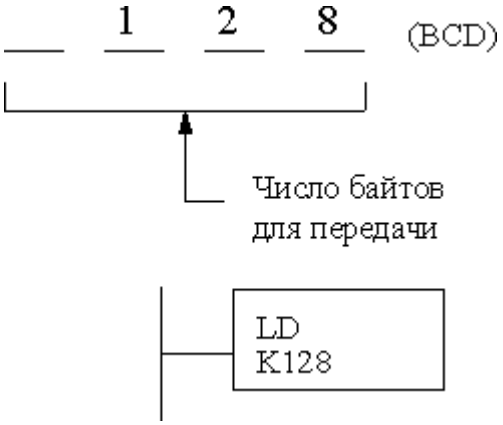
Шаг 1: указать номер порта ведущего устройства и номер ведомого устройства

Первая команда загрузки (LD) определяет номер коммуникационного порта ведущего устройства сети (DL06) и адрес ведомой станции. Эта команда может адресовать до 99 ведомых устройств MODBUS или до 90 ведомых устройств DirectNET. Справа показан формат слова. «F2» в верхнем байте указывает на использования правого порта процессора DL06 с номером 2. Младший байт содержит номер адреса ведомого устройства в BCD (от 01 до 99).



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 должно быть в диапазоне от 1 до 128 в формате BCD.
Число задаваемых байт зависит от типа данных, который вы хотите получить. Например, входные точки DL06 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 DirectLOGIC



Память DL 05 / 06 / 205/350/405	Бит в блоке	Байт
V-память	16	2
Текущее значение T / C	16	2
Входы (X, SP)	8	1
Выходы (биты Y, C, Stage, T/C)	8	1
Временная системная память (Scratch Pad)	8	1
Диагностическое состояние	8	1

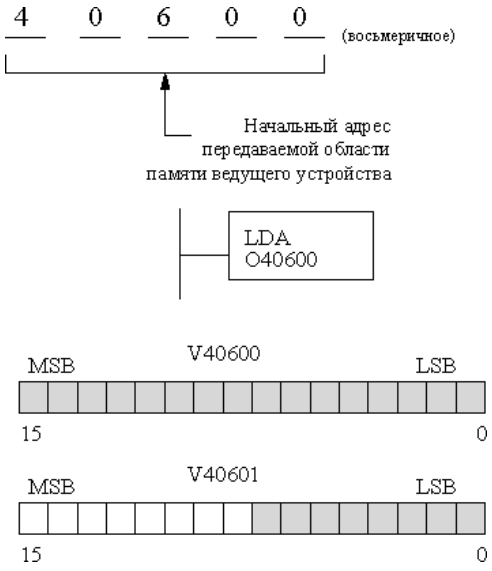
Память DL330/340	Бит в блоке	Байт
Регистры данных	8	1
Накапливающий сумматор T / C	16	2
Биты Ввода/Вывода, внутренних реле, регистр со сдвигами, биты T/C биты Stage	1	1
Временная системная память (Scratch Pad)	8	1
Диагностическое Состояние (5 слов R/W)	16	10

Шаг 3: указать область памяти ведущего устройства

Третьей командой в последовательности RX или WX является команда загрузки адреса (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 DL06 посылает предварительно определенное число байтов из области памяти, начиная с определенного LDA адреса

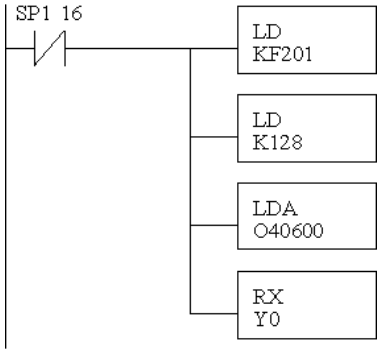
По команде RX процессор DL06 считывает предварительно определенное число байтов из ведомого устройства, помещает полученные данные в область памяти, начиная с определенного LDA адреса.



ПРИМЕЧАНИЕ. Поскольку слова V-памяти всегда имеют 16 бит, 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 младшие 8 бит ячейки последнего слова будут нести информацию. Старшие 8 бит не имеют значения.

Шаг 4: указать 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.



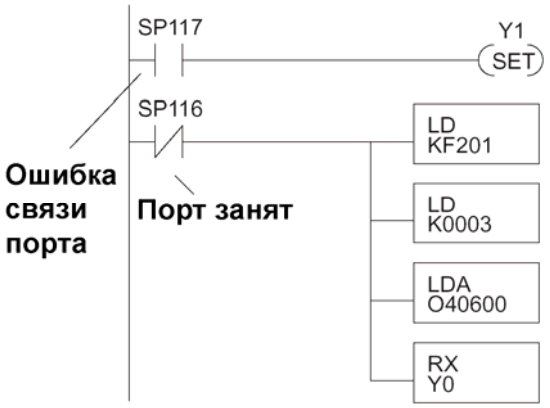
- Ведомые устройства DirectNET — в командах WX и RX указывают те же адреса, что и собственные адреса ввода/вывода .
- Ведомые устройства MODBUS DL405, DL205 или DL06 — в командах WX и RX указывают те же адреса, что и собственные адреса ввода/вывода.
- Ведомые устройства MODBUS DL305 — используют следующую таблицу для преобразования адресов DL305 в адреса MODBUS.

Перекрестные ссылки адресов: память процессора серии DL305 (Кроме 350) — 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 стадий (только для D3-330P)	S0	GY200			

Передача данных из программы релейной логики

Обычно передача данных по сети продолжается дольше, чем одно сканирование. Перед началом очередной транзакции (операции) программа должна ожидать, пока закончится предыдущая передача данных

Порт 2, который может стать ведущим устройством, имеет два контакта связанных с ним Специальных Реле (См. Приложение D по специальным реле коммуникационных портов). Один из них указывает «Порт занят» (SP116), другой указывает «Ошибка связи порта» (SP117). Приведенный выше пример показывает использование этих контактов для сетевого ведущего устройства, который только считывает с устройства (RX). Бит «Порт занят» находится в состоянии «включен», когда ПЛК осуществляет обмен с ведомым устройством. Когда этот бит «выключен», программа может инициировать следующий сетевой запрос.

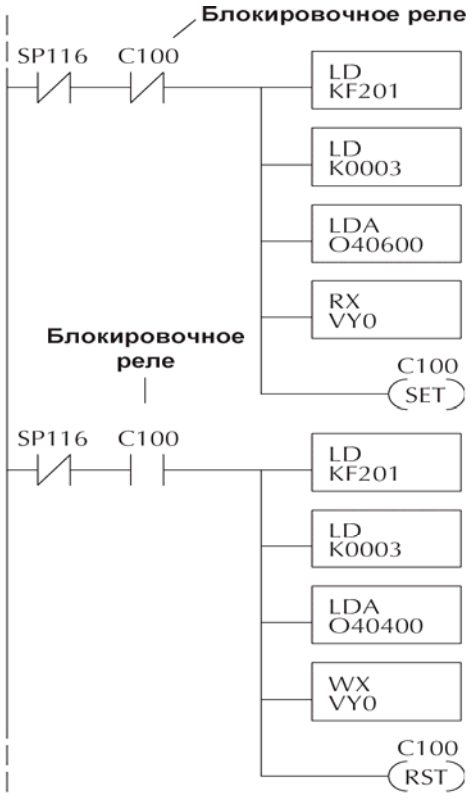


Блокировки многократного чтения и записи

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.

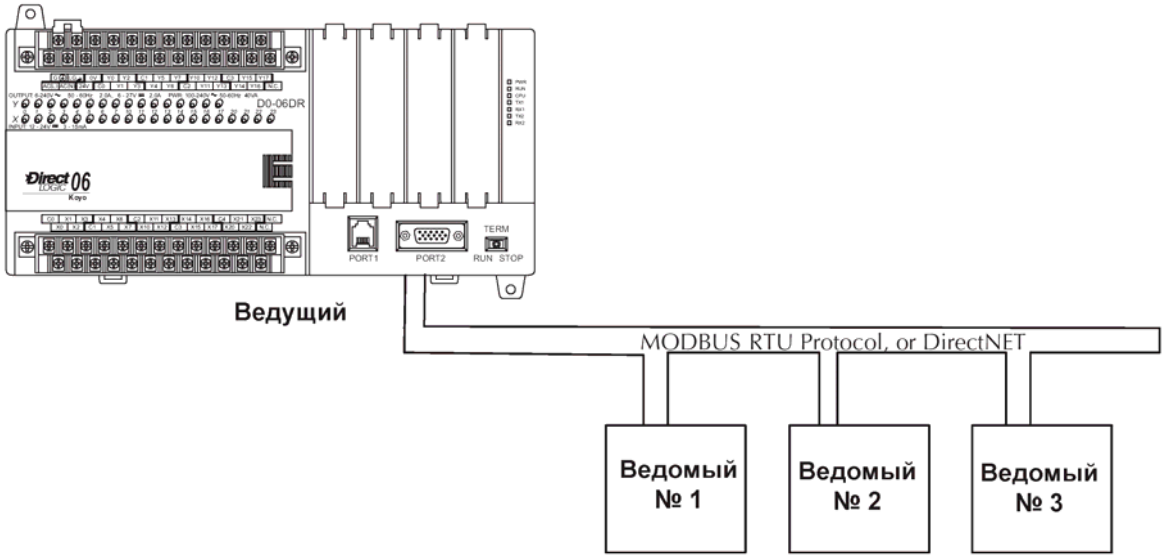
На примере справа после выполнения команды RX устанавливается C100. Когда порт завершает коммуникационную задачу, выполняется вторая подпрограмма и C100 сбрасывается.

Если вы используете Стадийное программирование RLL^{plus}, то вы можете включить каждую подпрограмму в отдельную стадию программы для ее корректного выполнения и переходить с одной стадии на другую, давая возможность только одной из подпрограмм быть активной в каждый момент времени.



Функционирование в режиме ведущего устройства сети MODBUS (команды MRX и MWX)

Этот раздел описывает, как DL06 может связываться по сети MODBUS в качестве ведущего устройства, примененяя команды чтения/записи: MRX и MWX. Эти команды дают вам возможность использовать стандартную адресацию MODBUS в программе релейной логики без необходимости производить преобразование восмеричного в десятичное число. В сети MODBUS один ведущий / множество ведомых устройств. Ведущий - единственный участник сети, который может инициировать запросы в сети. Этот раздел учит, как разработать программу релейной логики для функционирования DL06 в режиме ведущего устройства MODBUS.



Поддерживаемые функциональные коды MODBUS

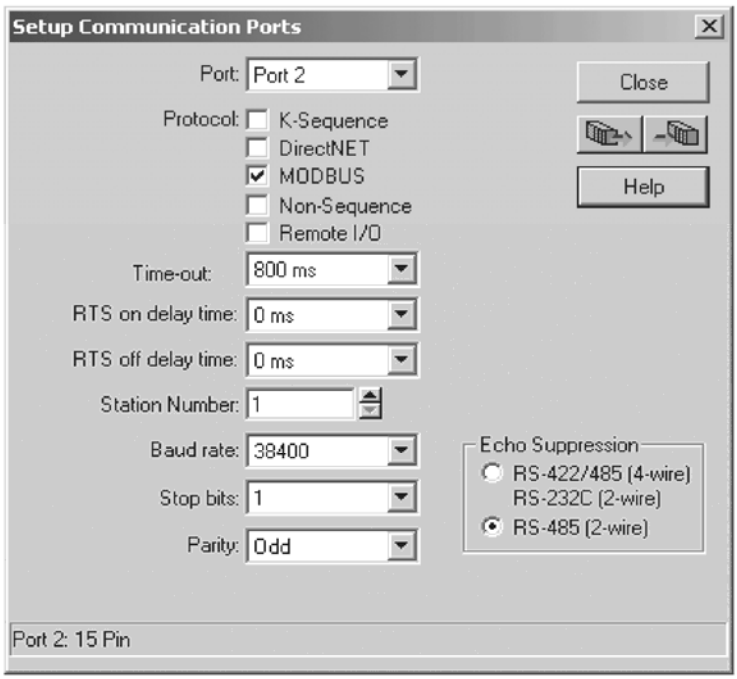
Коды функций MODBUS определяют тип запроса: чтение или запись, а также относится ли запрос к одному данному, либо к группе данных. DL06 поддерживает коды команд MODBUS, описанные ниже.

код MODBUS	Назначение	Доступный тип данных DL06
01	Считывание группы реле	Y,CR,T,CT
02	Считывание группы входов	X,SP
05	Установка/сброс одиночного реле (только для одного ведомого)	Y,CR,T,CT
15	Установка/сброс группы реле	Y,CR,T,CT
03,04	Считывание значения из одного или из большего количества регистров	V
06	Запись значения в один регистр (только для ведомого)	V
07	Считывание регистров состояния(Exeption status)	V
08	Диагностика	V
16	Запись значения в группу регистров	V

Настройка порта MODBUS

Выберите в меню программы ***Direct***SOFT32 пункт PLC, затем Setup, затем “Secondary Comm Port”.

- Порт (Port). Из списка номеров портов в верхней части окна выберите «Port2».
- Протокол (Protocol): Поставьте метку слева от надписи «MODBUS» (используйте функцию AUX 56 на ручном программаторе и выберите “MBUS”), и появится следующий экран.

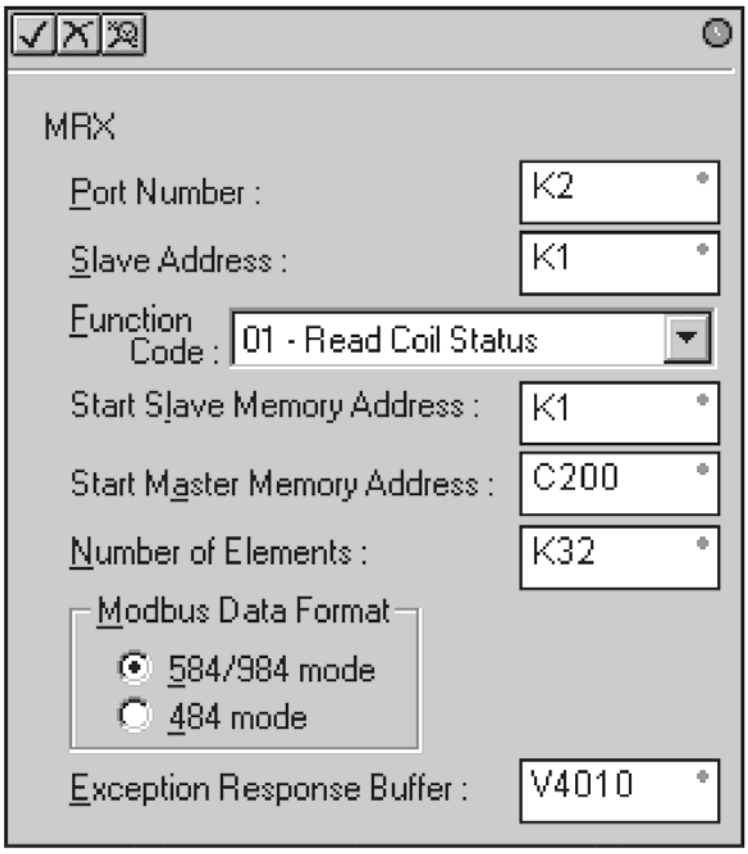


- Время ожидания (Timeout). время ожидания порта после посылки им сообщения и до получения реакции перед регистрацией ошибки.
- Время задержки RTS (Ready to Send) ON/OFF. Время задержки RTS ON — это время между установлением сигнала RTS и передачей данных. Время задержки RTS OFF — это время после окончания передачи данных DL06 и снятием сигнала RTS.
- Номер станции (Station Number): Для порта MODBUS, работающего в режиме ведущего необходимо выбрать “1”. Диапазон адресов MODBUS для ведомых станций от 1 до 247. Каждый ведомый в сети должен иметь уникальный номер. При включении питания, порт автоматически устанавливается в режим ведомого, до тех пор пока не начнут выполнятся команды MWX/MRX, которые используют порт в режиме ведущего. После этого, порт возвращается в режим ведомого до тех пор пока программа релейной логики не использует порт снова.
- Скорость передачи (Baud Rate). Доступны скорости передачи 300, 600, 900, 2400, 4800, 9600, 19200 и 38400 бод. Сначала выберите наибольшую скорость передачи, и снижайте скоростям, если при передаче возникают ошибки в данных или помехи в линии. Вы должны установить одинаковую скорость передачи данных для всех устройств сети.
- Стоповые биты (Stop Bits). Выберите 1 или 2 стоповых бита, используемых в протоколе.
- Контроль четности (Parity). Для контроля ошибок выберите «нет», «четное» или «нечетное».
- Подавления эхо (Echo Suppression): Выберите кнопку, соответствующую Вашему виду подключения порта 2.

Затем нажмите клавишу, указывающую на запоминание конфигурации Процессором DL06, и потом нажмите «Close» - «Закрыть».

Чтение по сети MODBUS (MRX)

Команда чтения по сети MODBUS (MRX) используется DL06 в режиме ведущего устройства сети для чтения блока данных из подключенного ведомого устройства и записи данных в свои адреса V-памяти. Команда допускает выбор пользователем функционального кода MODBUS, адреса ведомого устройства, начальные адреса в ведущем и ведомом устрйстве, число передаваемых элементов, формат данных MODBUS и Буфер ошибок выполнения(Exception Response Buffer).



- Номер порта (Port Number): должен быть Port 2 (K2)
- Адрес ведомого (Slave Address): Определите адрес ведомой станции (0–247)
- Функциональный код (Function Code): Следующие функциональные коды MODBUS поддерживаются командой MRX:
  - 01 – Чтение группы реле
  - 02 – Чтение группы входов
  - 03 – Чтение регистров хранения
  - 04 – Чтение входных регистров
  - 07 – Чтение регистров состояния (Exeption status)
  - 08 – Диагностика
- Начальный адрес памяти ведомого (Start Slave Memory Address): Определите начальный адрес памяти ведомого устройства откуда должны быть считаны данные. Смотри таблицу на следующей странице.
- Начальный адрес памяти ведущего (Start Master Memory Address): Определите начальный адрес памяти ведомого устройства куда должны быть записаны данные. Смотри таблицу на следующей странице.
- Число элементов (Number of Elements): Укажите количество реле, входов, регистров, которое необходимо прочитать. Можно использовать константу или ячейку памяти. Смотри таблицу на следующей странице.
- Формат данных (MODBUS Data Format): Определите формат данных, который будет использоваться MODBUS 584/984 или 484.
- Буфер ошибок выпонения (Exception Response Buffer): Определите адрес памяти ведущего устройства, куда будет помещены ошибки выполнения.

Адреса памяти ведомого устройства для команды MRX

Диапазоны адресов ведомого устройства для команды MRX		
Функциональный код	**Формат данных MODBUS**	**Диапазоны адресов ведомого**
01 – Чтение реле	484	1–999
01 – Чтение реле	584/984	1–65535
02 – Чтение состояния входов	484	1001–1999
02 – Чтение состояния входов	584/984	10001–19999 (5 digit) или 100001–165535 (6 digit)
03 – Чтение регистров хранения	484	4001–4999
03 – Чтение регистров хранения	584/984	40001–49999 (5 digit) или 4000001–465535(6 digit)
04 – Чтение входных регистров	484	3001–3999
04 – Чтение входных регистров	584/984	30001–39999 (5 digit) или 3000001–365535 (6 digit)
07 – Чтение регистров состояния	484 и 584/984	Нет доступа
08 – Диагностика	484 и 584/984	0–65535

Адреса памяти ведущего устройства команды MRX

Диапазоны адресов ведущего устройства для команды MRX	
Тип данных операнда	**Диапазон DL06**
Входы X	0–777
Выходы Y	0–777
Управляющие реле C	0–1777
Биты стадий S	0–1777
Биты таймеров T	0–377
Биты счетчиков CT	0–177
Специальные реле SP	0–777
V–память V	Вся
Удаленные входы GX	0–3777
Удаленные выходы GY	0–3777

Задание числа элементов в команде MRX

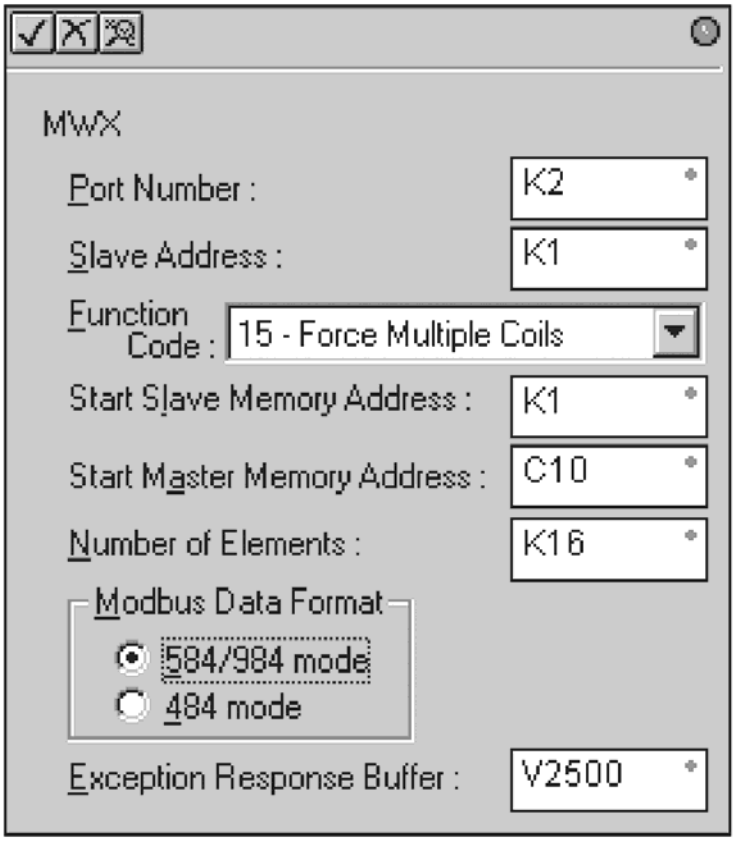
Число элементов	
Тип данных операнда	**Диапазон DL06**
V–память V	Вся
Константы K	1–2000

Буфер ошибок выполнения команды MRX

Буфер ошибок выполнения	
Тип данных операнда	**Диапазон DL06**
V–память V	Вся

Запись в сети MODBUS (MWX)

Команда записи в сети MODBUS (MWX) используется DL06 в режиме ведущего устройства сети для записи блока данных из ведушего в память подключенного ведомого устройства. Команда допускает выбор пользователем функционального кода MODBUS, адреса ведомого устройства, начальных адресов в ведущем и ведомом устройстве, числа передаваемых элементов, формата данных MODBUS и места буфера ошибок выполнения.



- Номер порта (Port Number): должен быть Port 2 (K2)
- Адрес ведомого (Slave Address): Определите адрес ведомой станции (0–247)
- Функциональный код (Function Code): Следующие функциональные коды MODBUS поддерживаются командой MWX:
  - 05 – Установка одного реле (Force Single Coil)
  - 06 – Установка одного регистра (Preset Single Register)
  - 08 – Диагностика
  - 15 – Установка нескольких реле (Force Multiple Coil)
  - 16 – Установка нескольких регистров (Preset Multiple Register)
- Начальный адрес памяти ведомого (Start Slave Memory Address): Определите начальный адрес памяти ведомого устройства куда должны быть записаны данные.
- Начальный адрес памяти ведущего (Start Master Memory Address): Определите начальный адрес памяти ведомого устройства откуда должны быть записаны данные.
- Число элементов (Number of Elements): Определите количество реле или регистров, которое необходимо записать. Это поле активно только при выборе функциональных кодов 15 или 16.
- Формат данных (MODBUS Data Format): Определите формат данных, который будет использоваться MODBUS 584/984 или 484.
- Буфер ошибок выпонения (Exception Response Buffer): Определите адрес памяти ведущего устройства, куда будет помещены ошибки выполнения.

Адреса памяти ведомого устройства для команды MWX

Диапазоны адресов ведомого устройства для команды MWX		
Функциональный код	**Формат данных MODBUS**	**Диапазоны адресов ведомого**
05 – Установка одного реле	484	1–999
05 – Установка одного реле	584/984	1–65535
06 – Установка одного регистра	484	4001–4999
06 – Установка одного регистра	584/984	40001–49999 (5 digit) или 400001–465535 (6 digit)
08 – Диагностика	484 и 584/984	0-65535
15 – Установка нескольких реле	484	1-999
15 – Установка нескольких реле	584/984	1-65535
16 – Установка нескольких регистров	484	4001-4999
16 – Установка нескольких регистров	584/984	40001–49999 (5 digit) или 400001–465535 (6 digit)

Адреса памяти ведущего устройства команды MWX

Диапазоны адресов ведущего устройства для команды MWX	
Тип данных операнда	**Диапазон DL06**
Входы X	0–777
Выходы Y	0–777
Управляющие реле C	0–1777
Биты стадий S	0–1777
Биты таймеров T	0–377
Биты счетчиков CT	0–177
Специальные реле SP	0–777
V–память V	Вся
Удаленные входы GX	0–3777
Удаленные выходы GY	0–3777

Задание числа элементов в команде MWX

Число элементов	
Тип данных операнда	**Диапазон DL06**
V–память V	Вся
Константы K	1–2000

Буфер ошибок выполнения команды MWX

Буфер ошибок выполнения	
Тип данных операнда	**Диапазон DL06**
V–память V	Вся

Пример работы с командами MRX/MWX в DirectSOFT32

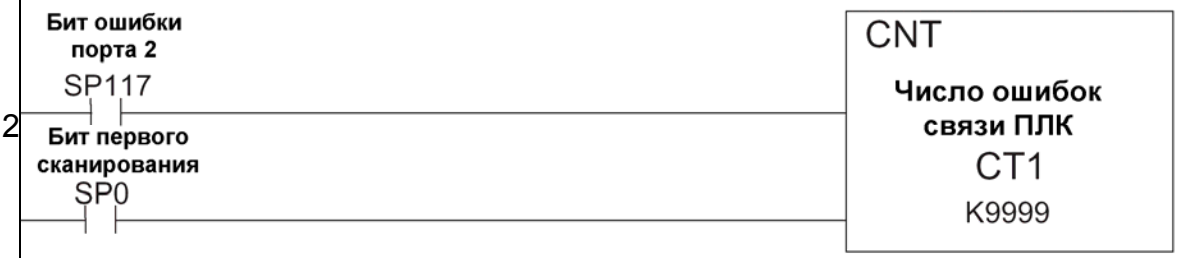
Порт 2 DL06 имеет два, связанных с ним, специальных контакта реле. (смотри Приложение Д для дополнительной информации по специальным реле порта связи). SP116 указывает “Порт занят”, а SP117 - ”Ошибка связи порта”. Бит “Порт занят” включен, в то время, когда ПЛК связывается с ведомым устройством. Если этот бит выключен, то программа может инициировать следующий сетевой запрос.
Бит “Ошибка связи порта” включается, когда ПЛК обнаружил ошибку. Он должен проверяться перед любым обращением к сети, так как бит ошибки сбросывается после того, как команда MRX или MWX выполнена. Обычно связь по сети продолжается дольше одного цикла сканирования процессора. Программа должна ждать окончания сеанса связи перед стартом следующей посылки.

Взаимная блокировка многократных чтения и записи

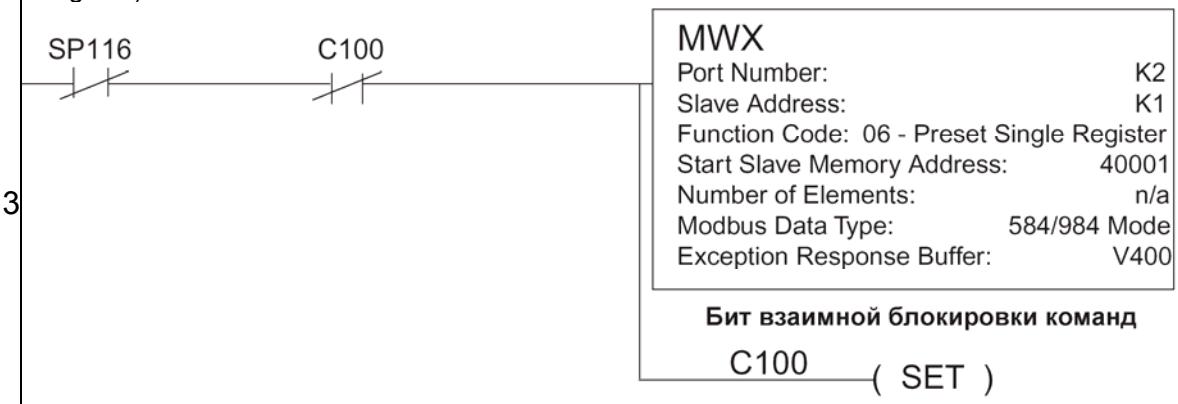
Если вы применяете многократные чтения и записи в программе RLL, то вы должны взаимно блокировать команды, чтобы обеспечить их выполнение. Если вы не применяете взаимные блокировки, то процессор выполнит только первую команду. Это происходит потому, что порт может одновременно управлять только одной транзакцией. В показанном примере, после выполнения команды RX устанавливается C100. Когда порт завершает коммуникационную задачу, C100 сбрасывается. Если вы используете Стадийное программирование RLL^{plus}, то вы можете включить каждую цепь в отдельную стадию программы для ее корректного выполнения и переходить с одной стадии на другую, давая возможность только одной цепи из программ быть активной в каждый момент времени.

SP116 будет включаться каждый раз при опросе сети. Вы должны проверить увеличение этого счетчика перед выполнением команд MWX и MRX. Возможные ошибки при которых счетчик не будет увеличиваться 1) Нет перемычки выводов RTS и CTS COM-порта. 2) порт не установлен в режим MODBUS RTU. 3) Проблема в логике программы, которая не разрешает выполнить команды MWX или MRX.

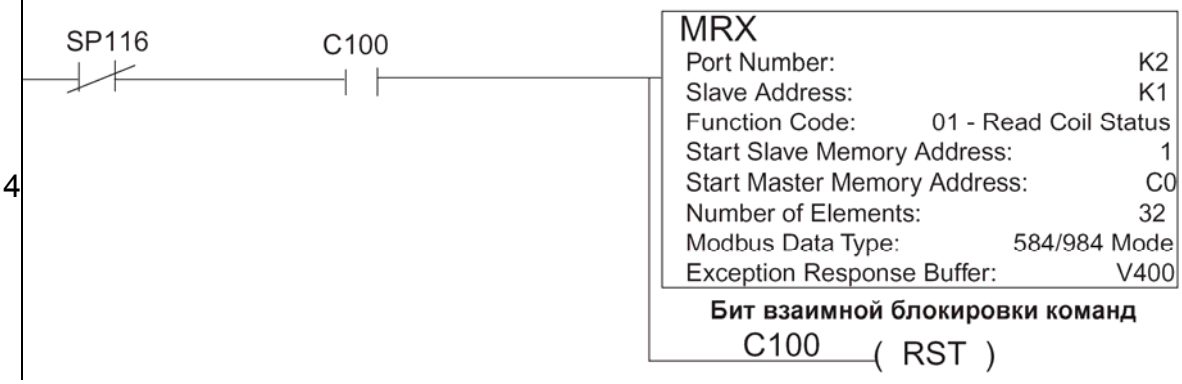
SP117 включится когда: 1) Подчиненное устройство посылает ответ "Ошибка выполнения" Если эта ситуация появилась, то просмотрите ячеки V-памяти, связанные с той командой, и обратитесь к руководству по MODICON MODBUS для дополнительной информации. 2) Проблемы с кабелем. Просмотрите монтажную схему в инструкции по эксплуатации и проверить подключения. 3) Установки связи не соответствуют друг другу. Например: Скорость обмена, контроль четности, количество стоповых бит все должно быть согласовано. 4) Опрос несуществующего в сети адреса .
При хорошем состоянии связи, SP116 будет увеличивать счетчик, а SP117 - нет. Могут появляться случайные ошибки связи из-за высокого уровня электромагнитных помех. Для каждого проекта необходимо задать какой-то допустимый "процент" ошибок по связи. Обычно количество ошибок ниже 10 % не слишком сильно влияет на пропускную способность сети.



Эта цепь делает запись по сети MODBUS в первый регистр хранения 40001 ведомого устройства с адресом 1. Записываемое значение хранится в ячейке V2000. Функциональный код (06) записывает только в один регистр. Используйте функциональный код 16, чтобы записать в несколько регистров. Только одна сетевая команда (WX, RX, MWX, MRX) может быть выполнена в одном цикле сканирования. В этом причина необходимости использования битов для взаимной блокировки. При использовании нескольких сетевых команд на одном порту используйте команду сдвига регистра (Shift Register).



Эта ветвь выполняет чтение сети MODBUS из первых 32 реле ведомого устройства с адресом 1. Данная команда последовательно разместит 32 значения реле в битовую память ведущего, начиная с C0.



Глава 5

5. Стандартные команды RLL

В этой главе…

Введение ..... 5—2
Использование булевых команд ..... 5—5
Булевые команды ..... 5—10
Булевые команды сравнения ..... 5—26
Немедленные команды ..... 5—32
Команды таймера, счетчика и регистра сдвига ..... 5—39
Команды загрузки аккумулятора/стека и вывода данных ..... 5—52
Логические команды аккумулятора ..... 5—69
Математические команды ..... 5—86
Трансцендентные функции ..... 5—118
Команды работы с битами ..... 5—120
Команды преобразования чисел в аккумуляторе ..... 5—127
Табличные команды ..... 5—141
Команды Часы / Календарь ..... 5—171
Команды управления процессором ..... 5—173
Команды управления программой ..... 5—175
Команды прерывания ..... 5—183
Команды вывода сообщений ..... 5—186
Команды MODBUS RTU ..... 5—201
Команды ASCII ..... 5—207

Введение

Микроконтроллер DL06 имеет широкий набор команд для выполнения множества различных типов действий. Эта глава покажет, как использовать каждую стандартную команду языка релейной логики (RLL). В дополнение к этим командам, Вы можете также использовать команды барабанного командоаппарата, описанного в Главе 6, или команды стадийного программирования приведенных в Главе 7 (Во второй части руководства).
Имеются два способа быстро найти требуемую команду.

- Если Вы знаете категорию команды (Булевая, Булево сравнение, и т.д.) , то используйте заголовок на верху страниц, чтобы найти страницы, которые описывают команды нужного Вам класса.
- Если Вы знаете имя команды, используйте следующую таблицу, чтобы найти страницу, которая описывает требуемую команду.

Команда	Страница
Accumulating Fast Timer (TMRAF)	5–42
Accumulating Timer (TMRA)	5–42
Add (ADD)	5–86
Add Binary (ADDB)	5–99
Add Binary Double (ADDBD)	5–100
Add Binary Top of Stack (ADDBS)	5–114
Add Double (ADDD)	5–87
Add Formatted (ADDF)	5–106
Add Real (ADDR)	5–88
Add to Top (ATT)	5–162
Add Top of Stack (ADDS)	5–110
And (AND)	5–69
And (AND) - Comparative	5–31
And (AND)	5–14
AND Bit-of-Word (ANDB)	5–15
And Double (ANDD)	5–70
And Formatted (ANDF)	5–71
And If Equal (ANDE)	5–28
And If Not Equal (ANDNE)	5–28
And Immediate (ANDI)	5–33
AND Move (ANDMOV)	5–167
And Negative Differential (ANDND)	5–22
And Not (ANDN)	5–31
And Not (ANDN)	5–14
And Not Bit-of-Word (ANDNB)	5–15
And Not Immediate (ANDNI)	5–33
And Positive Differential (ANDPD)	5–22

Команда	Страница
And Store (AND STR)	5–16
And with Stack (ANDS)	5–72
Arc Cosine Real (ACOSR)	5–119
Arc Sine Real (ASINR)	5–118
Arc Tangent Real (ATANR)	5–119
ASCII Clear Buffer (ACRB)	5–225
ASCII Compare (CMPV)	5–217
ASCII Constant (ACON)	5–187
ASCII Extract (AEX)	5–216
ASCII Find (AFIND)	5–213
ASCII Input (AIN)	5–209
ASCII Print from V–memory (PRINTV)	5–223
ASCII Print to V–memory (VPRINT)	5–218
ASCII Swap Bytes (SWAPB)	5–224
ASCII to HEX (ATH)	5–134
Binary (BIN)	5–127
Binary Coded Decimal (BCD)	5–128
Binary to Real Conversion (BTOR)	5–131
Compare (CMP)	5–81
Compare Double (CMPD)	5–82
Compare Formatted (CMPF)	5–83
Compare Real Number (CMPR)	5–85
Compare with Stack (CMPS)	5–84
Cosine Real (COSR)	5–118
Counter (CNT)	5–45
Data Label (DLBL)	5–187
Date (DATE)	5–171

Команда	Страница
Decode (DECO)	5–126
Decrement (DEC)	5–98
Decrement Binary (DECB)	5–105
Degree Real Conversion (DEGR)	5–133
Disable Interrupts (DISI)	5–184
Divide (DIV)	5–95
Divide Binary (DIVB)	5–104
Divide Binary by Top OF Stack (DIVBS)	5–117
Divide by Top of Stack (DIVS)	5–113
Divide Double (DIVD)	5–96
Divide Formatted (DIVF)	5–109
Divide Real (DIVR)	5–97
Enable Interrupts (ENI)	5–183
Encode (ENCO)	5–125
End (END)	5–173
Exclusive Or (XOR)	5–77
Exclusive Or Double (XORD)	5–78
Exclusive Or Formatted (XORF)	5–79
Exclusive OR Move (XORMOV)	5–167
Exclusive Or with Stack (XORS)	5–80
External Interrupt Program Example	5–184
Fault (FAULT)	5–186
Fill (FILL)	5–146
Find (FIND)	5–147
Find Block (FINDB)	5–169
Find Greater Than (FDGT)	5–148
For / Next (FOR) (NEXT)	5–176
Goto Label (GOTO) (LBL)	5–175
Goto Subroutine (GTS) (SBR)	5–178
Gray Code (GRAY)	5–138
HEX to ASCII (HTA)	5–135
Increment (INC)	5–98
Increment Binary (INCB)	5–105
Interrupt (INT)	5–183
Interrupt Return (IRT)	5–183
Interrupt Return Conditional (IRTC)	5–183
Invert (INV)	5–129
LCD	5–197
Load (LD)	5–57

Команда	Страница
Load Accumulator Indexed (LDX)	5–61
Load Accumulator Indexed from Data Constants (LDSX)	5–62
Load Address (LDA)	5–60
Load Double (LDD)	5–58
Load Formatted (LDF)	5–59
Load Immediate (LDI)	5–37
Load Immediate Formatted (LDIF)	5–38
Load Label (LDLBL)	5–142
Load Real Number (LDR)	5–63
Master Line Reset (MLR)	5–181
Master Line Set (MLS)	5–181
MODBUS Read from Network (MRX)	5–201
MODBUS Write to Network (MWX)	5–204
Move (MOV)	5–141
Move Memory Cartridge (MOVMC)	5–142
Multiply (MUL)	5–92
Multiply Binary (MULB)	5–103
Multiply Binary Top of Stack (MULBS)	5–116
Multiply Double (MULD)	5–93
Multiply Formatted (MULF)	5–108
Multiply Real (MULR)	5–94
Multiply Top of Stack (MULS)	5–112
No Operation (NOP)	5–173
Not (NOT)	5–19
Numerical Constant (NCON)	5–187
Or (OR)	5–73
Or (OR)- Comparative	5–30
Or (OR)	5–12
Or Bit-of-Word (ORB)	5–13
Or Double (ORD)	5–74
Or Formatted (ORF)	5–75
Or If Equal (ORE)	5–27
Or Immediate (ORI)	5–32
OR Move (ORMOV)	5–167
Or Negative Differential (ORND)	5–21
Or Not (ORN) - Comparative	5–30
Or Not (ORN)	5–12
Or Not Bit-of-Word (ORNB)	5–13
Or Not Immediate (ORNI)	5–32

Команда	Страница
OR Not Immediate Instructions Cont'd	5–33
Or Out (OR OUT)	5–17
Or Out Immediate (OROUTI)	5–34
Or Positive Differential (ORPD)	5–21
Or Store (OR STR)	5–16
Or with Stack (ORS)	5–76
Out (OUT)	5–64
Out (OUT)	5–17
Out Bit-of-Word (OUTB)	5–18
Out Double (OUTD)	5–64
Out Formatted (OUTF)	5–65
Out Immediate (OUTI)	5–34
Out Immediate Formatted (OUTIF)	5–35
Out Indexed (OUTX)	5–67
Out Least (OUTL)	5–68
Out Most (OUTM)	5–68
Pause (PAUSE)	5–25
Pop (POP)	5–65
Positive Differential (PD)	5–19
Print Message (PRINT)	5–189
Radian Real Conversion (RADR)	5–133
Read from Network (RX)	5–193
Real to Binary Conversion (RTOB)	5–132
Remove from Bottom (RFB)	5–153
Remove from Table (RFT)	5–159
Reset (RST)	5–23
Reset Bit-of-Word (RSTB)	5–24
Reset Immediate (RSTI)	5–36
Reset Watch Dog Timer (RSTWT)	5–174
Rotate Left (ROTL)	5–123
Rotate Right (ROTR)	5–124
RSTBIT	5–144
Segment (SEG)	5–137
Set (SET)	5–23
Set Bit-of-Word (SETB)	5–24
Set Immediate (SETI)	5–36
SETBIT	5–144
Shift Left (SHFL)	5–121
Shift Register (SR)	5–51
Shift Right (SHFR)	5–122

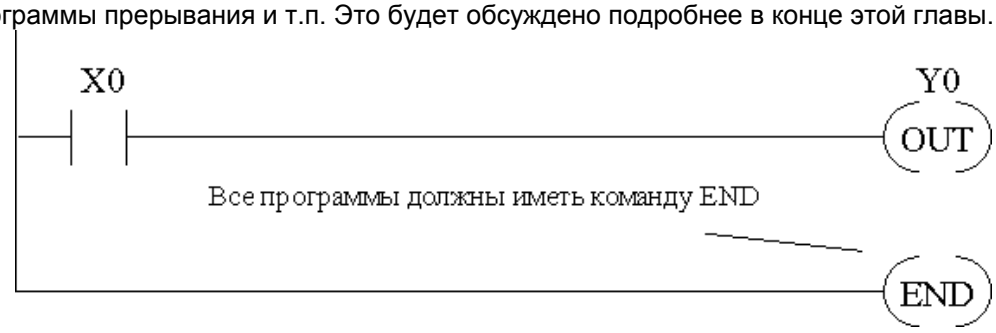
Команда	Страница
Shuffle Digits (SFLDGT)	5–139
Sine Real (SINR)	5–118
Source to Table (STT)	5–156
Square Root Real (SQRTR)	5–119
Stage Counter (SGCNT)	5–47
Stop (STOP)	5–173
Store (STR)- Comparative	5–29
Store (STR)	5–10
Store Bit-of-Word (STRB)	5–11
Store If Equal (STRE)	5–26
Store If Not Equal (STRNE)	5–26
Store Immediate (STRI)	5–32
Store Negative Differential (STRND)	5–20
Store Not (STRN)- Comparative	5–29
Store Not (STRN)	5–10
Store Not Bit-of-Word (STRNB)	5–11
Store Not Immediate (STRNI)	5–32
Store Positive Differential (STRPD)	5–20
Subroutine Return (RT)	5–178
Subroutine Return Conditional (RTC)	5–178
Subtract (SUB)	5–89
Subtract Binary (SUBB)	5–101
Subtract Binary Double (SUBBD)	5–102
Subtract Binary Top of Stack (SUBBS)	5–115
Subtract Double (SUBD)	5–90
Subtract Formatted (SUBF)	5–107
Subtract Real (SUBR)	5–91
Subtract Top of Stack (SUBS)	5–111
Sum (SUM)	5–120
Swap (SWAP)	5–170
Table Shift Left (TSHFL)	5–165
Table Shift Right (TSHFR)	5–165
Table to Destination (TTD)	5–150
Tangent Real (TANR)	5–118
Ten's Complement (BCDCPL)	5–130
Time (TIME)	5–172
Timer (TMR) and Timer Fast (TMRF)	5–40
Understanding Master Control Relays	5–181
Up Down Counter (UDC)	5–49
Write to Network (WX)	5–195

Использование булевых команд

Вы когда-нибудь удивлялись, почему многие производители ПЛК всегда приводят время выполнения 1К булевых операций? Они делают это потому, что большинство программ используют различные булевые команды. Это обычно очень простые команды, созданные для соединения входных и выходных контактов в различных последовательных и параллельных комбинациях. Так как пакет DirectSOFT позволяет использовать графические символы для построения программы, то Вы не должны знать мнемонику абсолютно всех команд. Однако, это может быть полезным, если Вы хотите исправлять команды с помощью ручного программатора.
Следующие параграфы показывают, как эти команды используются для построения простых программ.

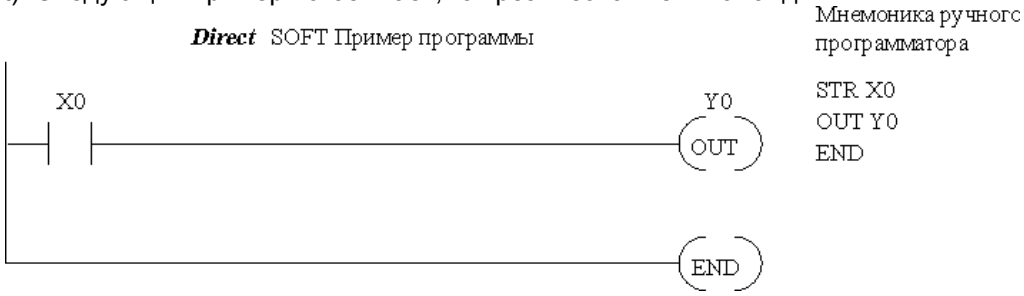
Команда END

Все программы для DL06 требуют, чтобы последней командой была команда END. Эта команда сообщает ЦПУ об окончании программы. Обычно, любые команды находящиеся после команды END не будут выполняться. Но бывают исключения из этого, как, например, подпрограммы прерывания и т.п. Это будет обсуждено подробнее в конце этой главы.



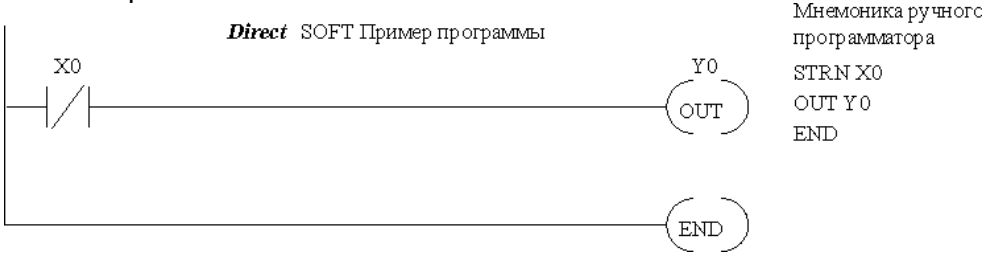
Простая логическая цепь с нормально-открытым контактом

Контакт используется, как начало логической цепи, содержащей контакт и реле. Команда STR (Store) выполняет эту функцию. Конец цепи — обмотка реле формируется командой OUT (Output). Следующий пример показывает, как реализовать эти команды.



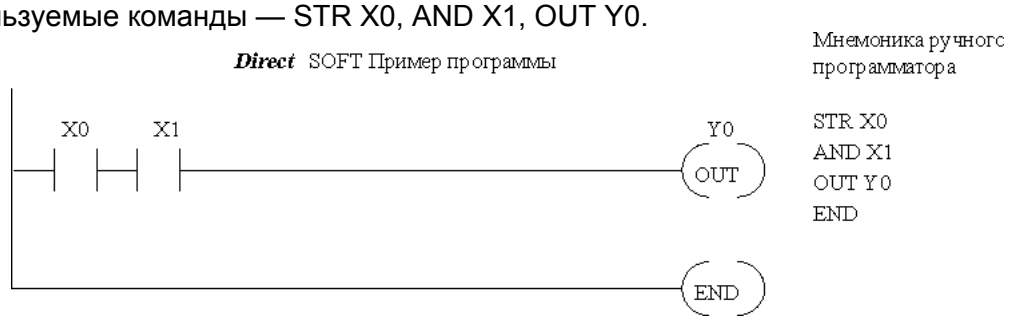
Нормально закрытый контакт

Часто употребляются нормально-закрытые контакты. Для этого используется команда STRN (Store Not). Следующий пример показывает простую логическую цепь с нормально-закрытым контактом.



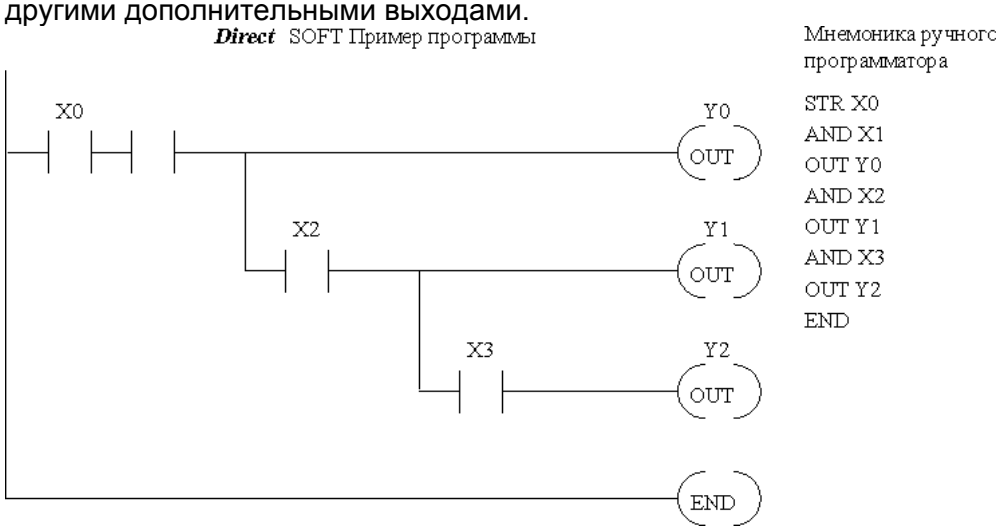
Последовательно соединенные контакты

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.



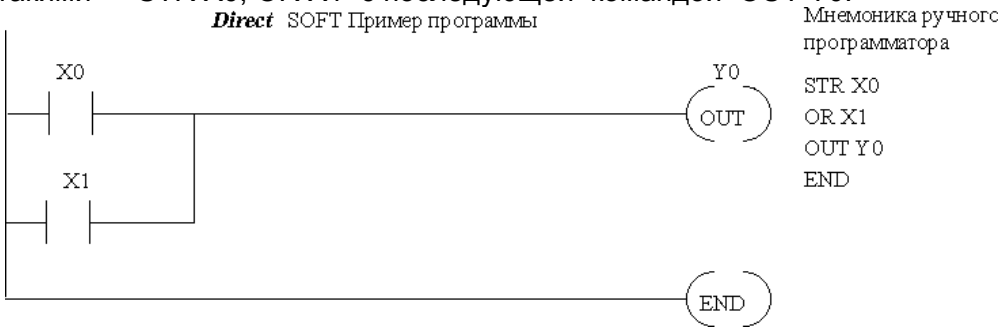
Промежуточные выходы

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.



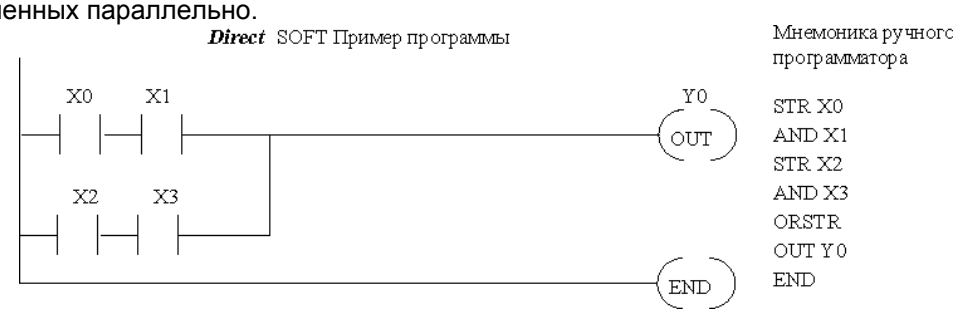
Параллельно соединенные элементы

Контакты могут быть соединены параллельно. Это позволяет сделать команда OR. Следующий пример показывает два параллельно соединенных контакта. Команды будут такими — STR X0, OR X1 с последующей командой OUT Y0.



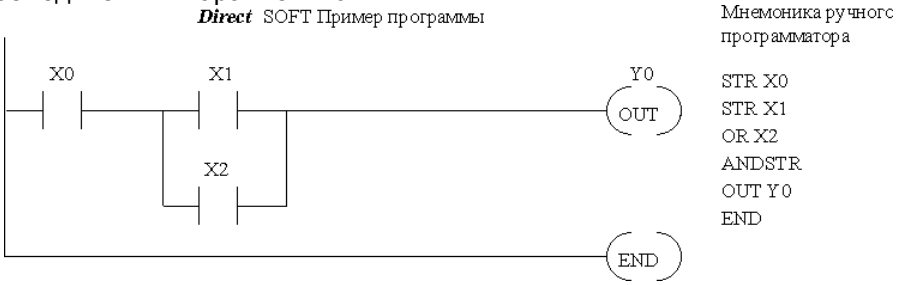
Объединение последовательных цепей параллельно

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.



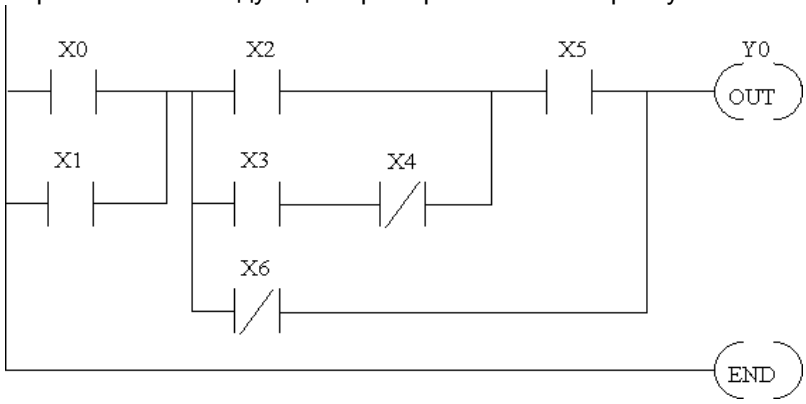
Объединение параллельных цепей последовательно

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 контактов, объединенных последовательно, и контактов, объединенных параллельно.



Комбинации цепей

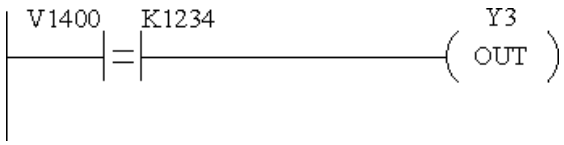
Вы можете комбинировать различные типы последовательных и параллельных цепей для решения любых приложений. Следующий пример показывает простую комбинацию цепей.



Булевое сравнение

Процессор DL06 может работать с булевыми командами сравнения, которые позволяют быстро и легко сравнивать два числа. Булевы команды сравнения обеспечивают работу с двумя 4-знаковыми величинами, используя булевы контакты. Действующие оценки сравнения: равно, не равно, больше/равно и меньше.

В следующем примере, когда величина в ячейке памяти V1400 равна постоянной величине 1234, то Y3 срабатывает.

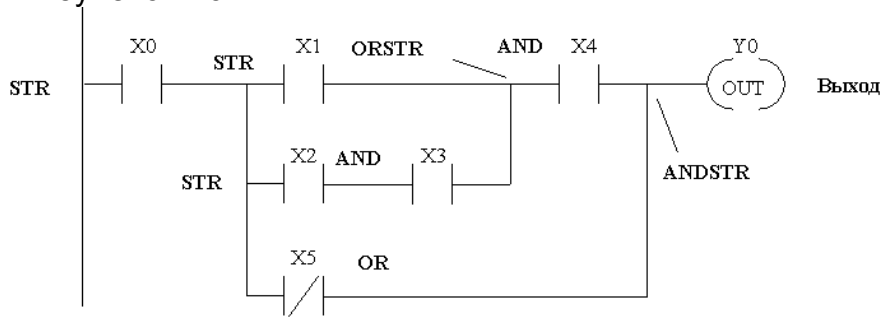


Булевый стек

Существуют ограничения на количество элементов в цепи. Это связано с тем, что ЦПУ DL06 используют 8-уровневые булевые стеки для вычисления различных элементов логического выражения.

Булевый 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 уровень вниз. Команды ANDSTR и ORSTR производят соответствующую логическую операцию над соседними уровнями стека. Так как стек имеет только восемь уровней, то если ЦПУ обнаружит цепь, которая использует больше чем восемь уровней, то случится ошибка.

Следующий пример показывает, как булевый стек используется для решения булевой 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	

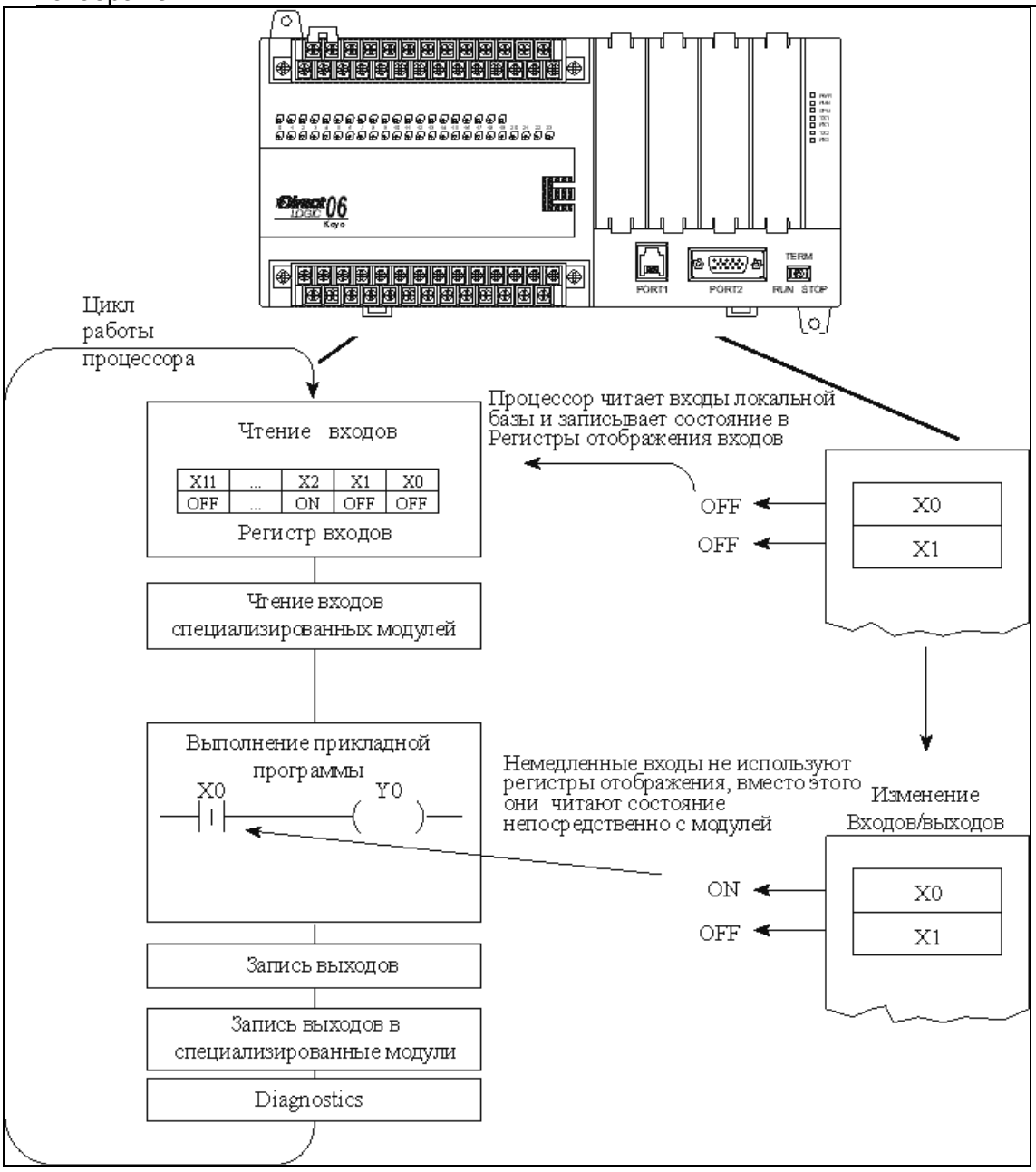
ANDSTR

1	X0 AND (NOT X5 OR X4) AND [X1 OR (X2 AND X3)]
2	
3	
.	
.	
8	

Немедленные булевые команды (Immediate Boolean)

ЦПУ DL06 обычно заканчивает цикл операций за несколько миллисекунд. Однако, в некоторых приложениях у Вас не будет возможности ждать несколько миллисекунд, пока произойдет следующее обновление входов/выходов. Процессор DL06 имеет специальные немедленные команды, которые позволяют выполнять чтение непосредственно со входов и запись непосредственно на выходы во время процесса выполнения программы. Вы можете сделать то, что обычно делается во время обновления входов/выходов. Немедленные команды используют больше времени для выполнения, потому что процесс выполнения программы прерывается, когда ЦПУ читает или записывает данные. Эта функция обычно не выполняется, когда идет цикл чтения входов и записи на выходы.

ПРИМЕЧАНИЕ. Несмотря на то, что команды немедленнного 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. Команды немедленного вывода изменяют состояния выходов и обновляют соответствующий регистр отображения.



Булевые команды

Store (STR)

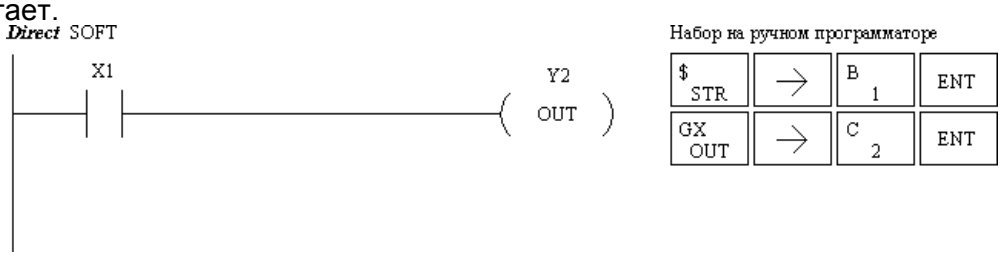
Команда Store начинает новую цепь или дополнительную ветвь в цепи с нормально открытым контактом. Состояние контакта будет тем же самым, что и состояние соответствующего регистра отображения или ячейки памяти.

Store Not (STRN)

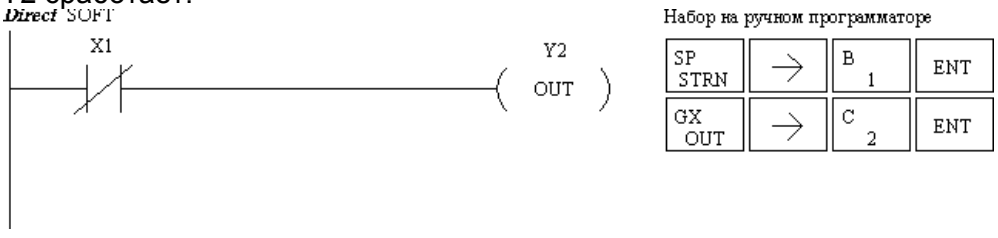
Команда Store Not начинает новую цепь или дополнительную ветвь в цепи с нормально закрытым контактом. Состояние контакта будет противоположно состоянию соответствующего регистра отображения или ячейки памяти.

Тип данных оператора		DL06 Диапазон
	A	**aaa**
Входы	X	0-777
Выходы	Y	0-777
Реле управления	C	0-1777
Стадия	S	0-1777
Таймер	T	0-377
Счетчик	CT	0-177
Специальное реле	SP	0-777

В следующем примере применения команды Store, когда вход X1 включен, выход Y2 сработает.

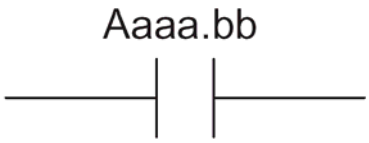


В следующем примере применения команды Store Not, когда вход X1 отключен, выход Y2 сработает.



Store Bit-of-Word (STRB)

Команда Store Bit-of-Word начинает новую цепь или дополнительную ветвь в цепи с нормально-открытым контактом. Состояние контакта будет то же самое как у бита, на который ссылаются в связанной ячейки памяти.



Store Not Bit-of-Word (STRNB)

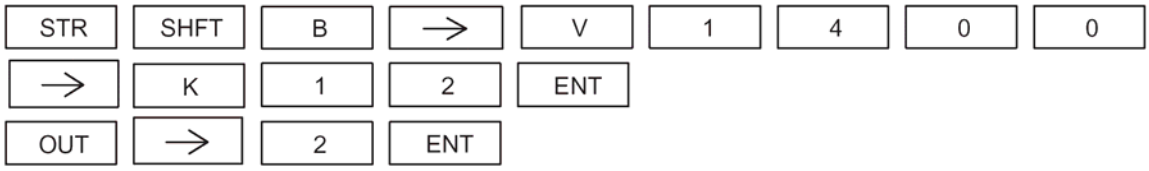
Команда Not Bit-of-Word начинает новую цепь или дополнительную ветвь в цепи с нормально-закрытым контактом. Состояние контакта будет противоположно состоянию бита, на который ссылаются в связанной ячейки памяти.

Тип данных оператора		DL06 Диапазон	
	A	**aaa**	**bb**
V-память	B	Смотри карту памяти	BCD, от 0 до 15
Указатель	PB	Смотри карту памяти	BCD, от 0 до 15

В следующем примере применения команды Store Bit-of-Word, когда 12-ый бит ячейки V-памяти V1400 включен, выход Y2 включается.

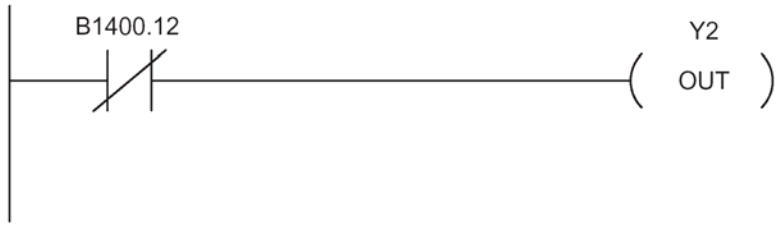
DirectSOFT32

Набор на ручном программаторе

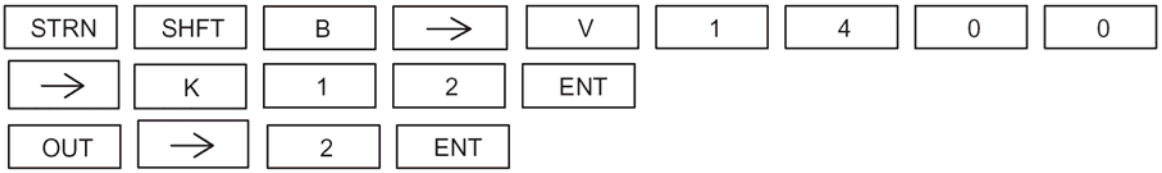


В следующем примере применения команды Store Not Bit-of-Word, когда 12-ый бит ячейки V-памяти V1400 выключен, выход Y2 включается.

DirectSOFT32

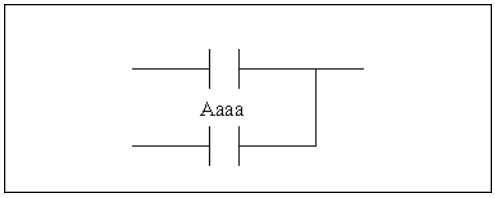


Набор на ручном программаторе



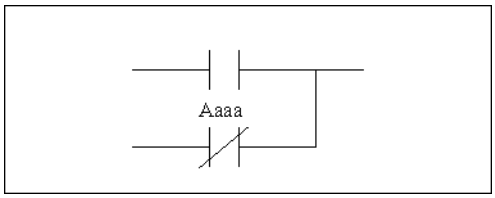
Or (OR)

Команда Or выполняет операцию логического ИЛИ для цепи, состоящего из нормально открытого контакта, соединенного параллельно с другим контактом. Состояние контакта будет тем же самым, что и соответствующая точка регистра отображения или ячейки памяти.



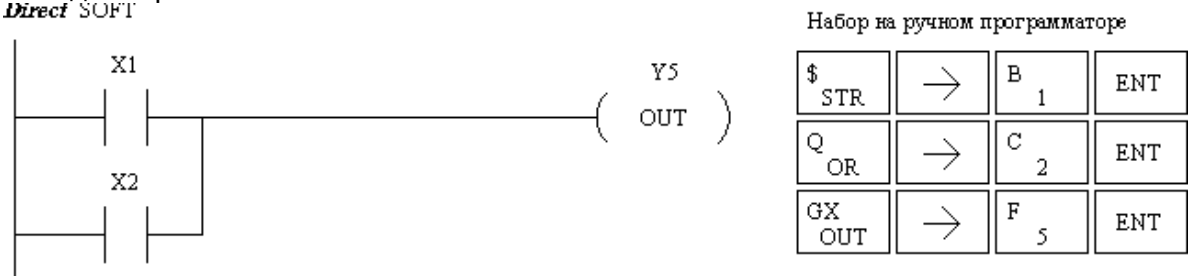
Or Not (ORN)

Команда Or Not выполняет операцию логического ИЛИ для цепи, состоящего из нормально закрытого контакта, соединенного параллельно с другим контактом. Состояние контакта будет противоположно состоянию соответствующей точки регистра отображения или ячейки памяти.

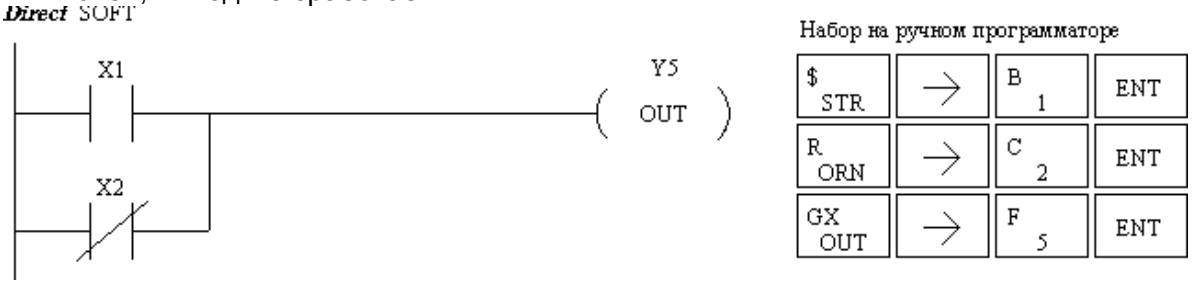


Тип данных оператора		DL06 Диапазон
	A	**aaa**
Входы	X	0-777
Выходы	Y	0-777
Реле управления	C	0-1777
Стадия	S	0-1777
Таймер	T	0-377
Счетчик	CT	0-177
Специальное реле	SP	0-777

В следующем примере применения команды Or , когда вход X1 или X2 включен, выход Y5 сработает.

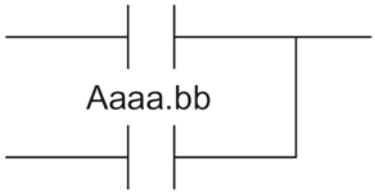


В следующем примере применения команды Or Not, когда вход X1 включен, а X2 выключен, выход Y5 сработает.



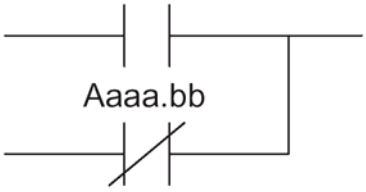
Or Bit-of-Word (ORB)

Команда Or Bit-of-Word выполняет операцию логического ИЛИ для цепи с нормально-открытым контактом, соединенного параллельно с другим контактом. Состояние контакта будет то же самое как у бита, на который ссылаются в связанной ячейки памяти.



Or Not Bit-of-Word (ORNB)

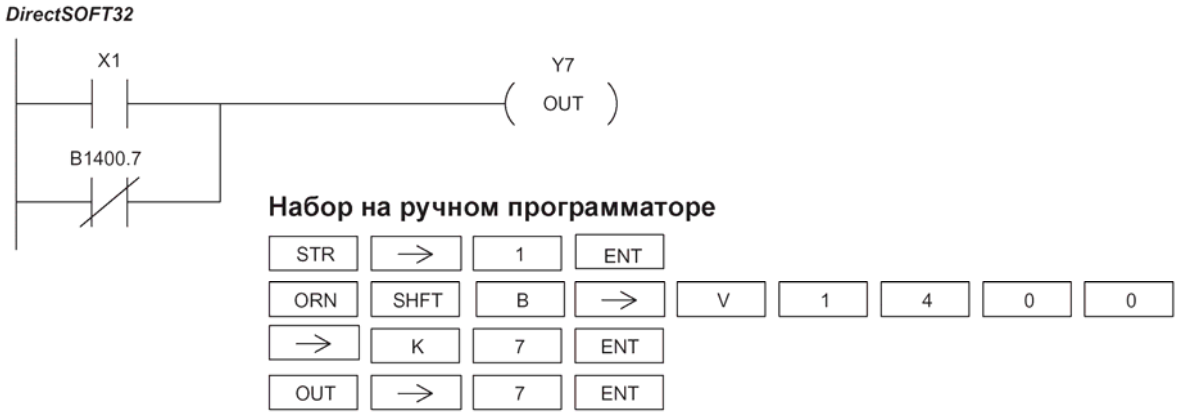
Команда Or Not Bit-of-Word выполняет операцию логического ИЛИ для цепи с нормально-закрытым контактом, соединенного параллельно с другим контактом. Состояние контакта будет противоположно состоянию бита, на который ссылаются в связанной ячейки памяти.



Тип данных оператора		DL06 Диапазон	
	A	**aaa**	**bb**
V-память	B	Смотри карту памяти	BCD, от 0 до 15
Указатель	PB	Смотри карту памяти	BCD, от 0 до 15

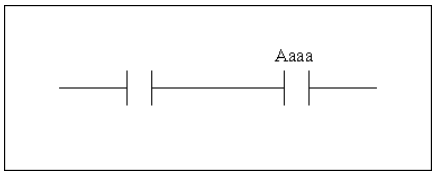
В следующем примере применения команды Or Bit-of-Word, когда вход X1 или 7-ой бит ячейки V-памяти V1400 включен, выход Y5 включается.

В следующем примере применения команды Or Not Bit-of-Word, когда вход X1 включен или 7-ой бит ячейки V-памяти V1400 выключен, выход Y2 включается.



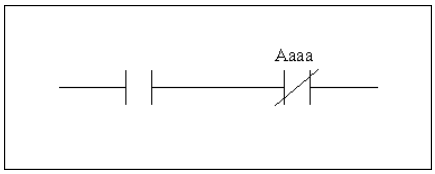
And (AND)

Команда And выполняет операцию логического И для цепи, состоящей из нормально открытого контакта, соединенного параллельно с другим контактом. Состояние контакта будет тем же самым, что и состояние соответствующего регистра отображения или ячейки памяти.



And Not (ANDN)

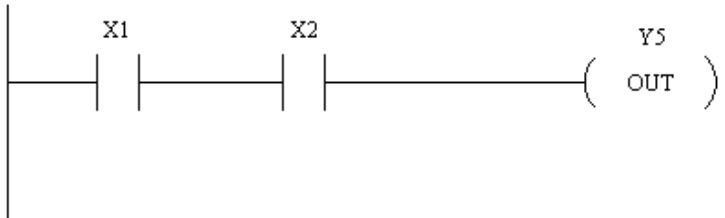
Команда And Not выполняет операцию логического И для цепи, состоящего из нормально закрытого контакта, соединенного параллельно с другим контактом. Состояние контакта будет противоположным состоянию соответствующего регистра отображения или ячейки памяти.



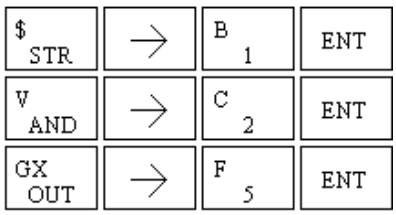
Тип данных оператора		DL06 Диапазон
	A	**aaa**
Входы	X	0-777
Выходы	Y	0-777
Реле управления	C	0-1777
Стадия	S	0-1777
Таймер	T	0-377
Счетчик	CT	0-177
Специальное реле	SP	0-777

В следующем примере применения команды And, когда вход X1 или X2 включен, выход Y5 сработает.

Direct SOFT

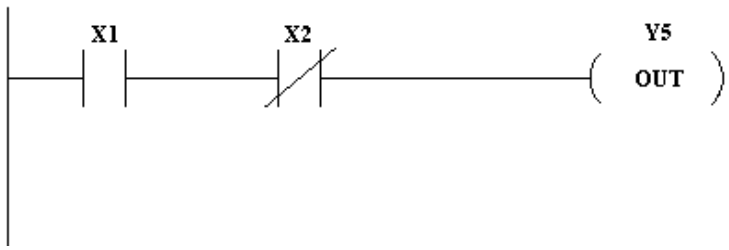


Набор на ручном программаторе

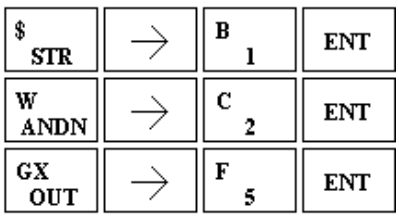


В следующем примере применения команды And Not, когда вход X1 включен и вход X2 выключен, выход Y5 сработает.

Direct SOFT



Набор на ручном программаторе



And Bit-of-Word (ANDB)

Команда And Bit-of-Word выполняет операцию логического И для цепи с нормально-открытым контактом, соединенного последовательно с другим контактом. Состояние контакта будет то же самое как у бита, на который ссылаются в связанной ячейки памяти.

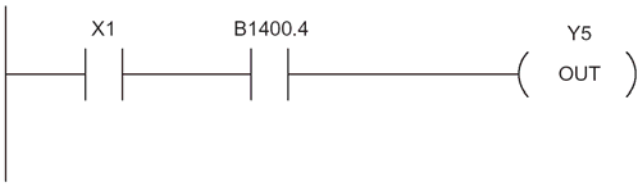
And Not Bit-of-Word (ANDNB)

Команда And Not Bit-of-Word выполняет операцию логического И для цепи с нормально-закрытым контактом, соединенного последовательно с другим контактом. Состояние контакта будет противоположно состоянию бита, на который ссылаются в связанной ячейки памяти.

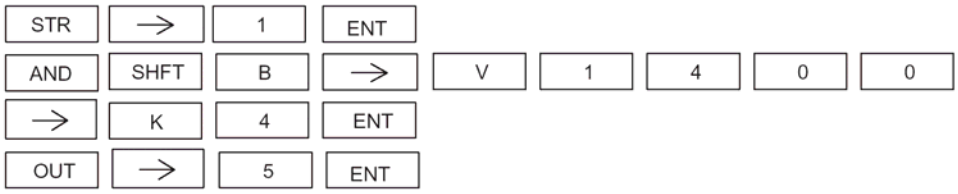
Тип данных оператора		DL06 Диапазон	
	A	**aaa**	**bb**
V-память	B	Смотри карту памяти	BCD, от 0 до 15
Указатель	PB	Смотри карту памяти	BCD, от 0 до 15

В следующем примере применения команды And Bit-of-Word, когда вход X1 и 4-ый бит ячейки V-памяти V1400 включен, выход Y5 включается.

DirectSOFT32

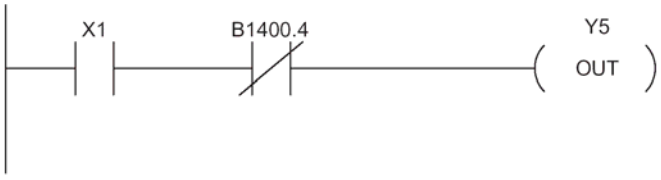


Набор на ручном программаторе

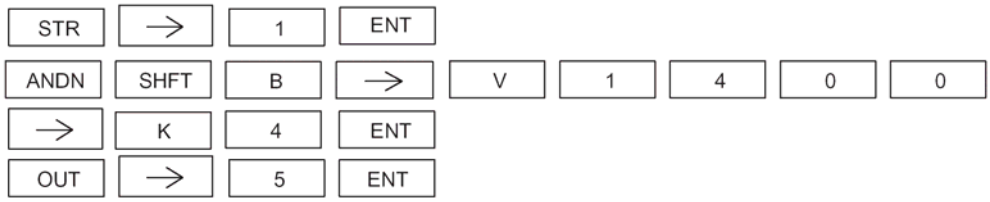


В следующем примере применения команды And Not Bit-of-Word, когда вход X1 включен и 4-ый бит ячейки V-памяти V1400 выключен, выход Y2 включается.

DirectSOFT32

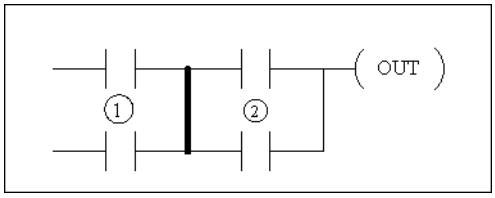


Набор на ручном программаторе



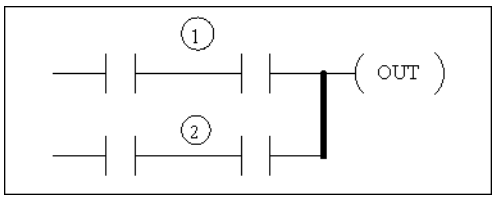
And Store (AND STR)

Команда And Store выполняет операцию логического И для цепи, состоящей из двух последовательных ветвей. Обе ветви должны начинаться с команды Store.



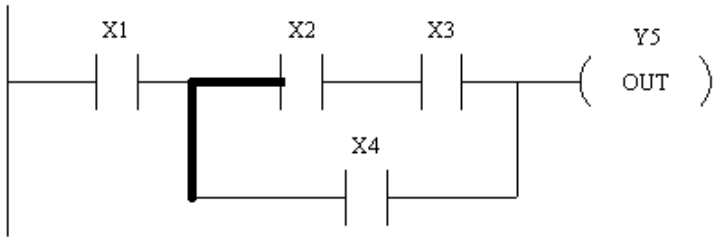
Or Store (OR STR)

Команда Or Store выполняет операцию логического ИЛИ для цепи, состоящей из двух параллельных ветвей. Обе ветви должны начинаться с команды Store.



В следующем примере применения команды And Store была выполнена операция AND над ветвью, состоящей из контактов X2, X3 и X4, и ветвью, состоящей из контакта X1.

Direct SOFT

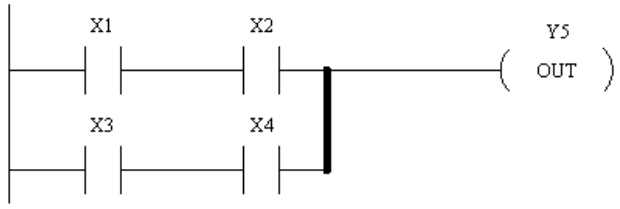


Набор на ручном программаторе

$ STR	→	B 1	ENT
$ STR	→	C 2	ENT
V AND	→	D 3	ENT
Q OR	→	E 4	ENT
L ANDST	ENT		
GX OUT	→	F 5	ENT

В следующем примере применения команды Or Store была выполнена операция OR над ветвью, состоящей из контактов X1 и X2, и ветвью, состоящей из контактов X3 и X4.

Direct SOFT

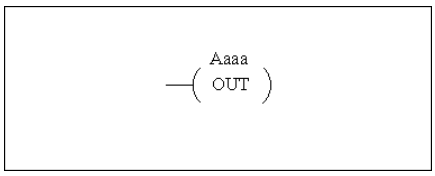


Набор на ручном программаторе

$ STR	→	B 1	ENT
V AND	→	C 2	ENT
$ STR	→	D 3	ENT
V AND	→	E 4	ENT
M ORST	ENT		
GX OUT	→	F 5	ENT

Out (OUT)

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.

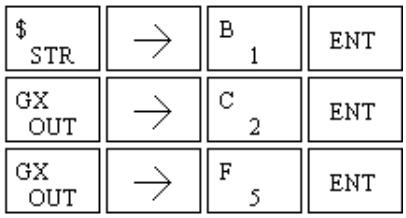


Тип данных оператора		DL06 Диапазон
	A	**aaa**
Входы	X	0-777
Выходы	Y	0-777
Реле управления	C	0-1777

В следующем примере Out, когда вход X1 включен, выходы Y2 и Y5 сработают.

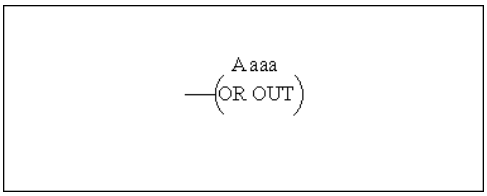
Direct SOFT

Набор на ручном программаторе



Or Out (OR OUT)

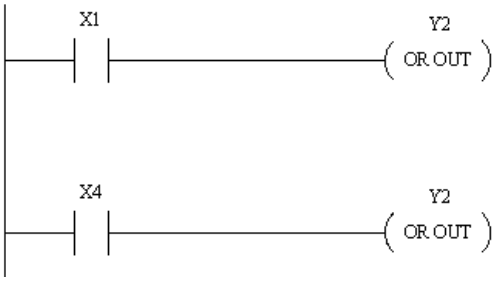
Команда Or Out позволяет использовать несколько цепей дискретной логики для управления одним выходом. Возможно использование нескольких команд Or Out, ссылающихся на один и тот же выход, так как операция ИЛИ выполняется одновременно над всеми контактами, управляющими выходом. Если состояние любого из звеньев включено, то выход будет также включен



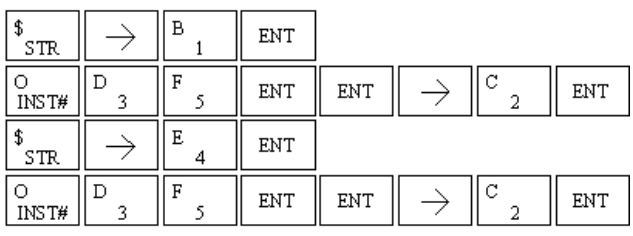
Тип данных оператора		DL06 Диапазон
	A	**aaa**
Входы	X	0-777
Выходы	Y	0-777
Реле управления	C	0-1777

В следующем примере, когда X1 или X4 включены, Y2 сработает

Direct SOFT

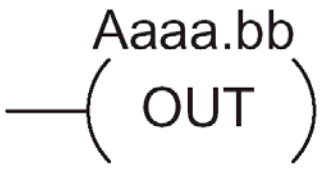


Набор на ручном программаторе



Out Bit-of-Word (OUTB)

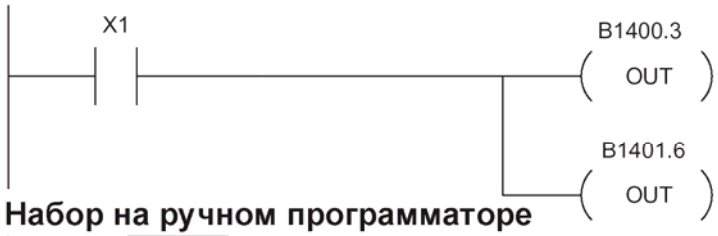
Команда Out Bit-of-Word отражает состояние цепи (вкл\выкл) и выводит дискретное состояние (вкл\выкл) в определенный бит в указанной ячейки памяти. Многократная команда Out Bit-of-Word, ссылающаяся на один и тот же бит одного и того же самого слова не должна использоваться, поскольку только последняя команда Out будет управлять состоянием бита.



Тип данных оператора		DL06 Диапазон	
	A	**aaa**	**bb**
V-память	B	Смотри карту памяти	BCD, от 0 до 15
Указатель	PB	Смотри карту памяти	BCD, от 0 до 15

В следующем примере применения команды Out Bit-of-Word, когда включен вход X1 и 3-ий бит ячейки V-памяти V1400 и 6-ой бит ячейки V-памяти V1401 будут включены.

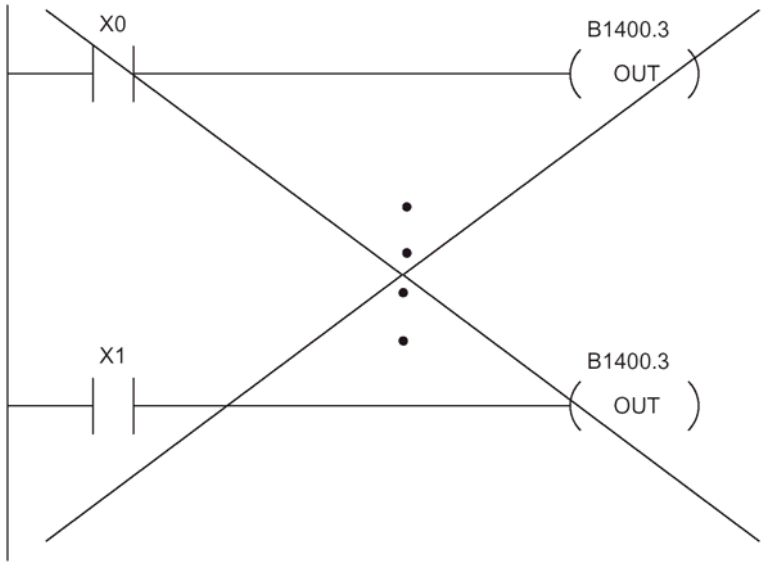
DirectSOFT32



Набор на ручном программаторе

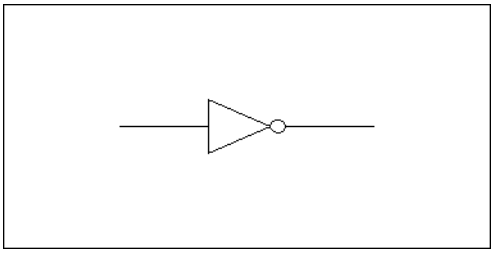
STR	→	1	ENT					
OUT	SHFT	B	→	V	1	4	0	0
→	K	3	ENT					
OUT	SHFT	B	→	V	1	4	0	1
→	K	6	ENT					

В следующем примере применения команды Out Bit-of-Word содержится две команды Out Bit-of-Word, управляющие одним и тем же битом в одном и том же слове памяти. Последняя команда управления 3-им битом ячейки V1400, в конечном счете, управляется последней ветвью программы, ссылающейся на этот бит. X1 отменит логическое состояние, управляемое X0. Чтобы избежать этой ситуации, не используйте несколько раз данную команду.

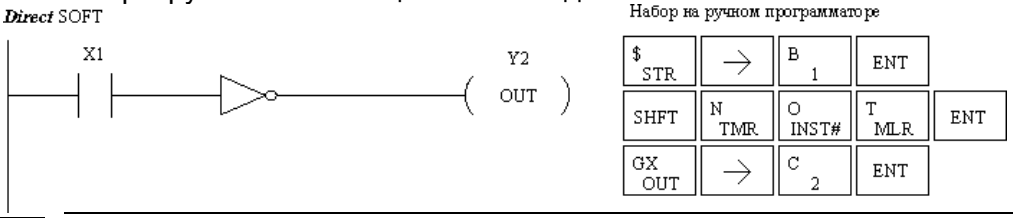


Not (NOT)

Команда Not инвертирует состояние цепи в точке команды.



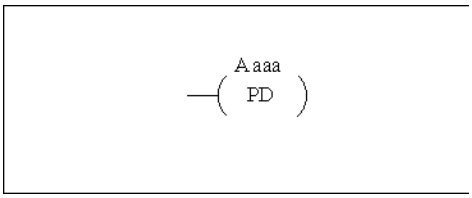
В следующем примере, когда X1 выключен, Y2 сработает. Это потому, что команда Not инвертирует состояние цепи в команде Not.



ПРИМЕЧАНИЕ. Direct SOFT версии 1.1i и более поздних версий поддерживает использование команды NOT. Эта цепь не может быть создана или отображена в версиях Direct SOFT более раних чем 1.1i.

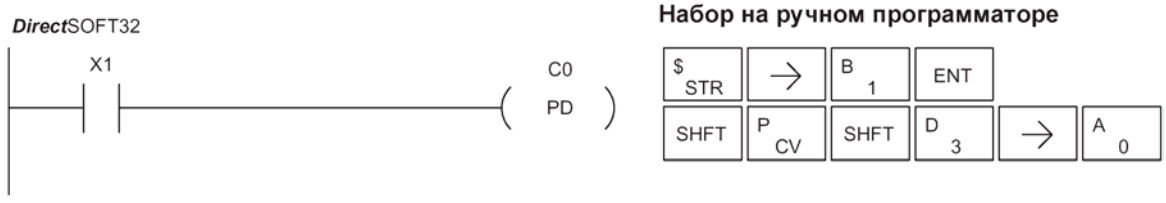
Positive Differential (PD)

Команду Positive Differential обычно называют «одиночный импульс».
Когда входная логика меняет сигнал с ВЫКЛ на ВКЛ., выход сработает на время одного цикла сканирования ЦПУ. Затем выход останется в состоянии ВЫКЛ. до тех пор пока вход не произведет следующее изменение.



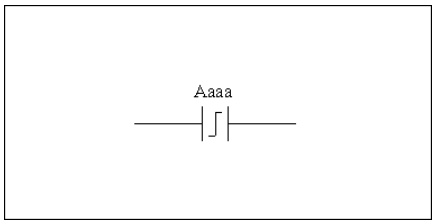
Тип данных оператора	DL06 Диапазон
A	**aaa**
Входы X	0-777
Выходы Y	0-777
Реле управления C	0-1777

В следующем примере, каждый раз, когда X1 переходит из состояния ВЫКЛ. в состояние ВКЛ, C0 сработает на время одного цикла сканирования.



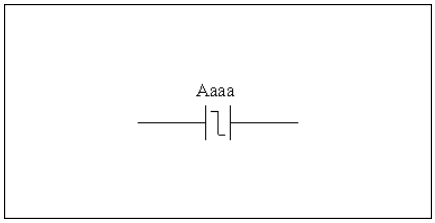
Store Positive Differential (STRPD)

Команда Store Positive Differential начинает новую цепь или новую ветвь цепи с нормально открытым контактом. Контакт замыкается на один цикл сканирования ЦПУ, когда состояние соответствующего бита регистра отображения изменит состояние ВЫКЛ на ВКЛ (Off/On). Затем контакт останется в разомкнутом состоянии до тех пор пока не произойдет следующее изменение состояния бита ВЫКЛ/ВКЛ (символ внутри контакта отражает это изменение). Эту функцию иногда называют «одиночный импульс».



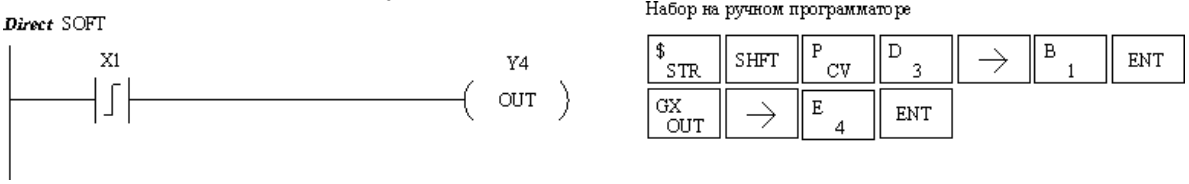
Store Negative Differential (STRND)

Команда Store Negative Differential начинает новую цепь или новую ветвь цепи с нормально открытым контактом. Контакт замыкается на один цикл сканирования ЦПУ, когда состояние соответствующего бита регистра отображения изменит состояние ВКЛ на ВЫКЛ (On/Off). Затем контакт останется в разомкнутом состоянии, до тех пор пока не произойдет следующее изменение состояния бита ВКЛ/ВЫКЛ (символ внутри контакта отражает это изменение).

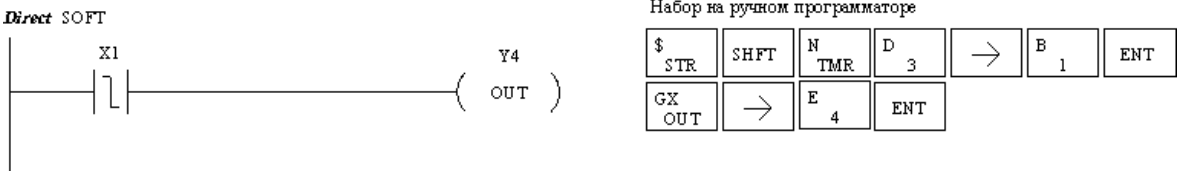


Тип данных оператора		DL06 Диапазон
	A	**aaa**
Входы	X	0-777
Выходы	Y	0-777
Реле управления	C	0-1777
Стадия	S	0-1777
Таймер	T	0-377
Счетчик	CT	0-177

В следующем примере, каждый раз, когда X1 переходит из состояния ВЫКЛ. в состояние ВКЛ., выход Y4 будет включен на время одного цикла сканирования.

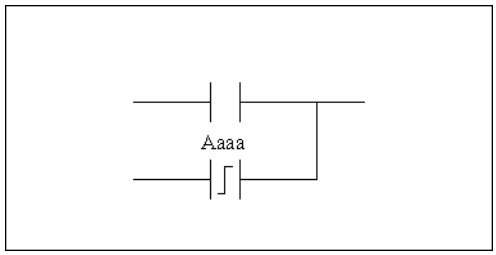


В следующем примере, каждый раз, когда X1 переходит из состояния ВКЛ. в состояние ВЫКЛ., выход Y4 будет включен на время одного цикла сканирования.



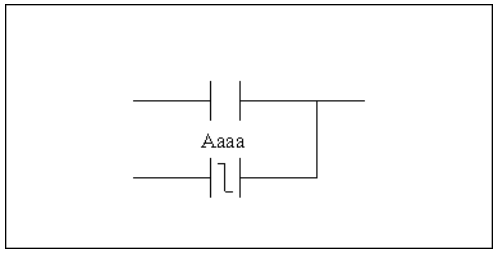
Or Positive Differential (ORPD)

Команда Or Positive Differential — это нормально открытый контакт параллельный с другим контактом цепи. Состояние контакта остается разомкнутым на один цикл сканирования ЦПУ, когда состояние соответствующего бита регистра отображения изменит состояние ВЫКЛ на ВКЛ (Off/On), замыкая его на один скан. Затем контакт останется в разомкнутом состоянии, до тех пор пока вход не произведет следующее изменение ВЫКЛ / ВКЛ.



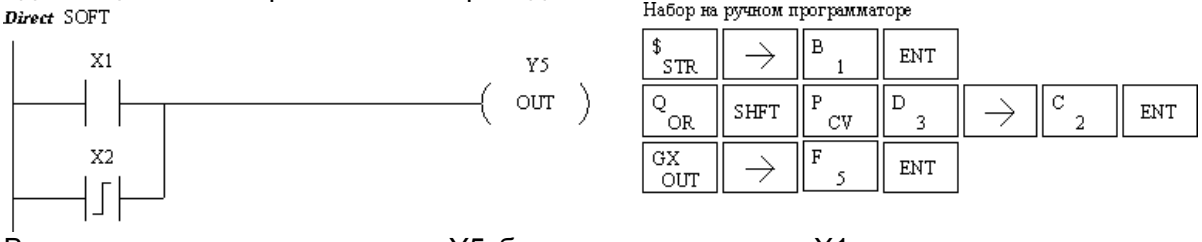
Or Negative Differential (ORRND)

Команда Or Negative Differential — это нормально открытый контакт параллельный с другим контактом цепи. Состояние контакта остается разомкнутым на один цикл сканирования ЦПУ, когда состояние соответствующего бита регистра отображения изменит состояние ВКЛ на ВЫКЛ (On /Off), замыкая его на один скан. Затем контакт останется в разомкнутом состоянии, до тех пор пока вход не произведет следующее изменение ВКЛ / ВЫКЛ.

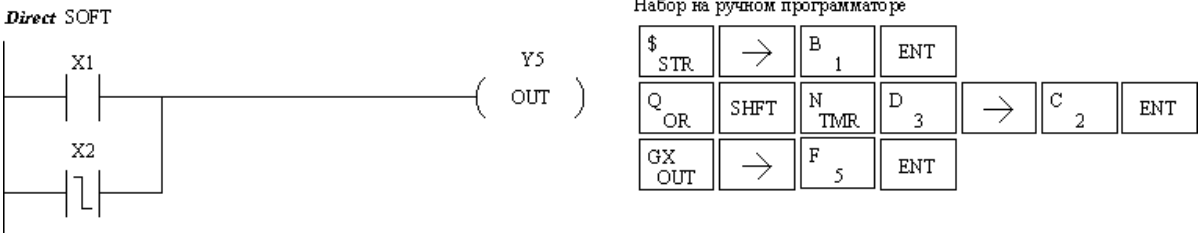


Тип данных оператора		DL06 Диапазон
	A	**aaa**
Входы	X	0-777
Выходы	Y	0-777
Реле управления	C	0-1777
Стадия	S	0-1777
Таймер	T	0-377
Счетчик	CT	0-177

В следующем примере, выход Y5 будет включен когда X1 включен или на время одного цикла сканирования X2 переходит из состояния ВЫКЛ в состояние ВКЛ.

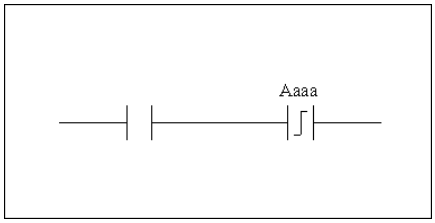


В следующем примере, выход Y5 будет включен, когда X1 включен или на время одного цикла сканирования X2 переходит из состояния ВКЛ в состояние ВЫКЛ.



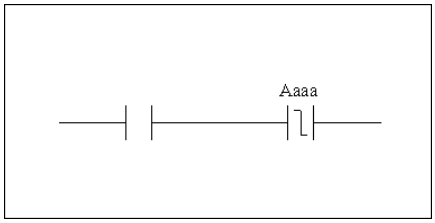
And Positive Differential (ANDPD)

Команда And Positive Differential это нормально открытый контакт последовательно подключенный к другому контакту цепи. Состояние контакта остается разомкнутым до тех пор, пока состояние соответствующего бита регистра отображения изменит состояние ВЫКЛ на ВКЛ (Off/On), замыкая его на один скан. Затем контакт останется в разомкнутом состоянии, до тех пор пока вход не произведет следующее изменение ВЫКЛ / ВКЛ.



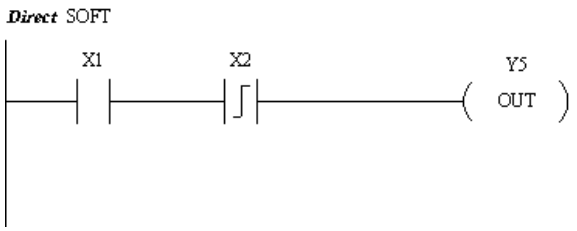
And Negative Differential (ANDND)

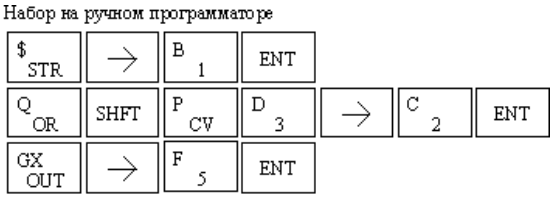
Команда And Negative Differential это нормально открытый контакт последовательно подключенный к другому контакту цепи. Состояние контакта остается разомкнутым до тех пор пока состояние соответствующего бита регистра отображения изменит состояние ВКЛ на ВЫКЛ (On /Off), замыкая его на один скан. Затем контакт останется в разомкнутым, до тех пор пока вход не произведет следующее изменение ВКЛ / ВЫКЛ.



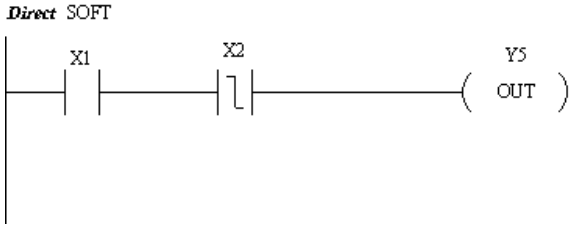
Тип данных оператора		DL06 Диапазон
	A	**aaa**
Входы	X	0-777
Выходы	Y	0-777
Реле управления	C	0-1777
Стадия	S	0-1777
Таймер	T	0-377
Счетчик	CT	0-177

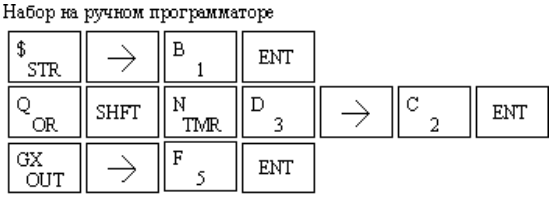
В следующем примере, выход Y5 будет включен на время одного цикла сканирования, когда X1 включен и X2 переходит из состояния ВЫКЛ в состояние ВКЛ.





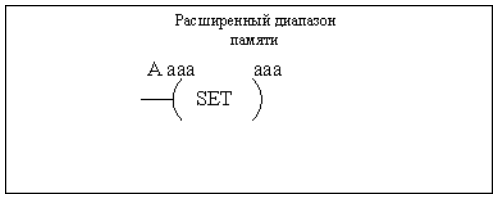
В следующем примере, выход Y5 будет включен на время одного цикла сканирования, когда X1 включен и X2 переходит из состояния ВКЛ в состояние ВЫКЛ.





Set (SET)

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 ее не сбросят командой Reset.
Необязательно сохранять состояние ON входа управления командой Set.

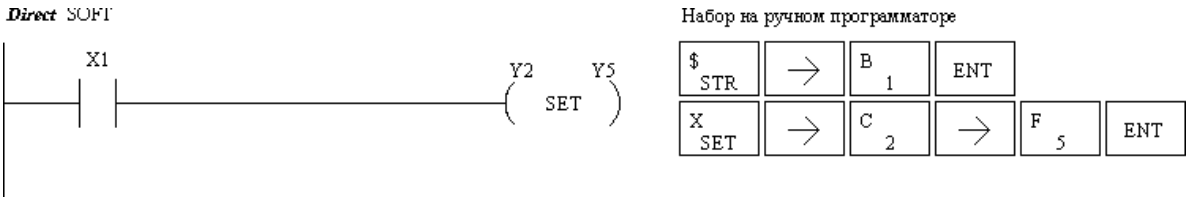


Reset (RST)

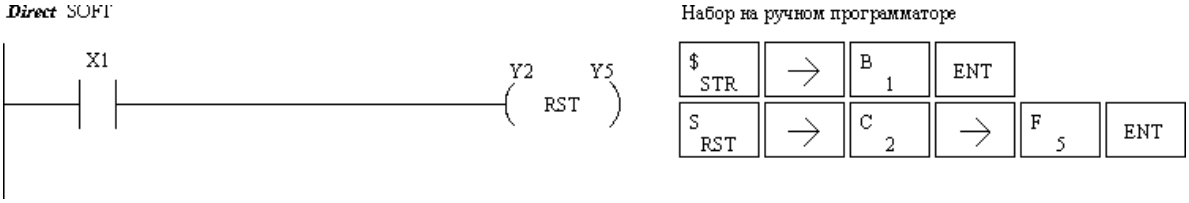
Команда Reset сбрасывает или выключает точку регистра отображения / ячейку памяти или диапазон точек регистра отображения / ячеек памяти. Как только точка/ячейка сброшена. она будет оставаться отключенной до тех пор, пока ее не установят командой Set.
Необязательно сохранять состояние ON входа управления командой RST.

Тип данных оператора		DL06 Диапазон
	A	**aaa**
Входы	X	0-777
Выходы	Y	0-777
Реле управления	C	0-1777
Стадия	S	0-1777
Таймер	T	0-377
Счетчик	CT	0-177

В следующем примере, когда X1 включен, включаются выходы Y2- Y5.

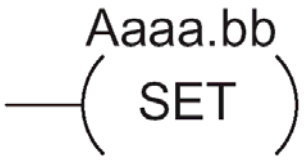


В следующем примере, когда X1 включен, Y2- Y5 будут сброшены или отключены.



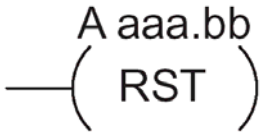
Set Bit-of-Word (SETB)

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 сохранять состояние ON входа управления командой Set Bit-of-Word.



Reset Bit-of-Word (RSTB)

Команда Reset Bit-of-Word сбрасывает или выключает бит в ячейке V-памяти. Как только бит сброшен, нет необходимости, сохранять включенное состояние входа управления.

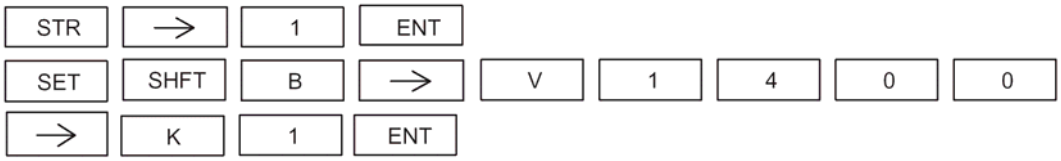


Тип данных оператора		DL06 Диапазон	
	A	**aaa**	**bb**
V-память	B	Смотри карту памяти	BCD, от 0 до 15
Указатель	PB	Смотри карту памяти	BCD, от 0 до 15

В следующем примере когда X1 включен, то бит 1 ячейки V1400 будет установлен.

DirectSOFT32

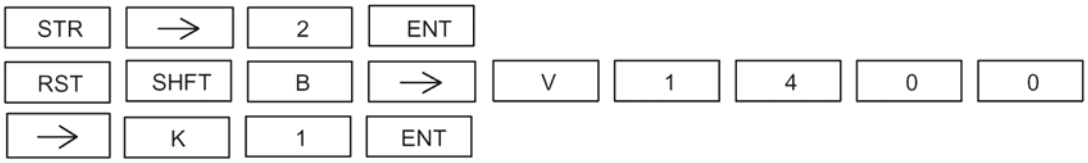
Набор на ручном программаторе



В следующем примере когда X2 включен, то бит 1 ячейки V1400 будет сброшен.

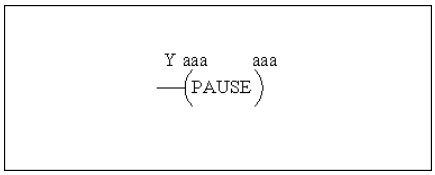
DirectSOFT32

Набор на ручном прогарамматоре



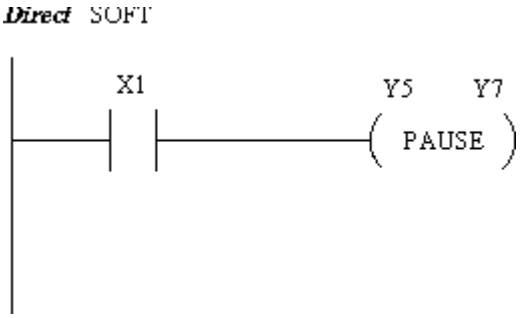
Pause (PAUSE)

Команда Pause отключает изменение выхода на заданном диапазоне выходов. Программа релейной логики будет продолжать выполнение и изменять регистр отображения, однако выходы модулей будут выключены в диапазоне, заданном в команде Pause(Пауза).



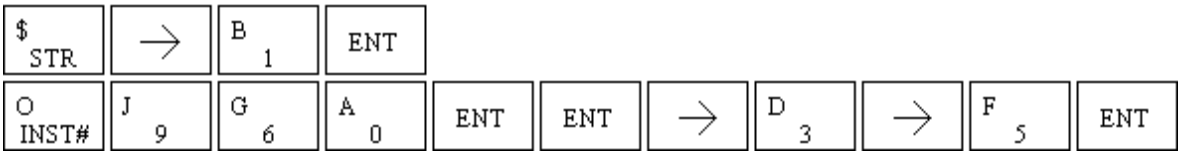
Тип данных оператора	DL06 Диапазон
	aaa
Выходы Y	0-777

В следующем примере, когда X1 в состоянии ВКЛ., Y5-Y7 будут выключены в выходном модуле. Выполнение программы не будет отражаться на них.



Так как на ручном программаторе нет специальной клавиши Pause для ввода команды можно использовать номер команды (# 960) или набрать имя команды по буквам.

Набор на ручном программаторе

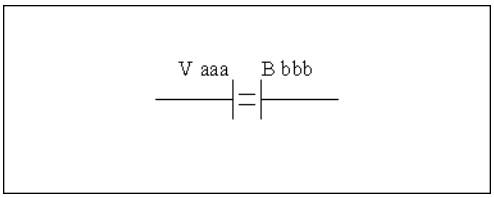


Иногда, необходимо работать в обычном режиме с некоторыми выходами из указанного диапазоне Pause. В этом случае, используйте Aux 58 отменяющую действие инструкция Pause.

Булевые команды сравнения

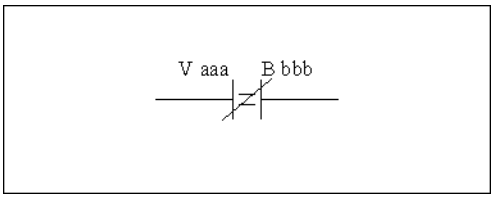
Store If Equal (STRE)

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)

Команда Store If Not Equal начинает новую цепь или дополнительную ветвь в цепи с нормально закрытым контактом сравнения. Контакт будет включен, когда Vaaa не равен Bbbb.



Тип данных операнда		DL06 Диапазон	
	B	aaa	bbb
V-память	V	Смотри карту памяти	Смотри карту памяти
Указатель (Pointer)	P	Смотри карту памяти	Смотри карту памяти
Константа	K	—	0–9999

В следующем примере, когда значение в ячейке памяти V2000 = 4933, Y3 сработает.

Набор на ручном программаторе

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0
→	E 4	J 9	D 3	D 3	ENT		
GX OUT	→	D 3	ENT				

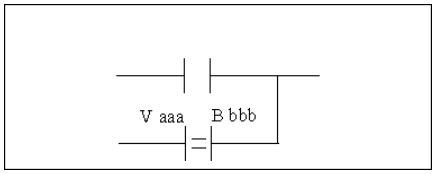
В следующем примере, когда значение в ячейке памяти V2000 не равно 5060, Y3 сработает.

Набор на ручном программаторе

SP STRN	SHFT	E 4	→	C 2	A 0	A 0	A 0
→	F 5	A 0	G 6	A 0	ENT		
GX OUT	→	D 3	ENT				

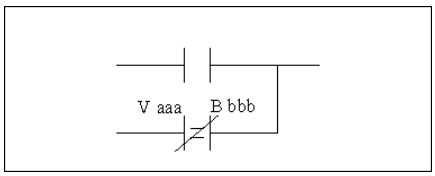
Or If Egual (ORE)

Команда Or If Equal соединяет нормально открытый контакт сравнения параллельно с другим контактом. Контакт будет включен, когда Vaaa = Bbbb.



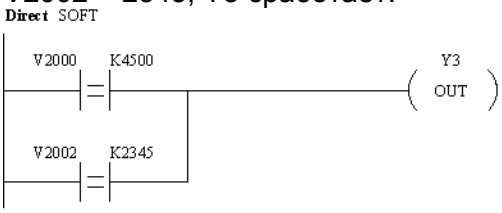
Or If Not Egual (ORNE)

Команда Or If Not Equal соединяет нормально закрытый контакт сравнения параллельно с другим контактом. Контакт будет включен, когда Vaaa не равен Bbbb.



Тип данных операнда		DL06 Диапазон	
	B	aaa	bbb
V-память	V	Смотри карту памяти	Смотри карту памяти
Указатель (Pointer)	P	Смотри карту памяти	Смотри карту памяти
Константа	K	—	0–9999

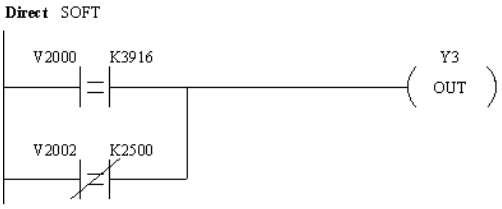
В следующем примере, когда значение в ячейке памяти V2000 = 4500 или в ячейке V2002 = 2345, Y3 сработает.



Набор на ручном программаторе

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
E 4	F 5	A 0	A 0	ENT				
Q OR	SHFT	E 4	→	C 2	A 0	A 0	C 2	→
C 2	D 3	E 4	F 5	ENT				
GX OUT	→	D 3	ENT					

В следующем примере, когда значение в ячейке памяти V2000 = 3916 или в ячейке V2002 не равно 2500, Y3 сработает.

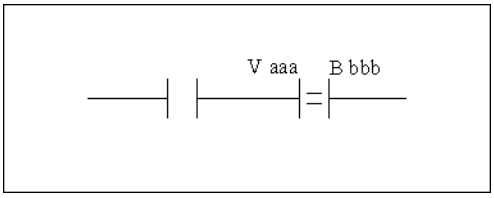


Набор на ручном программаторе

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
D 3	J 9	B 1	G 6	ENT				
R ORN	SHFT	E 4	→	C 2	A 0	A 0	C 2	→
C 2	F 5	A 0	A 0	ENT				
GX OUT	→	D 3	ENT					

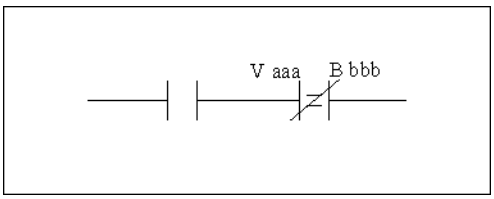
And If Equal (ANDE)

Команда And If Equal соединяет нормально открытый контакт сравнения последовательно с другим контактом. Контакт будет включен, когда Vaaa = Bbbb



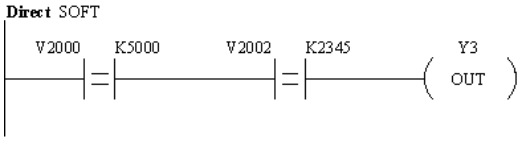
And If Not Equal (ANDNE)

Команда And If Not Equal соединяет нормально закрытый контакт сравнения последовательно с другим контактом. Контакт будет включен, когда Vaaa не равно Bbbb.



Тип данных операнда		DL06 Диапазон	
	A/B	aaa	bbb
V-память	V	Смотри карту памяти	Смотри карту памяти
Указатель	P	Смотри карту памяти	Смотри карту памяти
Константа	K	—	0–9999

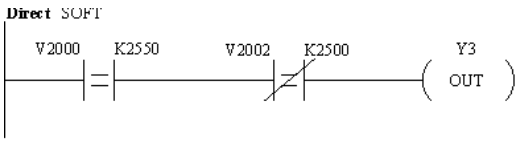
В следующем примере, когда значение в ячейке памяти V2000 = 5000 и в ячейке V2002 = 2345, Y3 сработает.



Набор на ручном программаторе

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
F 5	A 0	A 0	A 0	ENT				
V AND	SHFT	E 4	→	C 2	A 0	A 0	C 2	→
C 2	D 3	E 4	F 5	ENT				
GX OUT	→	D 3	ENT					

В следующем примере, когда значение в ячейке памяти V2000 = 2550 и в ячейке V2002 не равно 2500, Y3 сработает.

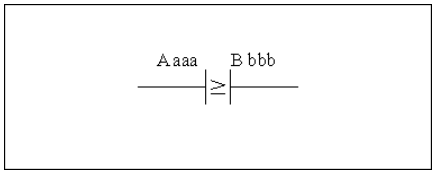


Набор на ручном программаторе

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
C 2	F 5	F 5	A 0	ENT				
W ANDN	SHFT	E 4	→	C 2	A 0	A 0	C 2	→
C 2	F 5	A 0	A 0	ENT				
GX OUT	→	D 3	ENT					

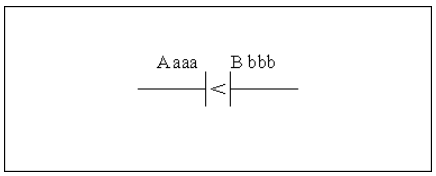
Store (STR)

Команда Comparative Store начинает новую цепь или дополнительную ветвь в цепи с нормально открытым контактом сравнения. Контакт будет включен, когда Aaaa >= Bbbb.



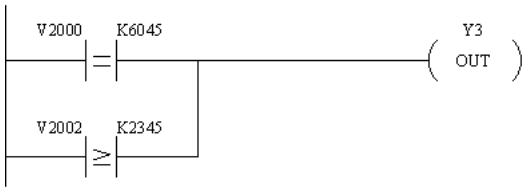
Store Not (STRN)

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		DL06 Диапазон	
	A/B	aaa	bbb
V-память	V	Смотри карту памяти	Смотри карту памяти
Указатель	P	Смотри карту памяти	Смотри карту памяти
Константа	K	—	0–9999
Таймер	T	0-377	
Счетчик	CT	0-177	

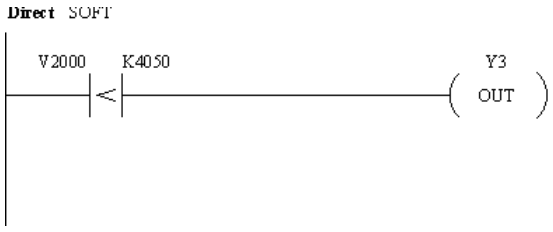
В следующем примере, когда значение в ячейке памяти V2000 >=1000, Y3 сработает.



Набор на ручном программаторе

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
G 6	A 0	E 4	F 5	ENT				
Q OR	→	SHFT	V AND	C 2	A 0	A 0	C 2	→
C 2	D 3	E 4	F 5	ENT				
GX OUT	→	D 3	ENT					

В следующем примере, когда значение в ячейке памяти V2000 < 4050, Y3 сработает.

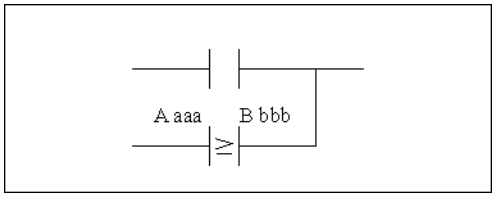


Набор на ручном программаторе

SP STRN	→	SHFT	V AND	C 2	A 0	A 0	A 0
→	E 4	A 0	F 5	A 0	ENT		
GX OUT	→	D 3	ENT				

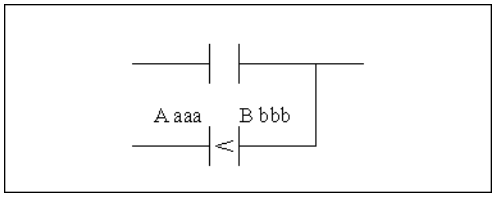
Or (OR)

Команда Comparative Or соединяет нормально открытый контакт сравнения параллельно с другим контактом. Контакт будет включен, когда Aaaa >= Bbbb.



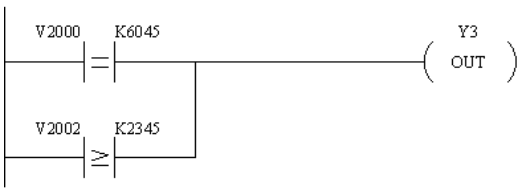
Or Not (ORN)

Команда Comparative Or Not соединяет нормально открытый контакт сравнения параллельно с другим контактом. Контакт будет включен, когда Aaaa < Bbbb.



Тип данных операнда		DL06 Диапазон	
	A/B	aaa	bbb
V-память	V	Смотри карту памяти	Смотри карту памяти
Указатель	P	Смотри карту памяти	Смотри карту памяти
Константа	K	—	0–9999
Таймер	T	0-377	
Счетчик	CT	0-177	

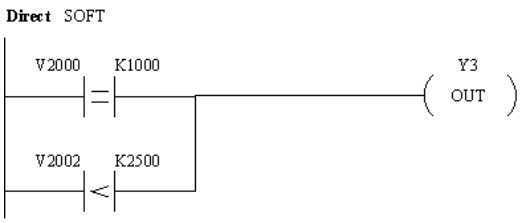
В следующем примере, когда значение в ячейке памяти V2000 = 6045 или значение в ячейке V2002 >= 2345, Y3 сработает.



Набор на ручном программаторе

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
G 6	A 0	E 4	F 5	ENT				
Q OR	→	SHFT	V AND	C 2	A 0	A 0	C 2	→
C 2	D 3	E 4	F 5	ENT				
GX OUT	→	D 3	ENT					

В следующем примере, когда значение в ячейке памяти V2000 = 1000 или значение в ячейке V2002 < 2500, Y3 сработает.

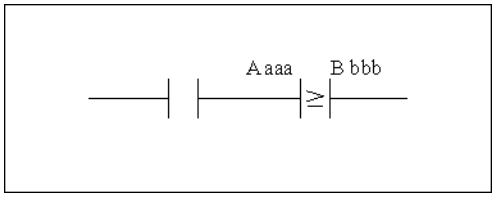


Набор на ручном программаторе

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
B 1	A 0	A 0	A 0	ENT				
R ORN	→	SHFT	V AND	C 2	A 0	A 0	C 2	→
C 2	F 5	A 0	A 0	ENT				
GX OUT	→	D 3	ENT					

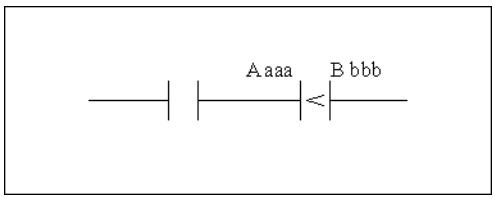
And (AND)

Команда Comparative And соединяет нормально открытый контакт сравнения последовательно с другим контактом. Контакт будет включен, когда Aaaa ≥ Bbbb.



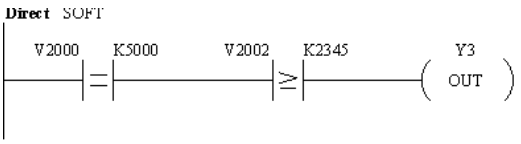
And Not (ANDN)

Команда Comparative And Not соединяет нормально открытый контакт сравнения последовательно с другим контактом. Контакт будет включен, когда Aaaa < Bbbb.



Тип данных операнда		DL06 Диапазон	
	A/B	aaa	bbb
V-память	V	Смотри карту памяти	Смотри карту памяти
Указатель	P	Смотри карту памяти	Смотри карту памяти
Константа	K	—	0–9999
Таймер	T	0-377	
Счетчик	CT	0-177	

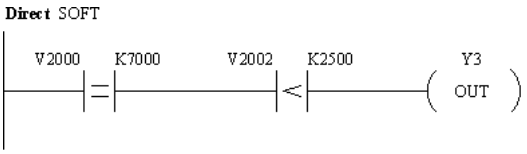
В следующем примере, когда значение в ячейке памяти V2000 = 5000 и значение в ячейке V2002 >= 2345, Y3 сработает.



Набор на ручном программаторе

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
F 5	A 0	A 0	A 0	ENT				
V AND	→	SHFT	V AND	C 2	A 0	A 0	C 2	→
C 2	D 3	E 4	F 5	ENT				
GX OUT	→	D 3	ENT					

В следующем примере, когда значение в ячейке памяти V2000 = 7000 и значение в ячейке V2002 < 2500, Y3 сработает.



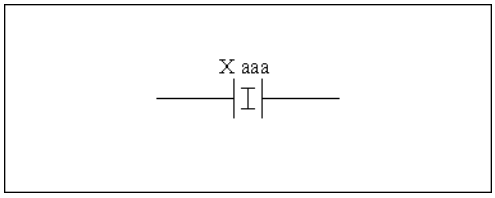
Набор на ручном программаторе

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
H 7	A 0	A 0	A 0	ENT				
W ANDN	→	SHFT	V AND	C 2	A 0	A 0	C 2	→
C 2	F 5	A 0	A 0	ENT				
GX OUT	→	SHFT	Y AND	D 3	ENT			

Немедленные команды

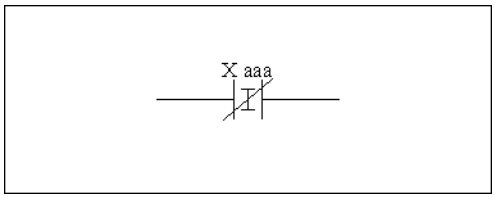
Store Immediate (STRI)

Команда Store Immediate начинает новую цепь или дополнительную ветвь в цепи. Состояние контакта будет таким же, как состояние соответствующей входной точки во время, когда команда выполняется. Регистр отбражения не изменяется.



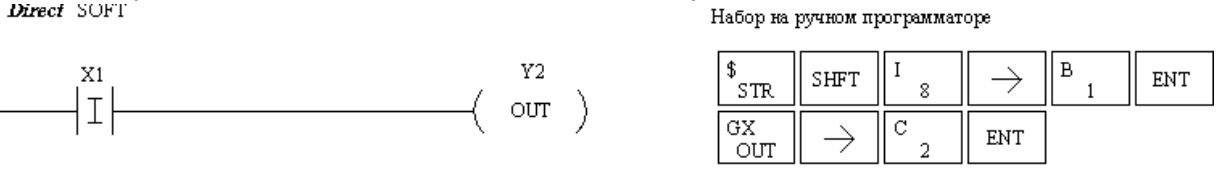
Store Not Immediate (STRNI)

Команда Store Not Immediate начинает новую цепь или дополнительную ветвь в цепи. Состояние контакта будет противоположным состоянию соответствующей входной точки во время, когда команда выполняется. Регистр отбражения не изменяется.

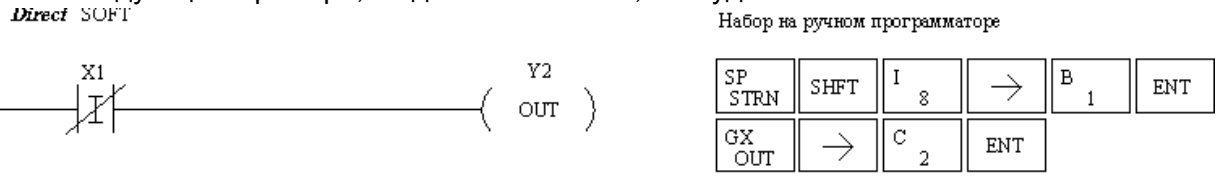


Тип данных оператора	DL06 Диапазон
	aaa
Входы X	0-777

В следующем примере, когда X1 включен, Y2 будет включен.

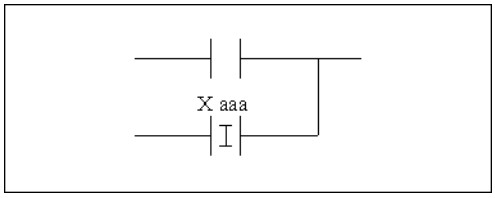


В следующем примере, когда X1 выключен, Y2 будет включен.



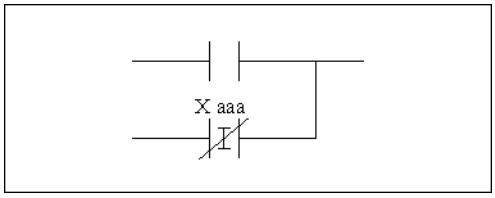
Or Immediate (ORI)

Команда Or Immediate соединяет два контакта параллельно. Состояние контакта будет таким же, как состояние соответствующей входной точки модуля во время, когда команда выполняется. Регистр отбражения не изменяется.



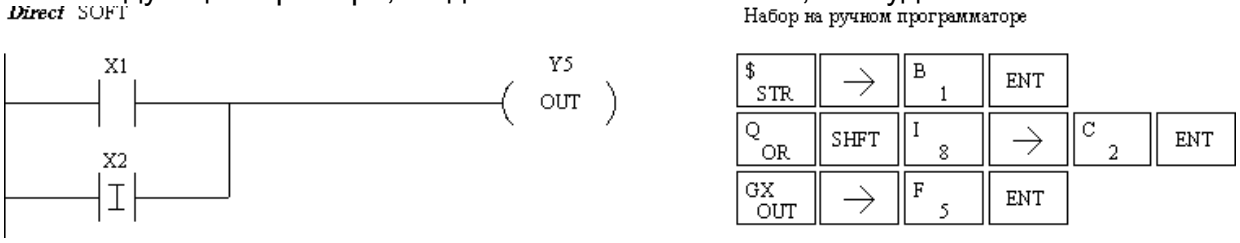
Or Not Immediate (ORNI)

Команда Or Not Immediate соединяет два контакта параллельно. Состояние контакта будет противоположным состоянию соответствующей входной точки модуля во время, когда команда выполняется. Регистр отбражения не изменяется.

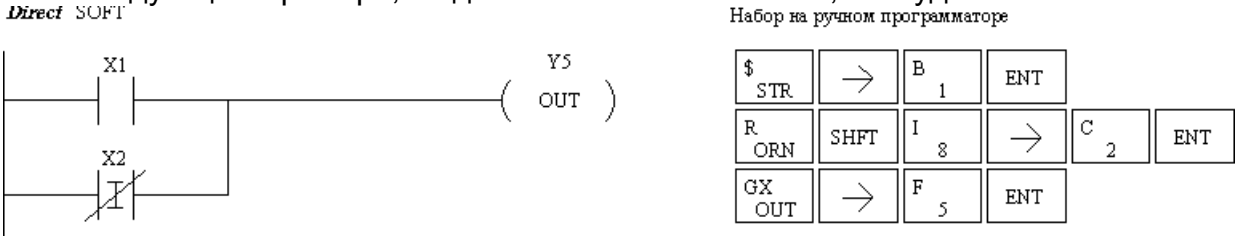


Тип данных оператора	DL06 Диапазон
	aaa
Входы X	0-777

В следующем примере, когда X1 или X2 включены, Y5 будет включен.

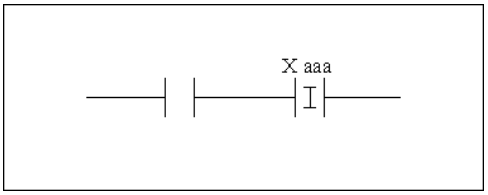


В следующем примере, когда X1 или X2 выключены, Y5 будет включен.



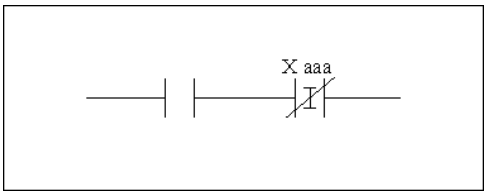
And Immediate (ANDI)

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 изменяется.



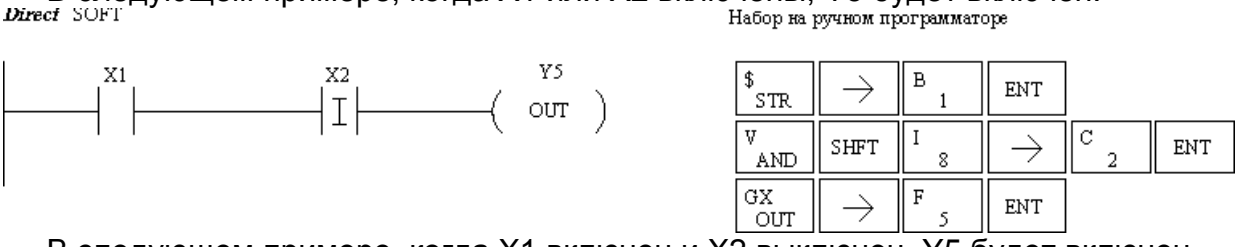
And Not Immediate (ANDNI)

Команда And Not Immediate соединяет два контакта последовательно. Состояние контакта будет противоположным состоянию соответствующей входной точки модуля во время, когда команда выполняется. Регистр отображения не изменяется

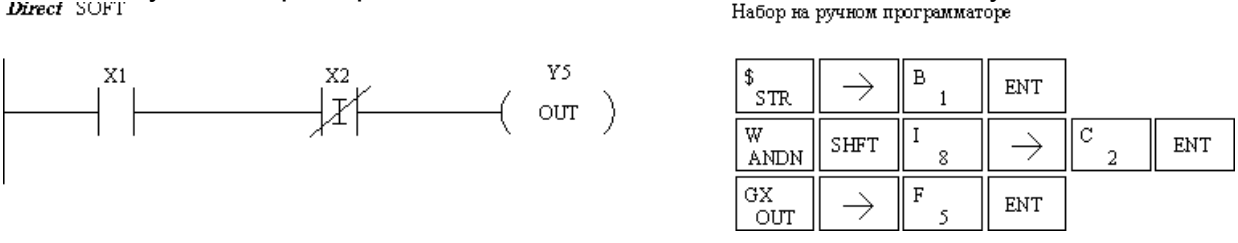


Тип данных оператора	DL06 Диапазон
	aaa
Входы X	0-777

В следующем примере, когда X1 или X2 включены, Y5 будет включен.

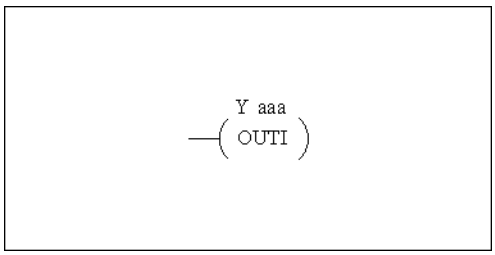


В следующем примере, когда X1 включен и X2 выключен, Y5 будет включен.



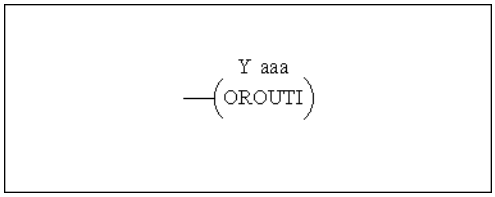
Out Immediate (OUTI)

Команда Out Immediate отражает состояние цепи (вкл/выкл) и выводит дискретное состояние в определенную канал модуля выхода и регистр отбражения 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)

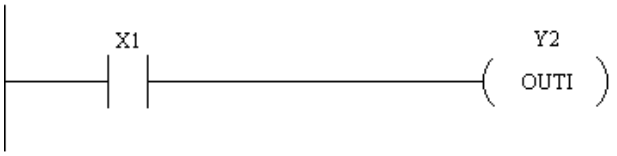
Команда Or Out Immediate была разработана, чтобы использовать несколько цепей дискретной логики для управления одним выходом. Можно многократно использовать команду Or Out Immediate, ссылающуюся на одну и ту же точку выхода, так как операция ИЛИ выполняется вместе над всеми контактами, управляющими выходом. Если цепь находится в состоянии ВКЛ. во время выполнения команды, выход будет также включен.



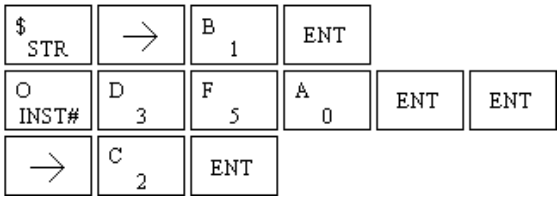
Тип данных оператора	DL06 Диапазон
	aaa
Выходы Y	0-777

В этом примере, когда X1 включен, Y2 будет включен. Для ввода команды с Ручного программатора Вы можете использовать номер команды (#350), как показано или набивать все буквы команды.

Direct SOFT

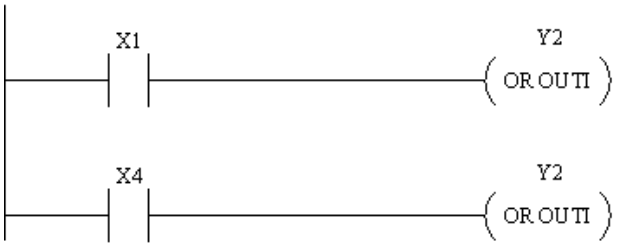


Набор на ручном программаторе

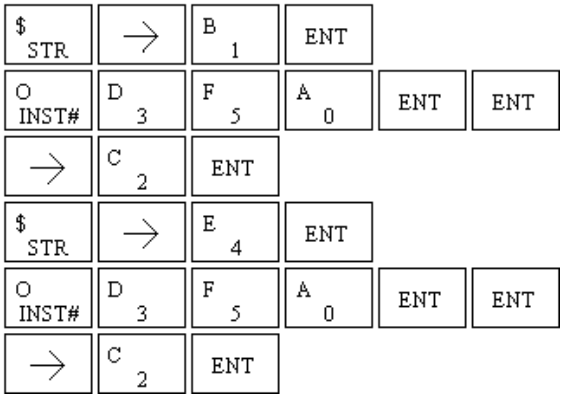


В следующем примере, когда X1 или X4 включены, Y2 будет включен.

Direct SOFT



Набор на ручном программаторе

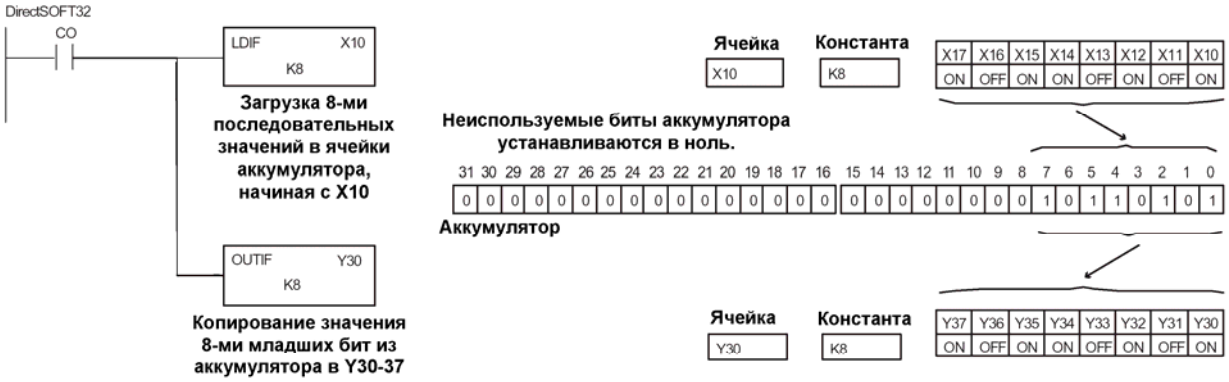


Out Immediate Formatted (OUTIF)

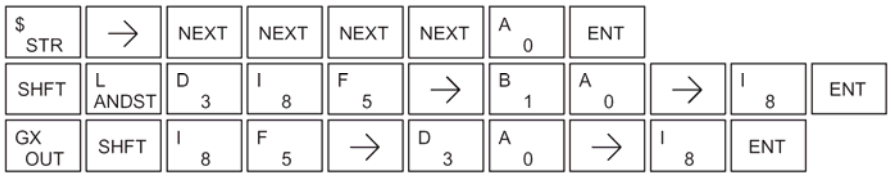
Команда Out Immediate Formatted выводит от 1 до 32 дискретных значения из аккумулятора в указанные выходные точки во время выполнения команды. Биты Аккумулятора, которые не используемые командой, устанавливаются равными нулю.

Тип данных оператора		DL06 Диапазон
		aaa
Выходы	Y	0-777
Константы	K	1-32

В следующем примере, когда C0 включен, двоичные значения для X10 -X17 загружены в аккумулятор, используя команду Load Immediate Formatted. Двоичное значение в аккумуляторе записываются в Y30-Y37, используя команду Out Immediate Formatted. Эта технология полезна когда необходимо быстро скопировать входные значения на выходы (без ожидания окончания цикла сканирования процессора).

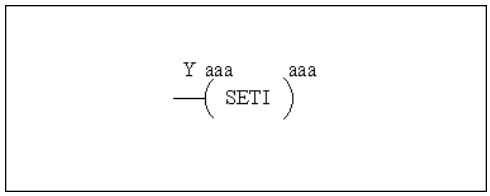


Набор на ручном программаторе



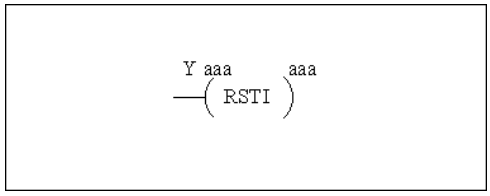
Set Immediate (SETI)

Команда Set Immediate немедленно устанавливает (включает) выход или группу выходов в регистре отображения и в соответствующем модуле (модулях) выхода во время выполнения команды. Когда выходы установлены, то необязательно входной цепи оставаться включенной. Команда Reset Immediate может использоваться для сброса выходов.



Reset Immediate (RSTI)

Команда Reset Immediate немедленно сбрасывает (выключает) выход или группу выходов в регистре отображения и в модуле (модулях) выхода во время выполнения команды. Если выходы сброшены, то входной цепи необязательно оставаться включенной.



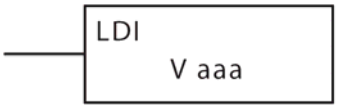
Тип данных оператора	DL06 Диапазон
	aaa
Выходы Y	0-777

В следующем примере, когда X1 включен, Y2-Y5 будут установлены в состояние ВКЛ. в регистре отображения и на соответствующем модуле (модулях) выхода.

В следующем примере, когда X1 включен, Y5-Y5 будут сброшены (выключены) в регистре отображения и на соответствующем модуле (модулях) выхода.

Load Immediate (LDI)

Команда Load Immediate загружает 16-разрядное значение ячейки V-памяти в аккумулятор. Интервал допустимых адресов включает все адреса входных точек в локальном каркасе. Значение отражает текущее состояние входных точек в момент выполнения команды. Эта команда может использоваться вместо команды LDIF, которая требует, чтобы Вы определили число входных точек.



Тип данных оператора	DL06 Диапазон
	aaa
Входы V	40400-40437

В следующем примере, когда C0 включен, двоичный код из X0-X17 будет загружен в аккумулятор, используя загрузочную команду Load Immediate. Выходные данные команды Load Immediate могут использоваться для копирования 16 бит в аккумулятор, и вывода в точки Y40-Y57. Этот метод полезен при быстром копировании входной структуры для вывода выходных точек (не ожидая завершения полного цикла сканирования ЦПУ).

Набор на ручном программаторе

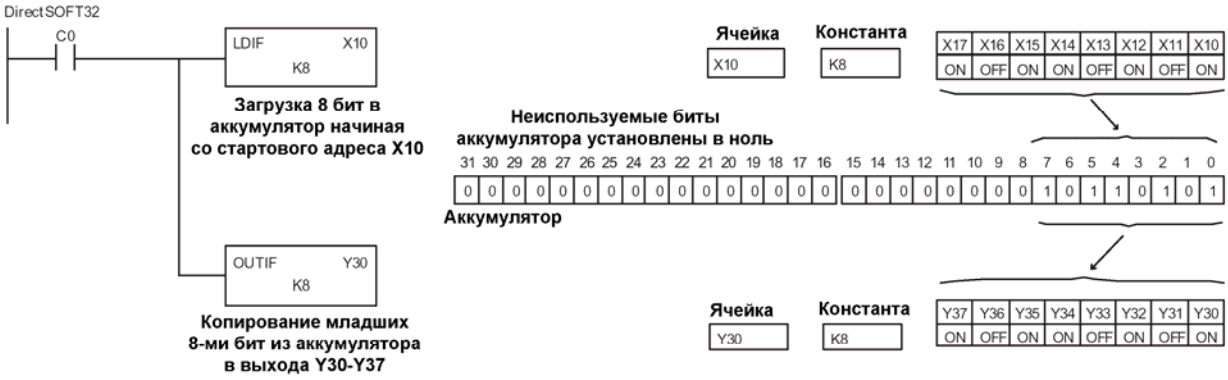
$ STR	→	NEXT	NEXT	NEXT	NEXT	A 0	ENT			
SHFT	L ANDST	D 3	I 8	→	E 4	A 0	E 4	A 0	A 0	ENT
GX OUT	SHFT	I 8	→	NEXT	E 4	A 0	F 5	A 0	C 2	ENT

Load Immediate Formatted (LDIF)

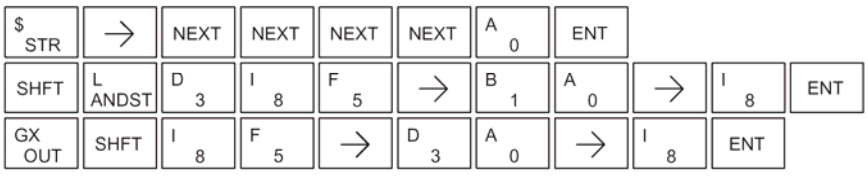
Команда Load Immediate Formatted загружает от 1 до 32 бит двоичных значений в аккумулятор. Значение отражает текущее состояние входов модуля (модулей) в момент выполнения инструкции. Неиспользуемые командой биты аккумулятора устанавливаются в ноль.

Тип данных оператора		DL06 Диапазон	
		aaa	bbb
Входы	X	0-777	--
Константы	K	--	1-32

В следующем примере, когда C0 включен, двоичный код X10-X17 будет загружен в аккумулятор, используя команду Load Immediate Formatted. Команда Out Immediate Formatted может быть использована для копирования указанного числа бит в аккумулятор, и вывода в точки Y30-Y37. Этот метод полезен при быстром копировании входной структуры для вывода выходных точек (не ожидая завершения полного цикла сканирования ЦПУ).



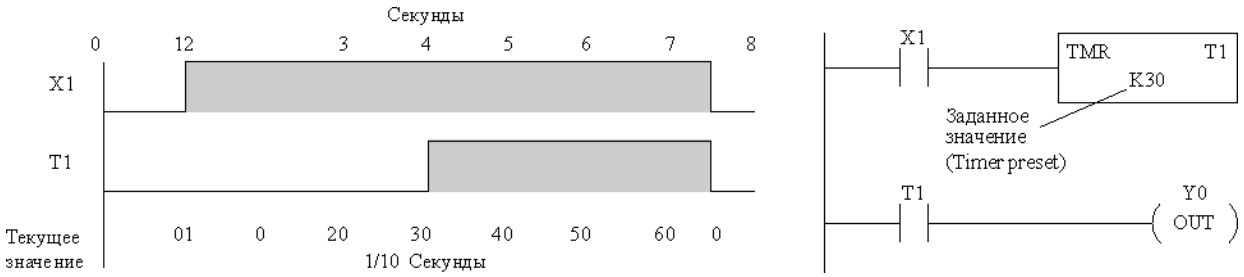
Набор на ручном программаторе



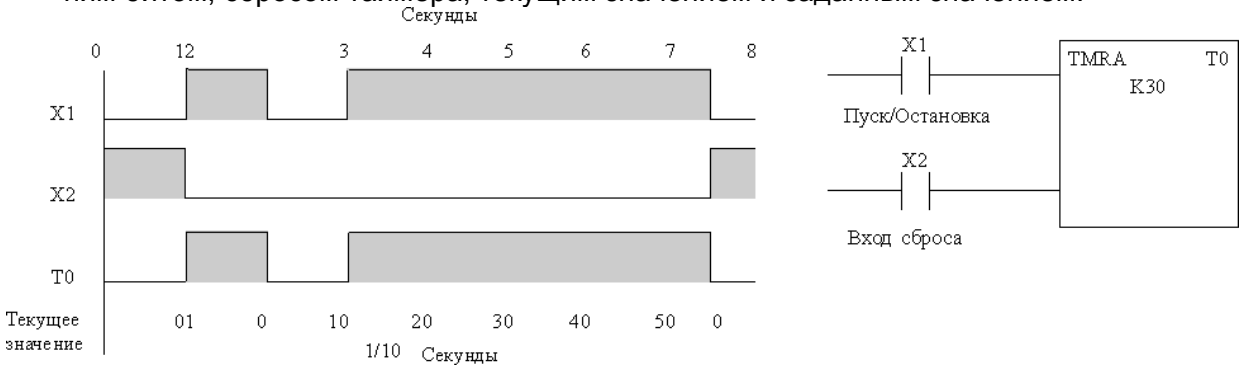
Команды таймера, счетчика и регистра сдвига

Использование таймеров

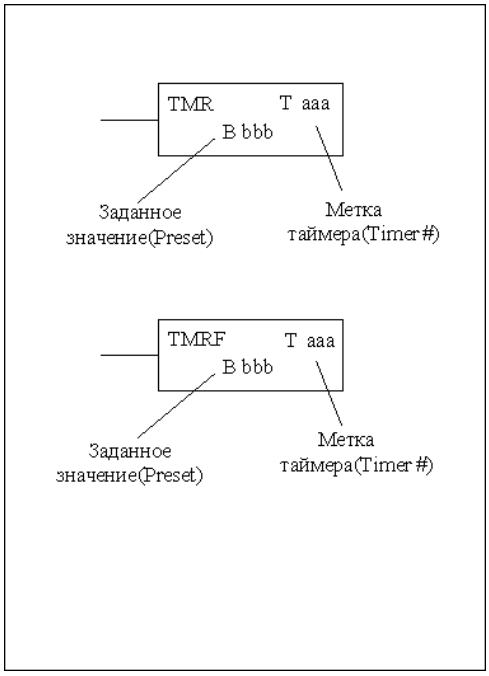
Таймеры используются для фиксации времени события и формирования временных интервалов. Одновходовой таймер будет отсчитывать время так долго, пока вход находится во включенном состоянии. Когда состояние входа изменится на противоположное, текущее значение таймера сбросится в 0. Существуют таймеры с максимальным временем 999.9 с и 99.99 с и с разрешением в десятые доли секунды и сотые доли секунды соответственно. С каждым таймером связаны биты 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 стирается и таймер при повторном запуске начнет считать с нуля. Существуют таймеры с максимальным временем 9999999.9 и 999999.99 секунд и с разрешением в десятые доли секунды и сотые доли секунды соответственно. Временная диаграмма ниже показывает связь между входом таймера, связанным с ним битом, сбросом таймера, текущим значением и заданным значением.



Timer (TMR) и Timer Fast (TMRF)



Команда Timer — это 0.1с таймер с одним входом, который считает максимально до 999.9с. Команда Timer Fast — это 0.01с 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.).

Спецификации команды

Метка таймера (Taaa): определяет номер таймера.

Заданное значение (Bbbb): константа (К) или ячейка V-памяти.

Текущее значение: Текущее значение таймера доступно при обращении к соответствующим ячейкам V- или Т-памяти*. Например, текущее значение таймера для Т3 физически находится в ячейке V-памяти V3.

Бит состояния: бит состояния доступен при обращении к соответствующей ячейке Т-памяти. Он будет ВКЛ. если текущая величина равна или больше заданного значения. Например, бит состояния для таймера 2 был бы Т2.

ПРИМЕЧАНИЕ: Предустановленное значение таймера - константа (К) может быть изменено с Ручного программатора даже в Рабочем режиме ПЛК. Поэтому Задания в V-памяти необходимы, если нужно изменять значения при помощи программы релейной логики.

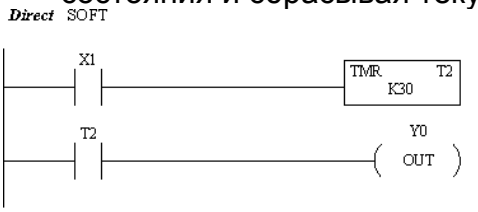
Тип данных операнда		Диапазон DL06	
	A/B	aaa	bbb
Таймеры	T	0-777	--
V-память для заданных значений	V	--	400-677 1200-7377 7400-7577 10000-17777
Указатели (только заданные значения)	P	--	400-677 1200-7377 7400-7577 10000-17777
Константы (только заданные)	K	--	0-9999
Дискретные биты состояния таймера	T/V	0-377 или V41100-41107	
Текущие значения таймера	V/T*	0-377	

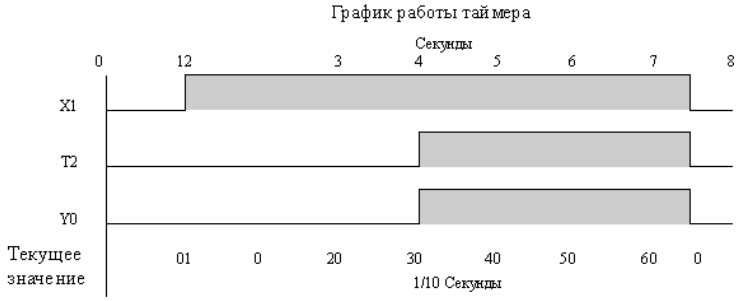
ПРИМЕЧАНИЕ. * При работе с Ручным программатором Бит состояния таймера и Текущее значение таймера используют одни и те же обозначения. При работе с Direct SOFT используются различные обозначения: Т2 для Бита состояния таймера 2 и ТА2 для Текущего значения таймера 2 .

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 в различные временные интервалы, используя один таймер. Следующие примеры показывают методы использования таймеров.

Пример таймера, использующего биты состояния

В следующем примере таймер с одни входом используется с заданным значением 3 с. Бит состояния таймера (T2) включится, когда таймер отсчитает 3 с. Таймер сбрасывается, когда X1 выключится, выключая бит состояния и сбрасывая текущее значение таймера в 0.



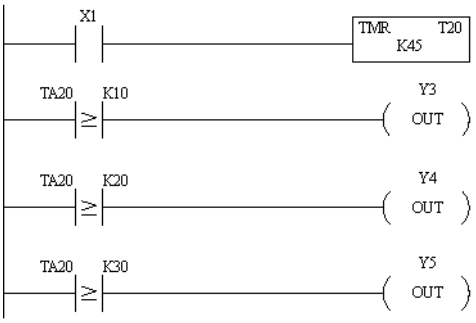


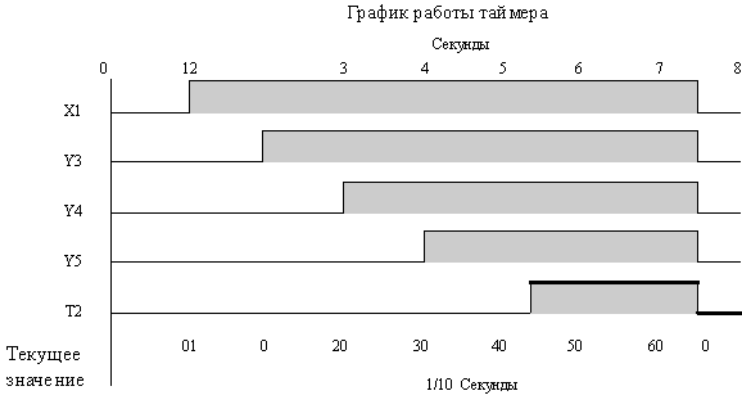
Набор на ручном программаторе

$ STR	→	B 1	ENT				
N TMR	→	C 2	→	D 3	A 0	ENT	
$ STR	→	SHFT	T MLR	C 2	ENT		
GX OUT	→	A 0	ENT				

Пример таймера, использующего контакты сравнения

В следующем примере таймер с одним входом используется с заданным значением 4.5 с. Контакты сравнения используются для включения через Y3, Y4 и Y5 в интервале 1с соответственно. Когда X1 выключится, таймер будет сброшен в 0 и контакты сравнения выключат Y3, Y4 и Y5.

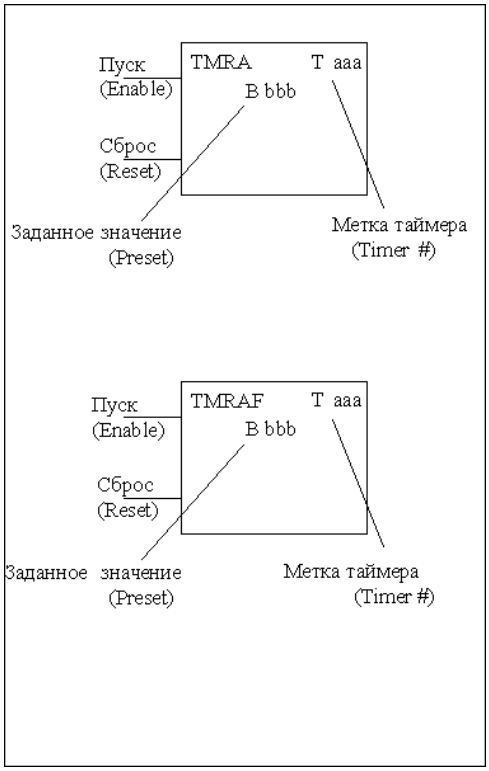




Набор на ручном программаторе

$ STR	→	B 1	ENT						
N TMR	→	C 2	A 0	→	E 4	F 5	ENT		
$ STR	→	SHFT	T MLR	C 2	A 0	→	B 1	A 0	ENT
GX OUT	→	D 3	ENT						
$ STR	→	SHFT	T MLR	C 2	A 0	→	C 2	A 0	ENT
GX OUT	→	E 4	ENT						
$ STR	→	SHFT	T MLR	C 2	A 0	→	D 3	A 0	ENT
GX OUT	→	F 5	ENT						

Accumulating Timer (TMRA)



Accumulating Timer — это 0.1 секундный таймер с двумя входами, который считает до 9999999.9с максимум. *TMRA использует два регистра в V-памяти.*

Accumulating Fast Time(TMRAF)

Команда **Accumulating Fast Timer** — это быстрый (0.01с) таймер с двумя входами, который считает до 99999.99с максимум.
TMRAF использует два регистра в V-памяти.
*Каждый таймер использует два регистра в V-памяти.*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.

Спецификации команды

Метка таймера (Taaa): определяет номер таймера.

Заданное значение (Bbbb): константа (К) или V-память.

Текущее значение: текущие значения таймера доступны при обращении к соответствующим ячейкам V- или T-памяти (см. примечание).Например, текущее значение таймера для T3 физически находится в V-памяти, V3

Бит состояния: бит состояния доступен при обращении к соответствующей ячейке T-памяти. Он будет включен если текущая величина равна или больше заданного значения. Например, бит состояния для таймера 2 - T2.

ПРИМЕЧАНИЕ. 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 accumulating таймер, то следующий доступный таймер будет T3.

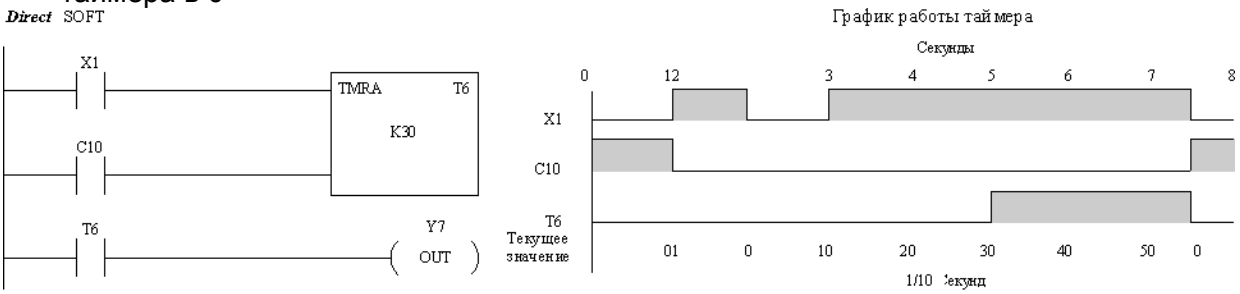
Тип данных операнда	A/B	Диапазон DL06 aaa	Диапазон DL06 bbb
Таймеры	T	0-376	--
V-память для заданных значений	V	--	400-677 1200-7377 7400-7577 10000-17777
Указатели (только заданные значения)	P	--	400-677 1200-7377 7400-7577 10000-17777
Константы (только заданные)	K	--	0- 99999999
Дискретные биты состояния таймера	T/V	0-376 или V41100-41117	
Текущие значения таймера	V/T*	0-376	

ПРИМЕЧАНИЕ. * При работе с Ручным программатором Бит состояния таймера и Текущее значение таймера используют одни и те же обозначения. При работе с Direct SOFT используются различные обозначения: T2 для Бита состояния таймера 2 и TA2 для Текущего значения таймера 2 .

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 в различные временные интервалы, используя один таймер. Следующие примеры показывают каждый метод использования таймеров.

Пример аккумулирующего таймера, с использованием бита состояния

В следующем примере, таймер с двумя входами (Accumulating Timer) используется с заданным значением 3 с. Бит состояния таймера (T6) включится, когда таймер отсчитает 3 с. Обратите внимание, в этом примере таймер работает 1 с, останавливается на 1 с, затем продолжает счет. Таймер будет сброшен, когда включится C10, выключив тем самым Бит состояния и сбросив текущее значение таймера в 0



Набор на ручном программаторе

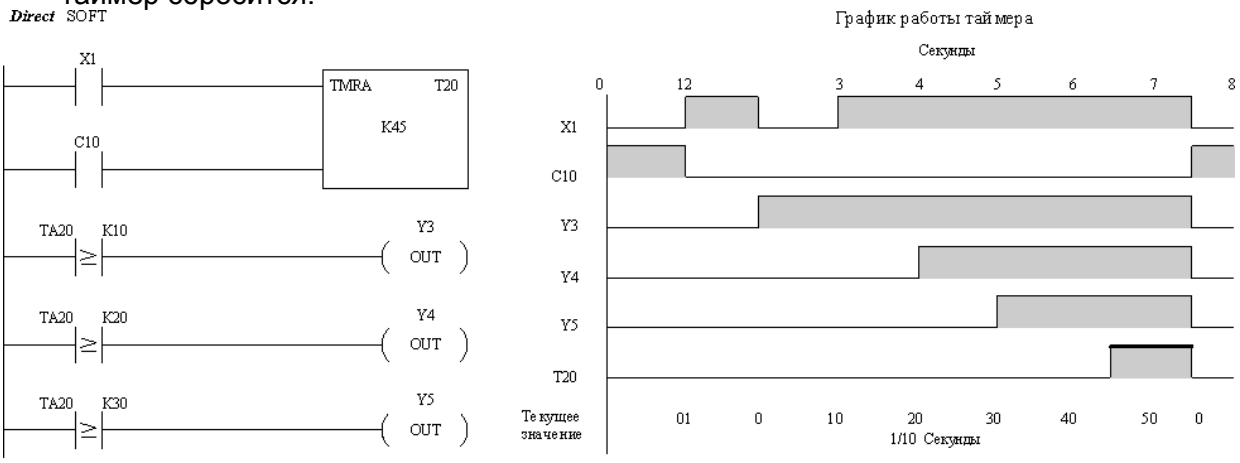
$ STR	→	B 1	ENT			
$ STR	→	SHFT	C 2	B 1	A 0	ENT
N TMR	SHFT	A 0	→	G 6	→	

Набор на ручном программаторе

D 3	A 0	ENT				
$ STR	→	SHFT	T MLR	G 6	ENT	
GX OUT	→	B 1	A 0	ENT		

Пример аккумулирующего таймера, с использованием контактов сравнения

В следующем примере, таймер с двумя входами используется с заданным значением 4.5 с. Контакты сравнения используются, для того, чтобы включить Y3, Y4 и Y5 с интервалом 1 с соответственно. Контакты сравнения выключатся, когда таймер сбросится.



Набор на ручном программаторе

$ STR	→	B 1	ENT						
$ STR	→	SHFT	C 2	B 1	A 0	ENT			
N TMR	SHFT	A 0	→	C 2	A 0	→	E 4	F 5	ENT
$ STR	→	SHFT	T MLR	C 2	A 0	→	B 1	A 0	ENT
GX OUT	→	D 3	ENT						
$ STR	→	SHFT	T MLR	C 2	A 0	→	C 2	A 0	ENT

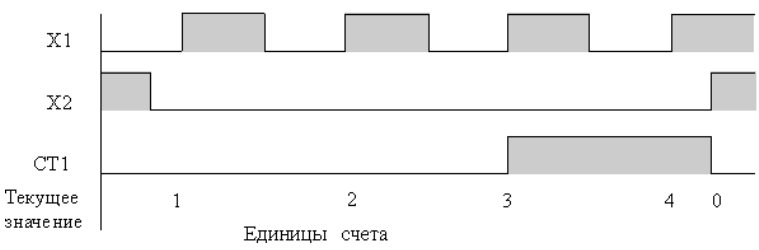
Набор на ручном программаторе

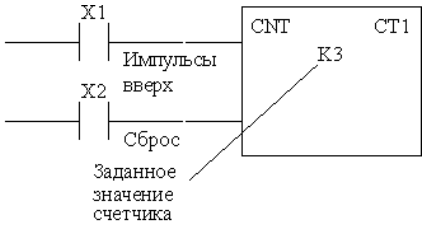
GX OUT	→	E 4	ENT			
$ STR	→	SHFT	T MLR	C 2	A 0	→
D 3	A 0	ENT				
GX OUT	→	F 5	ENT			

Использование счетчиков

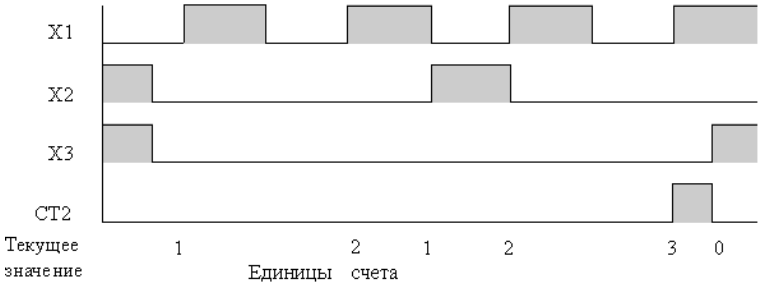
Счетчики используются для счета числа событий. Существуют просто счетчики (не реверсивные), реверсивные счетчики (Up/down) и счетчики стадий в RLL^{PLUS}.

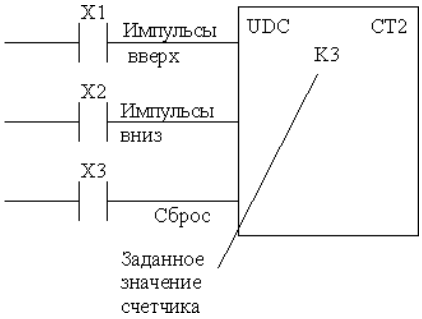
Счетчик нереверсивный имеет два входа: счетный вход и вход сброса. Максимальное значение 9999. Графики внизу показывают соотношения между счетным входом, входом сброса , битом состояния, текущим значением и уставкой счетчика.



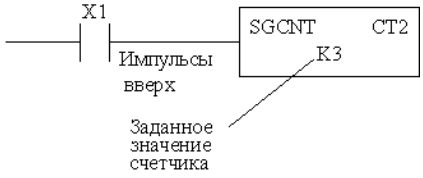


Реверсивный Счетчик имеет три входа: счетный вход — вверх (Up), счетный вход-вниз(Down) и вход сброса. Максимальное значение 99999999. Графики внизу показывают соотношения между счетными входами, входом сброса , битом состояния, текущим значением и уставкой счетчика.

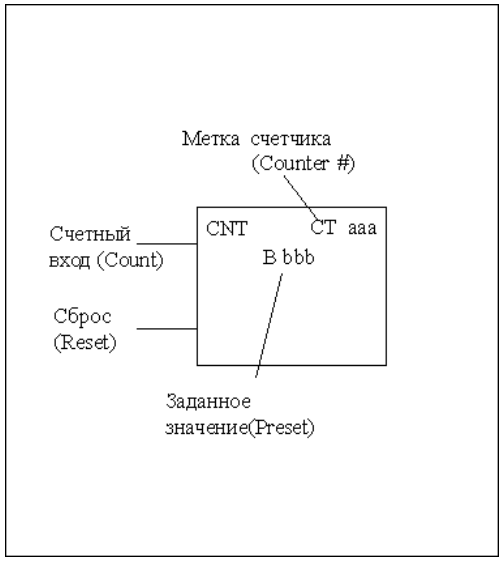




Счетчик стадий имеет два входа: счетный вход и вход сброса командой RST (RLL^{PLUS}). Максимальное значение 9999. Графики внизу показывают соотношения между счетным входом, входом сброса , битом состояния, текущим значением, уставкой счетчика и командой сброса -RST.



Counter (CNT)



Counter — это счетчик с двумя входами, который увеличивается, когда вход Count переходит из состояния ВЫКЛ. в состояние ВКЛ. Когда включается вход 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.

Спецификация команды:

Метка счетчика (CTaaa): указывает номер счетчика.

Заданное значение (Bbbb): константа (К) или ячейка V-памяти.

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.

Бит состояния: бит состояния доступен при обращении к соответствующей ячейке CT-памяти. Он будет включен если текущая величина равна или больше заданного значения. Например, Бит состояния для счетчика 2 был бы CT2.

ПРИМЕЧАНИЕ. Значение константы (К) может быть изменено с Ручного программатора даже в Рабочем режиме ПЛК. Поэтому Задания в V-памяти необходимы, если нужно изменять значения средствами программы релейной логики.

Тип данных операнда		Диапазон DL06	
	A/B	**aaa**	**bbb**
Счетчики	CT	0-177	--
V-память (только для заданных значений)	V	--	400-677 1200-7377 7400-7577 10000-17777
Указатели (только для заданных значений)	P	--	400-677 1200-7377 7400-7577 10000-17777
Константы (только для заданных значений)	K	--	0-9999
Дискретные биты состояния счетчика	CT/V	0-177 или V41140-41147	
Текущие значения счетчика	V/CT*	1000-1177	

ПРИМЕЧАНИЕ. * При работе с Ручным программатором Бит состояния счетчика и Текущее значение счетчика используют одни и те же обозначения. При работе с Direct SOFT используются различные обозначения: CT2 для Бита состояния счетчика 2 и CTA2 для Текущего значения счетчика 2 .

Пример счетчика, с использованием бит 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 Reset C10 включится, бит состояния счетчика выключится и текущее значение будет 0. Текущее значение для счетчика CT2 будет храниться в ячейке V-памяти V1002.

Direct SOFT

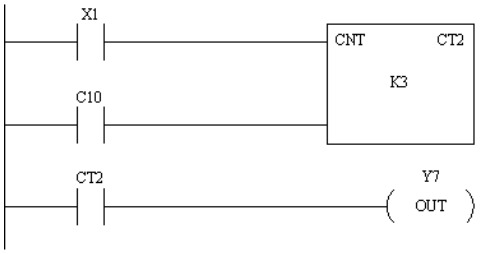
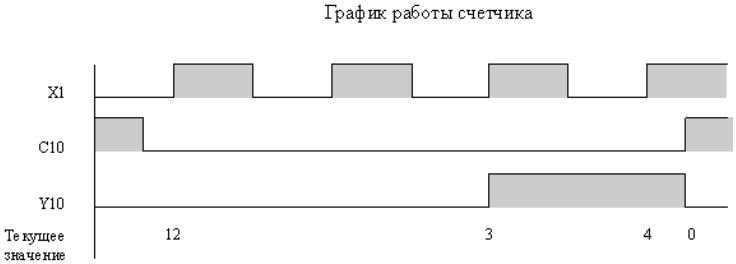
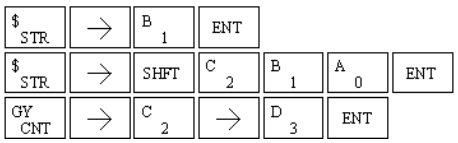


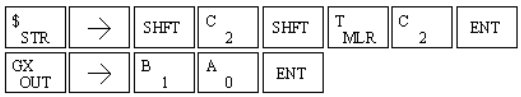
График работы счетчика



Набор на ручном программаторе



Набор на ручном программаторе



Пример счетчика, с использованием контактов 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 в разные отсчеты времени. Когда вход Reset C10 включится, бит состояния счетчика выключится, текущее значение будет 0, и контакты сравнения выключатся.

Direct SOFT

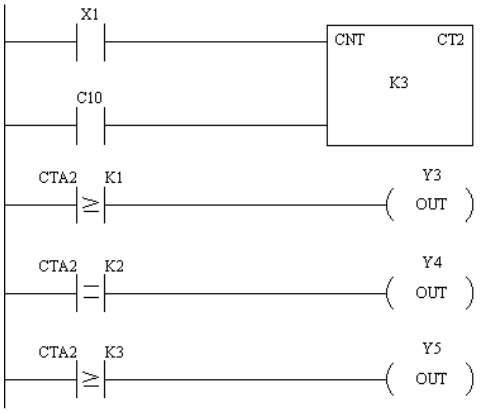
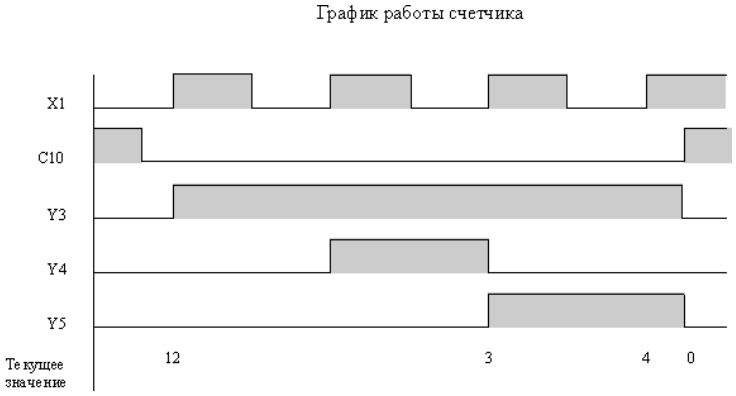
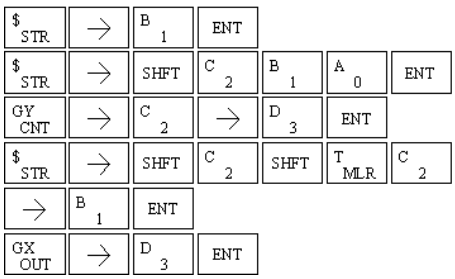


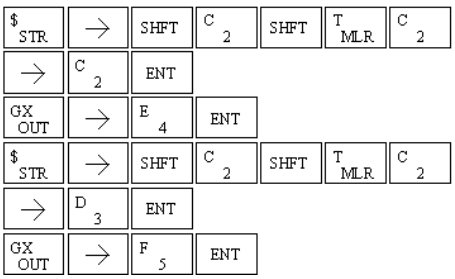
График работы счетчика



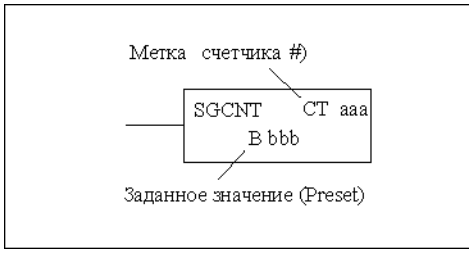
Набор на ручном программаторе



Набор на ручном программаторе



Stage Counter (SGCNT)



Stage Counter – это счетчик с одним входом, который увеличивается, когда на входе происходит переход из состояния ВЫКЛ. в состояние ВКЛ. Этот счетчик отличается от других счетчиков, так как он будет хранить текущее значение до тех пор, пока не будет сброшен командой RST(reset). 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 сбросится.

Спецификации команды

Метка счетчика (CTaaa): указывает номер счетчика.

Заданное значение (Bbbb): константа (K) или ячейка V-памяти. (Пойнтер (P) для DL240 или DL250).

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.

Бит состояния: бит состояния доступен при обращении к соответствующей ячейке CT-памяти. Он будет включен, если текущая величина равна или больше заданного значения. Например, бит состояния для счетчика 2 был бы CT2.

Тип данных операнда		Диапазон DL06	
	A/B	**aaa**	**bbb**
Счетчики	CT	0-177	--
V-память (только для заданных значений)	V	--	400-677 1200-7377 7400-7577 10000-17777
Указатели (только для заданных значений)	P	--	400-677 1200-7377 7400-7577 10000-17777
Константы (только для заданных значений)	K	--	0-9999
Дискретные биты состояния счетчика	CT/V	0-177 или V41140-41147	
Текущие значения счетчика	V/CT*	1000-1177	

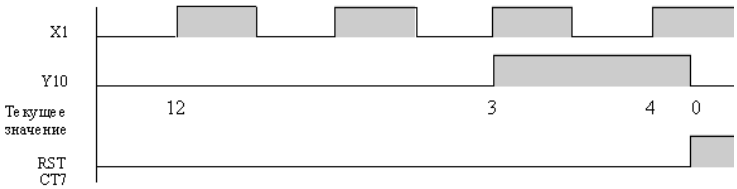
ПРИМЕЧАНИЕ. * При работе с Ручным программатором Бит состояния счетчика и Текущее значение счетчика используют одни и те же обозначения. При работе с Direct SOFT используются различные обозначения: CT2 для Бита состояния счетчика 2 и CTA2 для Текущего значения счетчика 2.

Пример Stage Counter, с использованием бит состояния

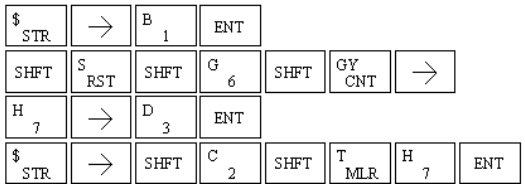
В следующем примере, когда X1 переходит из положения ВЫКЛ. в положение ВКЛ., счетчик CT7 увеличится на 1. Когда текущее значение достигнет 3, бит состояния счетчика CT7 включится и включит Y10. Бит состояния счетчика CT7 будет оставаться включенным, пока счетчик не будет сброшен командой RST. Когда счетчик сброшен, бит состояния счетчика выключится и текущее значение счетчика будет равно 0. Текущее значение для счетчика CT7 будет храниться в ячейке V-памяти V1007.

Direct SOFT

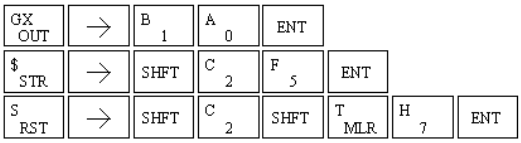
График работы счетчика



Набор на ручном программаторе



Набор на ручном программаторе



Пример Stage Counter, с использованием контактов 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 в разные отсчеты счетчика. Хотя это и не показано в примере, когда счетчик сбрасывается командой Reset, бит состояния счетчика выключится и текущее значение будет 0. Текущее значение для счетчика CT2 будет храниться в ячейке V-памяти V1002.

Direct SOFT

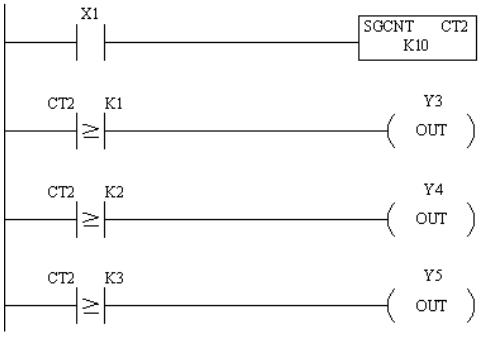
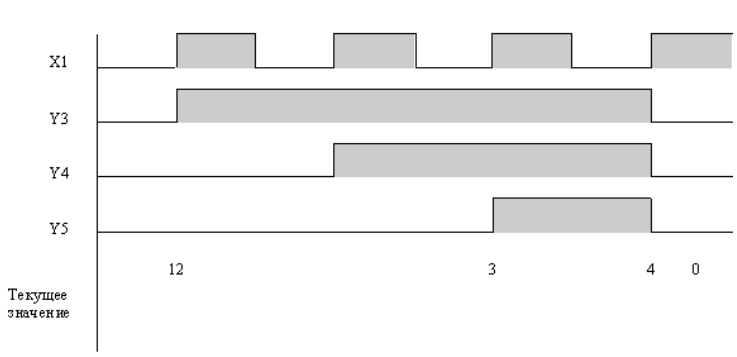
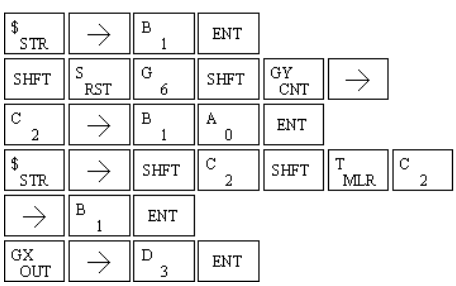


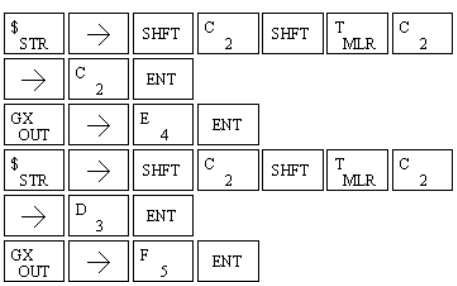
График работы счетчика



Набор на ручном программаторе

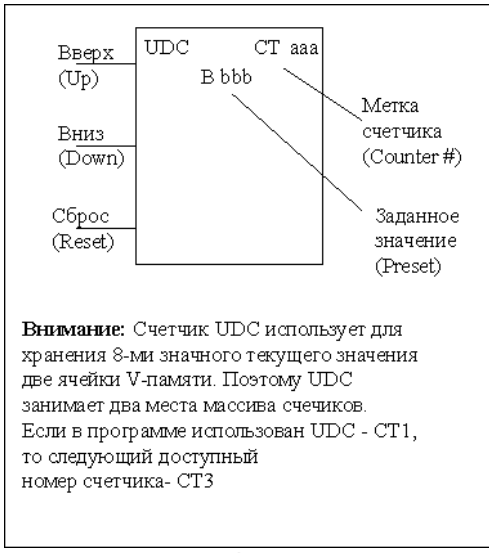


Набор на ручном программаторе



Up Down Counter (UDC)

Счетчик Up/Down Counter (Реверсивный счетчик) увеличивается каждый раз при переходе входа Up (Вверх) из состояния ВЫКЛ. в состояние ВКЛ. и уменьшается каждый раз при переходе входа Down (Вниз) из состояния ВЫКЛ. в состояние ВКЛ. Счетчик сбрасывается в 0, когда вход Reset (Сброс) включается. Диапазон счета: 0-99999999. Неиспользуемый счетный вход должен быть выключен, чтобы функционировал активный счетный вход.



Дискретный бит состояния счетчика и текущее значение не определены в команде счетчика

Спецификации команды:

Метка счетчика (CTaaa): указывает номер счетчика.

Заданное значение (Bbbb): константа (K) или ячейка V-памяти. (Пойнтер-указатели (P) для DL240 или DL250).

Текущее значение: текущий счет – значение двойного слова, доступного при обращении к соответствующим ячейкам V- или CT-памяти*. Ячейка V-памяти – ячейка счетчика + 1000. Например, текущее значение счетчика для CT5 находится в ячейках V-памяти V1005 и V1006.

Бит состояния: бит состояния доступен при обращении к соответствующей ячейке CT-памяти. Он будет включен, если текущая величина равна или больше заданного значения. Например, бит состояния для счетчика 2 был бы CT2.

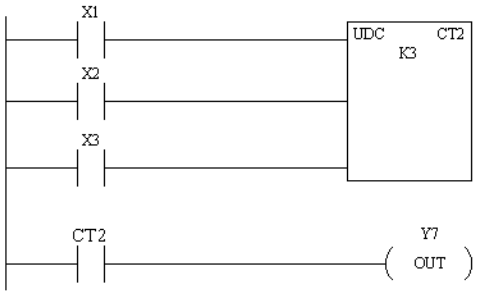
Тип данных операнда		Диапазон DL06	
	A/B	**aaa**	**bbb**
Счетчики	CT	0-176	--
V-память (только для заданных значений)	V	--	400-677 1200-7377 7400-7577 10000-17777
Указатели (только для заданных значений)	P	--	400-677 1200-7377 7400-7577 10000-17777
Константы (только для заданных значений)	K	--	0-99999999
Дискретные биты состояния счетчика	CT/V	0-176 или V41140-41147	
Текущие значения счетчика	V/CT*	1000-1176	

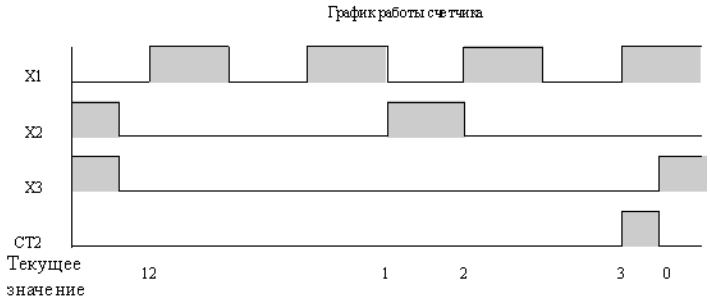
ПРИМЕЧАНИЕ. * При работе с Ручным программатором Бит состояния счетчика и Текущее значение счетчика используют одни и те же обозначения. При работе с Direct SOFT используются различные обозначения: CT2 для Бита состояния счетчика 2 и CTA2 для Текущего значения счетчика 2 .

Пример Up/Down счетчика, с использованием бита состояния

В следующем примере, если X2 и X3 выключены, когда X1 переходит из положения ВЫКЛ. в положение ВКЛ., счетчик увеличится на 1. Если X1 и X3 выключены, счетчик уменьшится на 1, когда X2 переходит из положения ВЫКЛ. в положение ВКЛ. Когда текущее значение достигнет заданного значения 3, бит состояния счетчика включится. Когда X3 включится, бит состояния счетчика выключится и текущее значение счетчика будет 0.

Direct SOFT





Набор на ручном программаторе

$ STR	→	B 1	ENT		
$ STR	→	C 2	ENT		
$ STR	→	D 3	ENT		
SHFT	U ISG	D 3	C 2	→	C 2

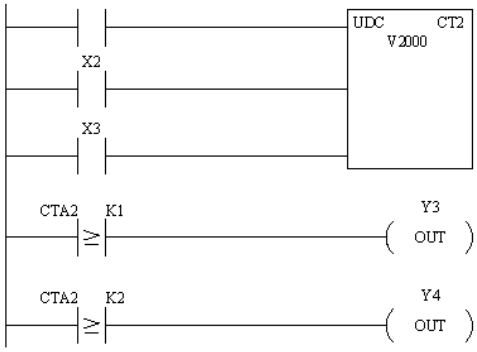
Набор на ручном программаторе

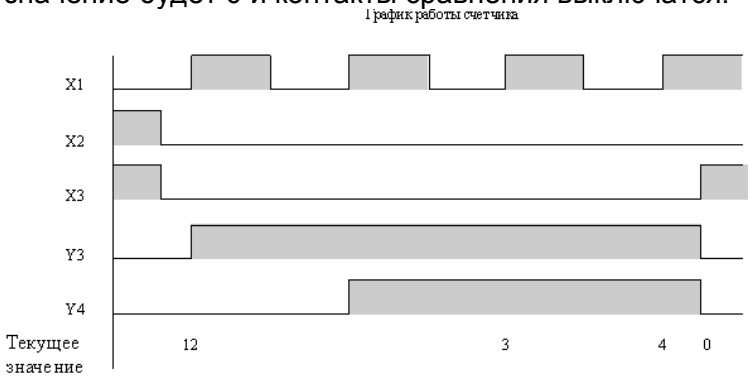
→	D 3	ENT					
$ STR	→	SHFT	C 2	SHFT	T MLR	C 2	ENT
GX OUT	→	B 1	A 0	ENT			

Пример Up/Down счетчика, с использованием контактов сравнения

В следующем примере, когда X1 переходит из положения ВЫКЛ. в положение ВКЛ., счетчик CT2 увеличится на 1. Контакты сравнения используются, чтобы включить Y3 и Y4 в разные отсчеты времени. Когда вход Reset X3 включится, бит состояния счетчика выключится, текущее значение будет 0 и контакты сравнения выключатся.

Direct SOFT





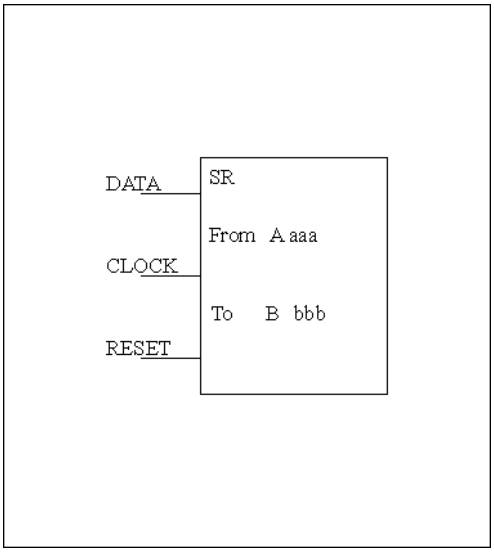
Набор на ручном программаторе

$ STR	→	B 1	ENT				
$ STR	→	C 2	ENT				
$ STR	→	D 3	ENT				
SHFT	U ISG	D 3	C 2	→	C 2	→	
SHFT	V AND	C 2	A 0	A 0	A 0	ENT	
$ STR	→	SHFT	C 2	SHFT	T MLR	C 2	

Набор на ручном программаторе

→	B 1	ENT					
GX OUT	→	D 3	ENT				
$ STR	→	SHFT	C 2	SHFT	T MLR	C 2	
→	C 2	ENT					
GX OUT	→	E 4	ENT				

Shift Register (SR)



Команда Shift Register (сдвиговый регистр) сдвигает данные на определенное число реле управления (C). Рабочий диапазон сдвигового регистра должен начинаться в начале 8-разрядной границы и использовать 8-битовые блоки.
Сдвиговый регистр имеет три контакта.

- **Data** – Определяет значение (1 или 0), которое введет регистр.
- **Clock** – Сдвигает биты по одной позиции при каждом переходе от низкого сигнала к высокому.
- **Reset** – Сбрасывает сдвиговый регистр в ноль.

При каждом переходе входа Clock из состояния ВЫКЛ. в состояние ВКЛ., биты сдвигового регистра перемещаются на один разряд и состояние входа Data помещается в начальную позицию сдвигового регистра. Направление сдвига зависит от ввода в поля From и To. From = C0 и To = C17 определили бы блок из шестнадцати битов, которые будут сдвинуты слева направо. From = C17 и To = C0 определили бы блок из шестнадцати битов, которые будут сдвинуты справа налево. Максимальный размер блока сдвигового регистра зависит от количества доступных реле управления. Минимальный размер блока – 8 реле управления (C).

Тип данных оператора	DL06 Диапазон	
A/B	aaa	bbb
Входы C	0-1777	0-1777

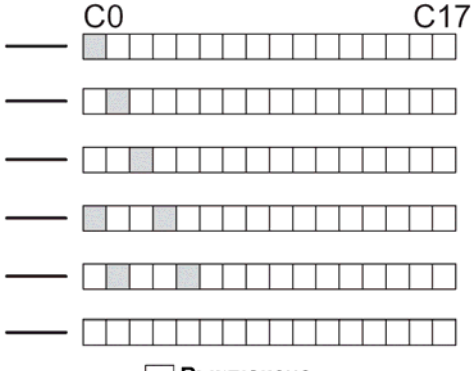
Direct SOFT32

Входы на последовательных скан-циклах

Набор на ручном программаторе

Data	Clock	Reset
1	0-1-0	0
0	0-1-0	0
0	0-1-0	0
1	0-1-0	0
0	0-1-0	0
0	0	1

Биты сдвигаемого регистра



Включено Выключено

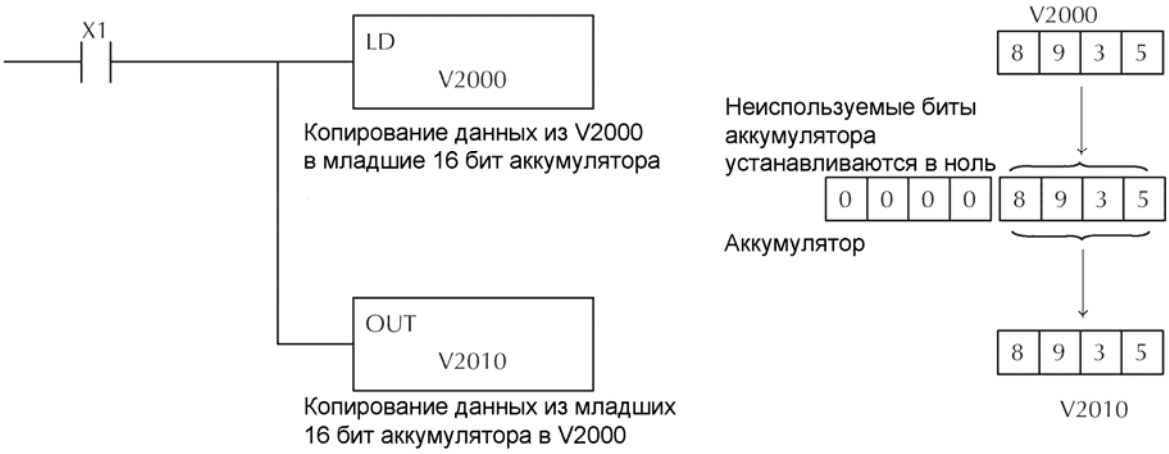
Команды загрузки и вывода данных для аккумулятора/стека

Использование аккумулятора

Аккумулятор в ЦПУ серии DL06 – это 32 разрядных регистр, который используется как временная ячейка памяти для хранения данных, которые копируется или управляется несколькими методами. 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 BCD. Аккумулятор сбрасывается в 0 в конце каждого цикла сканирования ЦПУ.

Копирование данных в аккумулятор

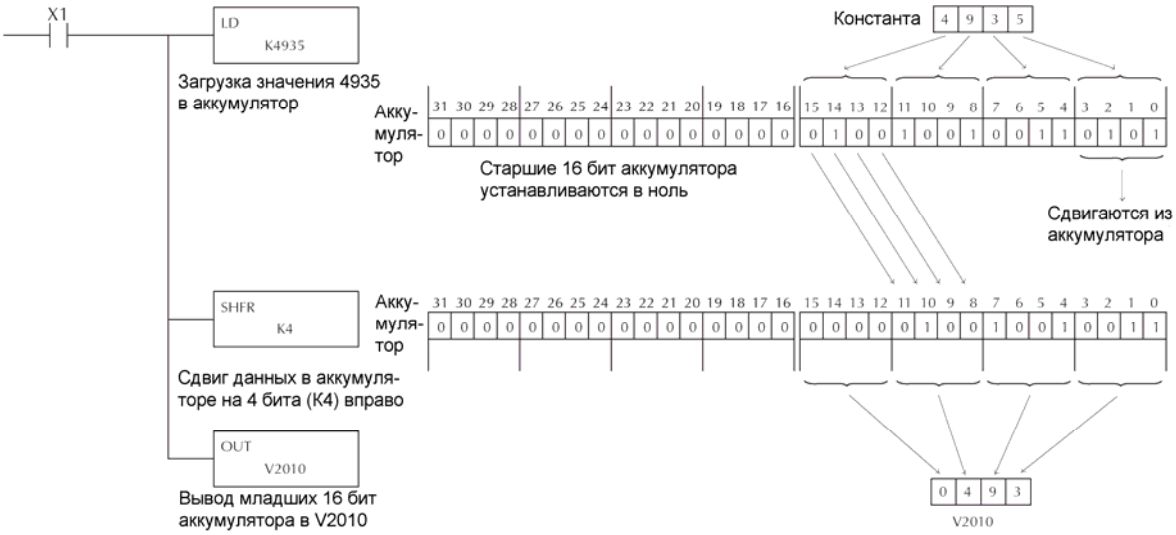
Команды Load и Out и их вариации используются для копирования данных из ячейки V-памяти в аккумулятор, или для копирования данных из аккумулятора в V-память. В следующем примере копируются данные из ячейки V-памяти V2000 в ячейку V-памяти V2010.



Так как аккумулятор 32-битный, а ячейки V-памяти 16-битовые, то Load Double и Out Double (или их вариации) используют две последовательные ячейки V-памяти или 8-значные BCD константы для копирования данных или из аккумулятора в адрес V-памяти, или из адреса V-памяти в аккумулятор. Например, если Вам требуется скопировать данные из ячеек V-памяти V2000 и V2001 в ячейки V-памяти V2010 и V2011, то наиболее эффективный способ выполнить эту операцию будет следующим:

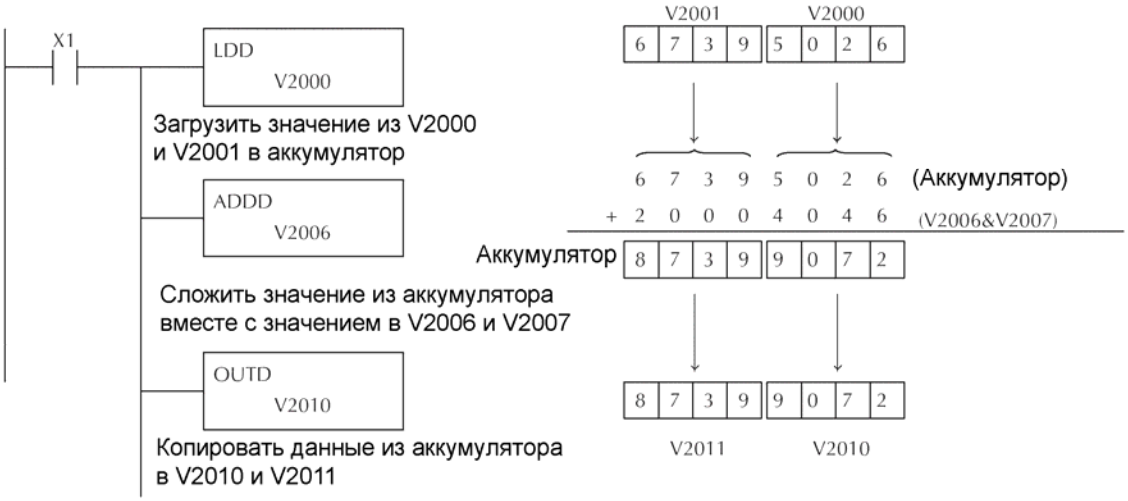
Изменение данных аккумулятора

Команды, которые оперируют данными, также используют аккумулятор. Результат операций с данными постоянно находится в аккумуляторе. Данные, которые были обработаны, стираются из аккумулятора. В следующем примере загрузка константы 4935 в аккумулятор смещает данные вправо на 4 бита и выводит результат в V2010.



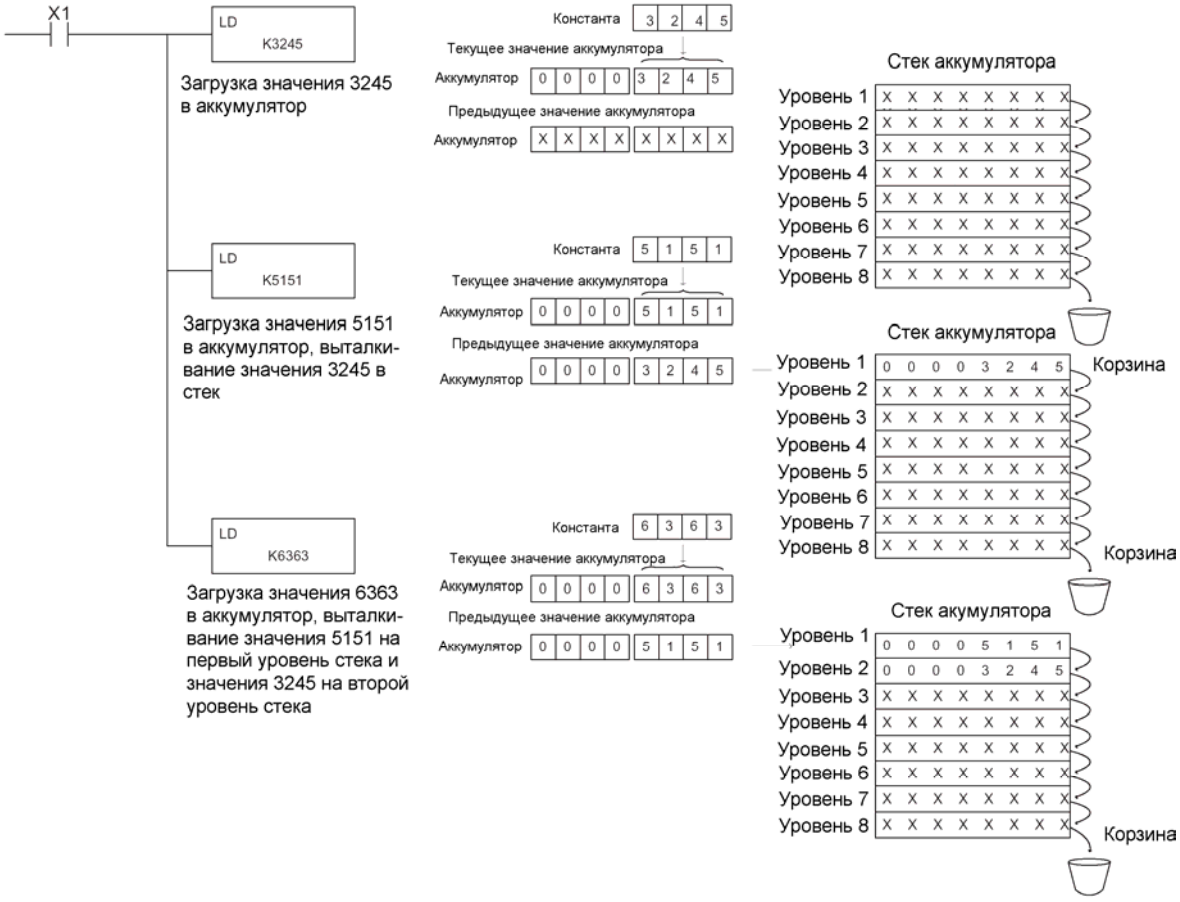
Некоторые из команд, оперирующих с данными, используют 32 бита. Они используют две последовательные ячейки V-памяти или 8-значные BCD константы для операций с данными в аккумуляторе.

В следующем примере, при включении X1, значение из V2000 и V2001 будет загружено в аккумулятор, используя команду Load Double. К содержимому аккумулятора добавляется значение из V2006 и V2007 командой Add Double. Значение в аккумуляторе копируется в V2010 и V2011, используя команду Out Double.

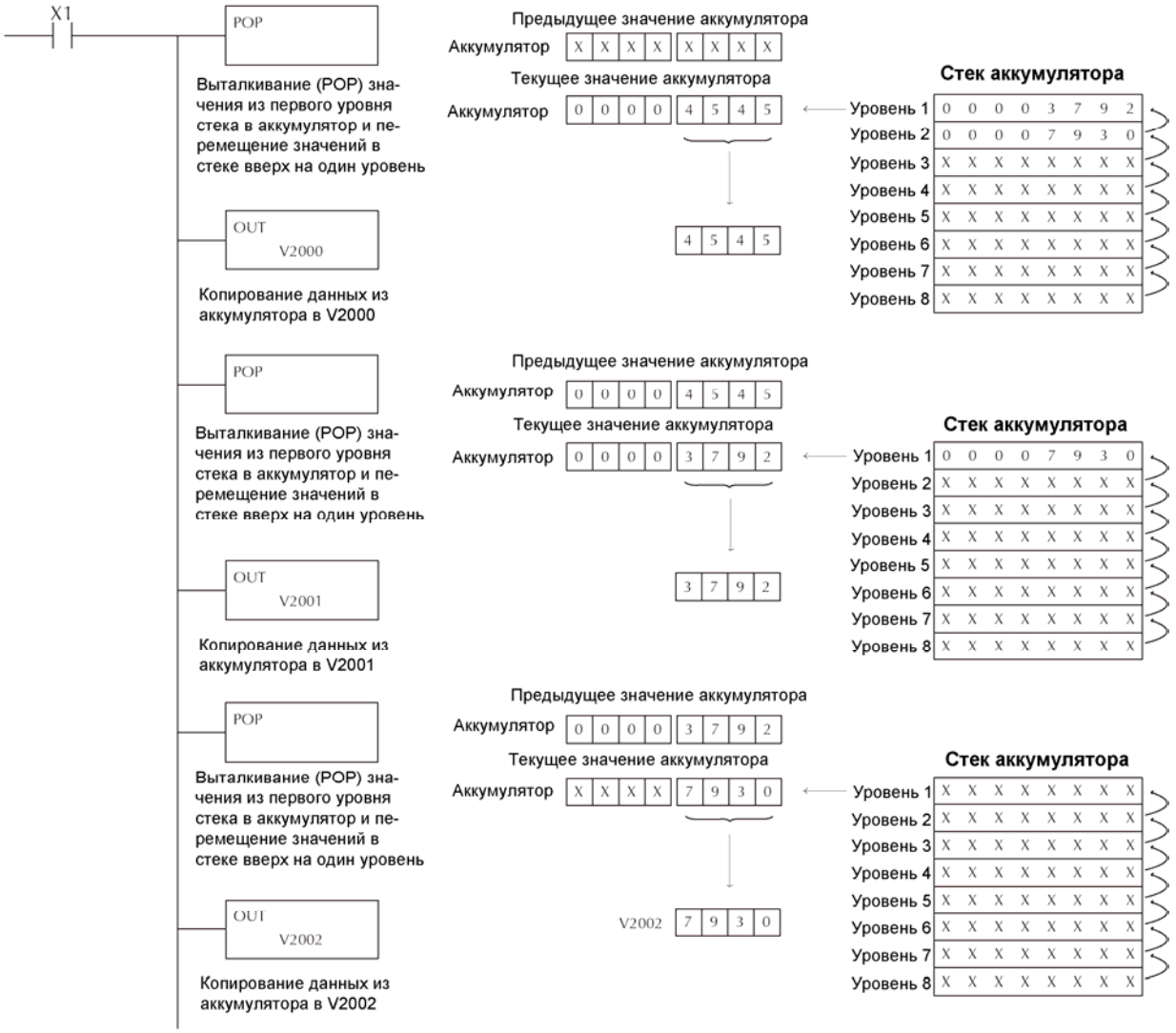


Использование стека аккумулятора

Стек аккумулятора используется для команд, которые требуют более одного параметра для выполнения функции или для функций, определяемых 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-битных 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 (выталкивает) 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 одну позицию.

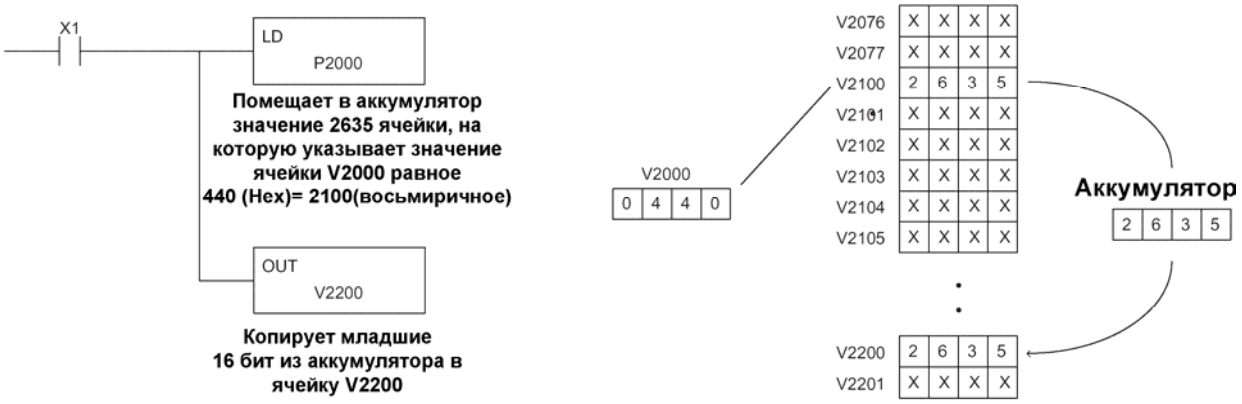


Использование указателей

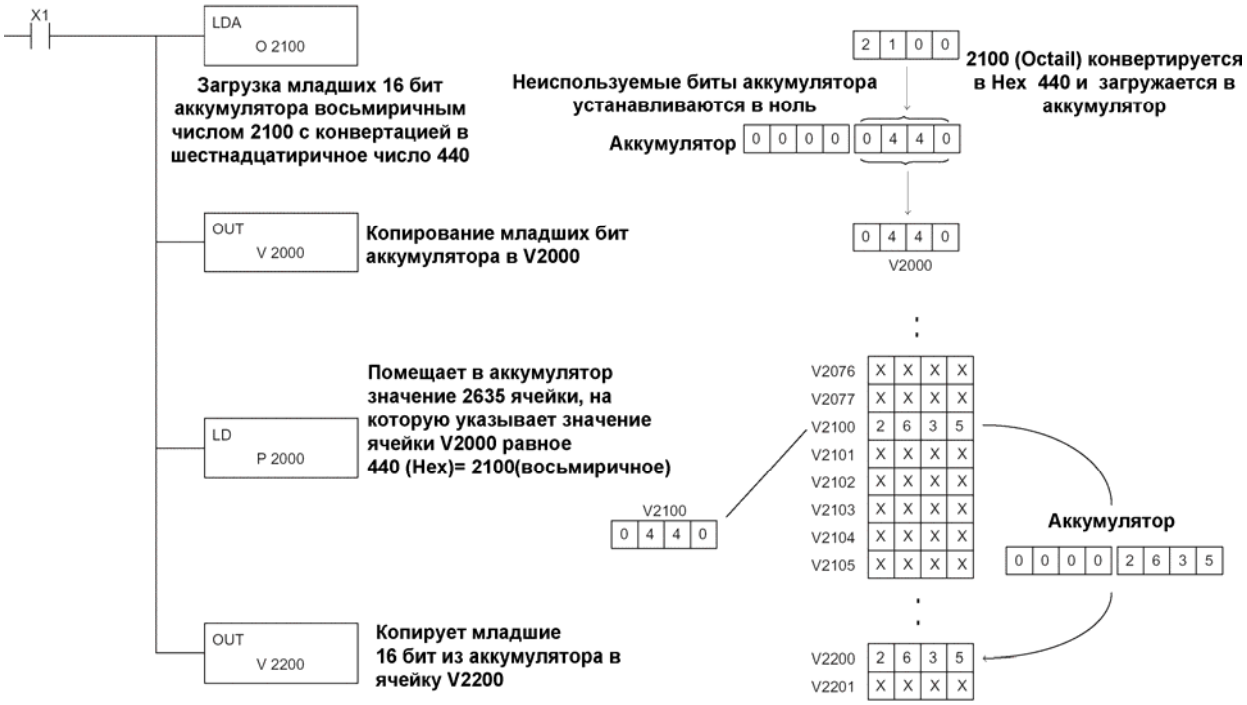
Многие из серии команд, используемых с DL06, позволяют использовать указатели V-памяти (Pointers) в качестве операнда (косвенная адресация). Указатели позволяют командам получать данные из ячеек V-памяти, на которые ссылается значение указателя..

ПРИМЕЧАНИЕ : в DL06 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 (LDA) для передачи адреса в ячейку указателя. Эта команда выполняет преобразование из восьмеричного формата в шестнадцатиричный автоматически..

В следующем примере мы используем операнд указателя в команде Load. Ячейка V-памяти V2000 является ячейкой указателя. V2000 содержит значение 440, которое является шестнадцатиричным эквивалентом восьмеричного адреса ячейки V-памяти V2100. ЦПУ копирует данные из V2100 (в примере — число 2635) в младшее слово аккумулятора.

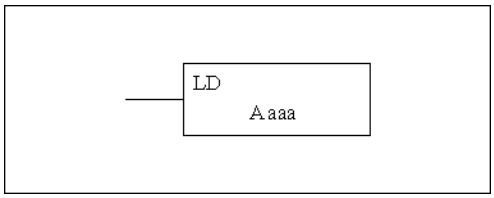


Следующий пример похож на пример, приведенный выше, если бы не команда LDA (Load Address), которая автоматически преобразует восьмеричный адрес в шестнадцатиричный эквивалент.



Load (LD)

Команда Load — это 16-битная команда, которая загружает значение (Aaaa), которое является или ячейкой V-памяти или 4-разрядной константой, в нижние 16 бит аккумулятора. Верхние 16 бит аккумулятора устанавливаются в 0.

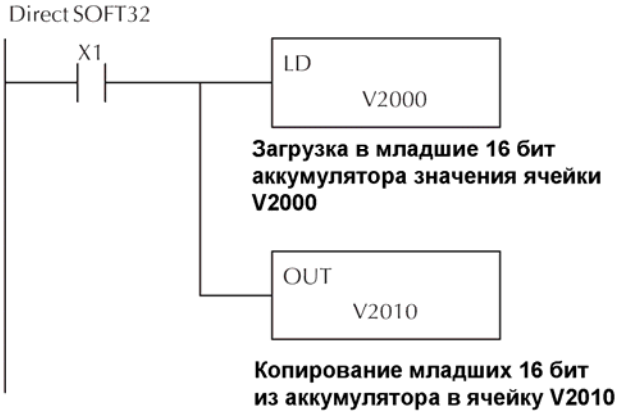


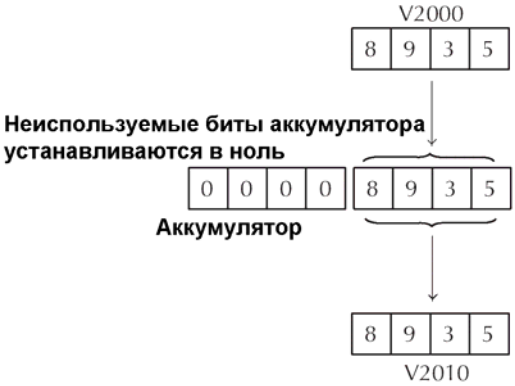
Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-FFFF

Флаги	Описание
SP53	«1», когда значение указателя вне доступного диапазона
SP70	«1», когда значение в аккумуляторе отрицательное
SP76	«1», когда значение, загружаемое в аккумулятор любой командой, является нулем

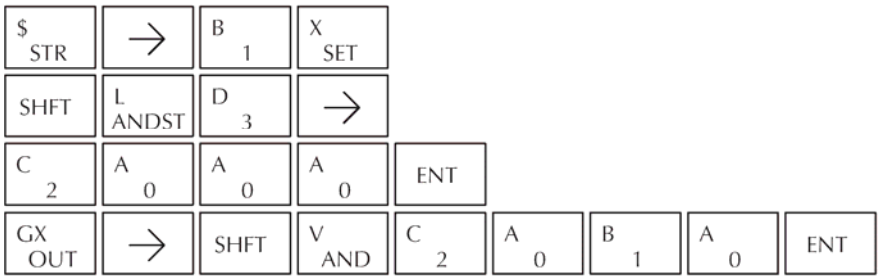
ПРИМЕЧАНИЕ. Две последовательных команды Load поместят значение первой команды Load в стек аккумулятора

В следующем примере, когда X1 включен, значение в V2000 будет загружено в аккумулятор и выведено в V2010.



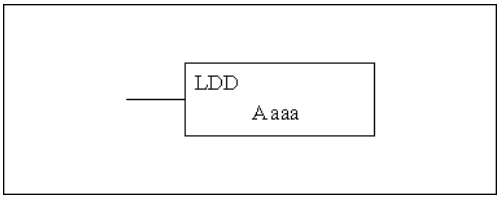


Набор на ручном программаторе



Load Double (LDD)

Команда Load Double — это 32-битная команда, которая загружает значение (Aaaa), которое является или двумя последовательными ячейками V-памяти или 8-разрядной константой, в аккумулятор.

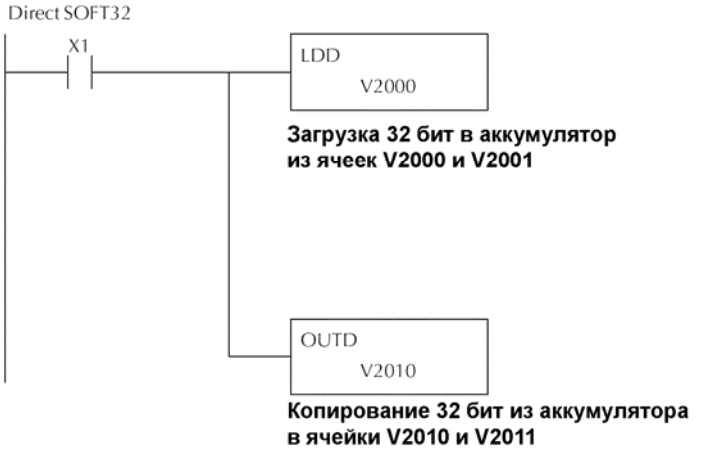


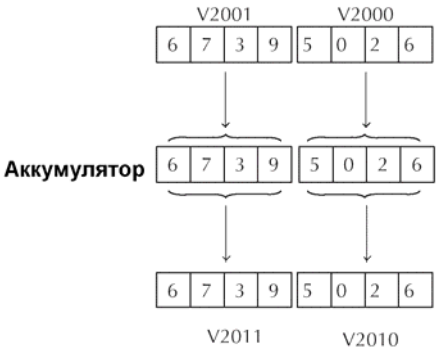
Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-FFFF

Флаги	Описание
SP53	«1», когда значение указателя вне доступного диапазона
SP70	«1», когда значение в аккумуляторе отрицательное
SP76	«1», 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, 32-битовое значение в V2000 и V2001 будет загружено в аккумулятор и выведено в V2010 и V2011.



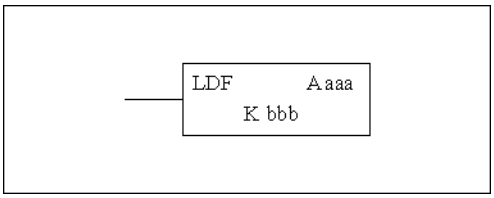


Набор на ручном программаторе

$ STR	→	B 1	ENT	
SHFT	L ANDST	D 3	D 3	→
C 2	A 0	A 0	A 0	ENT
GX OUT	SHFT	D 3	→	
C 2	A 0	B 1	A 0	ENT

Load Formatted (LDF)

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 ноль.



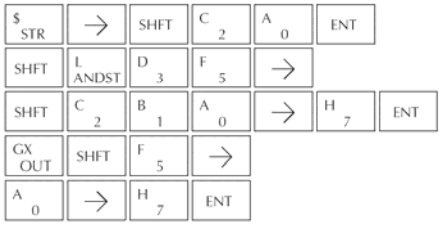
Тип данных операнда		Диапазон	
	A	**aaa**	**bbb**
Входы	X	0-777	--
Выходы	Y	0-777	--
Реле управления	C	0-1777	--
Биты стадии	S	0-1777	--
Биты таймера	T	0-377	--
Биты счетчика	CT	0-177	--
Специальные реле	SP	0-777	--
Константа	K	--	1-32

Флаги	Описание
SP70	«1», когда значение в аккумуляторе отрицательное
SP76	«1», когда значение, загружаемое в аккумулятор любой командой, является нулем

ПРИМЕЧАНИЕ. Две последовательных команды Load поместят значение первой команды Load в стек аккумулятора.

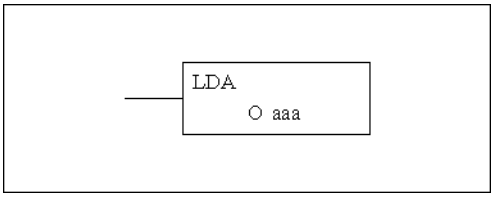
В следующем примере, когда C0 включен, двоичные ячейки C10-C16 (7 бит) будут загружены в аккумулятор используя команду Load Formatted. Младшие 6 бит аккумулятора выводятся на Y20-Y26 используя команду Out Formatted.

Набор на ручном программаторе



Load Address (LDA)

Команда Load Address является 16-битной командой. Она преобразует любое восьмеричное значение или адрес в шестнадцатиричный эквивалент и загружает шестнадцатиричное значение в аккумулятор. Эта команда полезна, когда требуется использовать в качестве параметра адрес, так как все адреса для системы DL06 являются в восьмеричными.

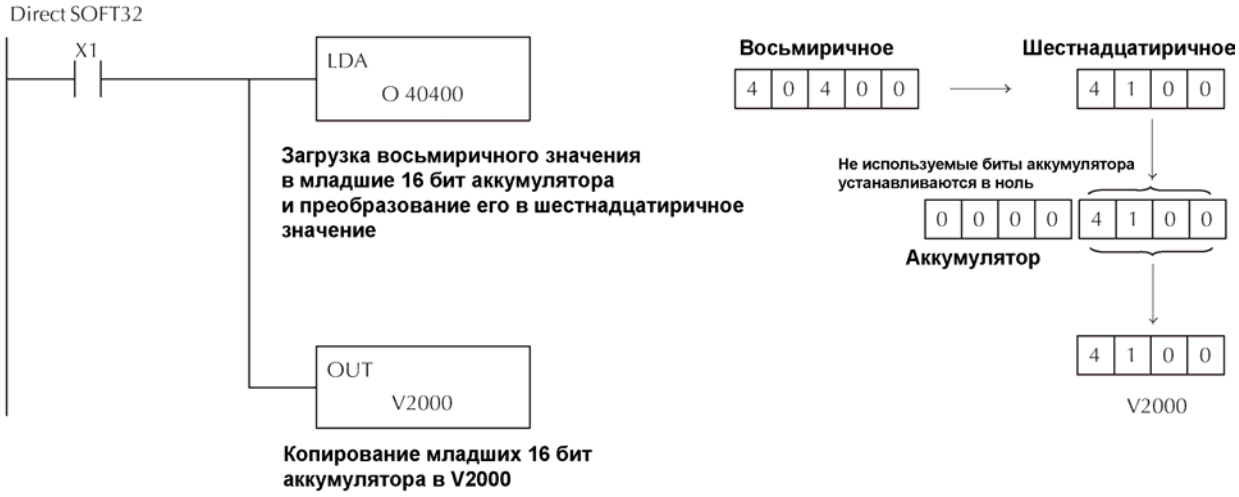


Тип данных операнда	Диапазон
	aaa
Восьмеричный адрес O	Смотри карту памяти

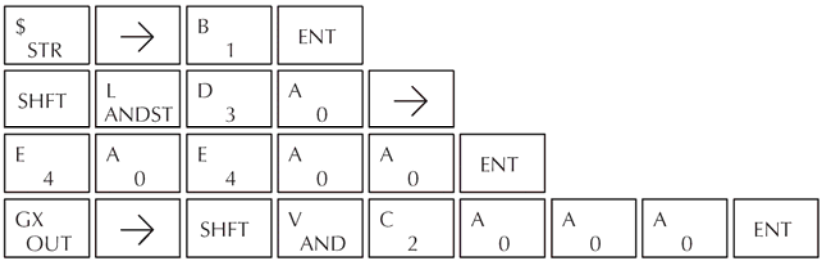
Флаги	Описание
SP70	«1», когда значение в аккумуляторе отрицательное
SP76	«1», когда значение, загружаемое в аккумулятор любой командой, является нулем

ПРИМЕЧАНИЕ: Две последовательных команды Load поместят значение первой команды Load в стек аккумулятора.

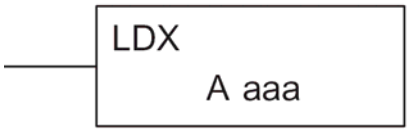
В следующем примере, когда X1 включен, восьмеричное значение 40400 будет преобразовано в шестнадцатиричное значение 4100 и загружено в аккумулятор используя команду Load Address. Значение из младших 16 бит аккумулятора будет скопировано в V2000 используя команду Out.



Набор на ручном программаторе



Load Accumulator Indexed (LDX)



Load Accumulator Indexed – это 16-битная команда, которая загружает в аккумулятор значение ячейки V-памяти, определяя исходный адрес и добавляя к исходному адресу смещение заданное в первой ячейке стека.

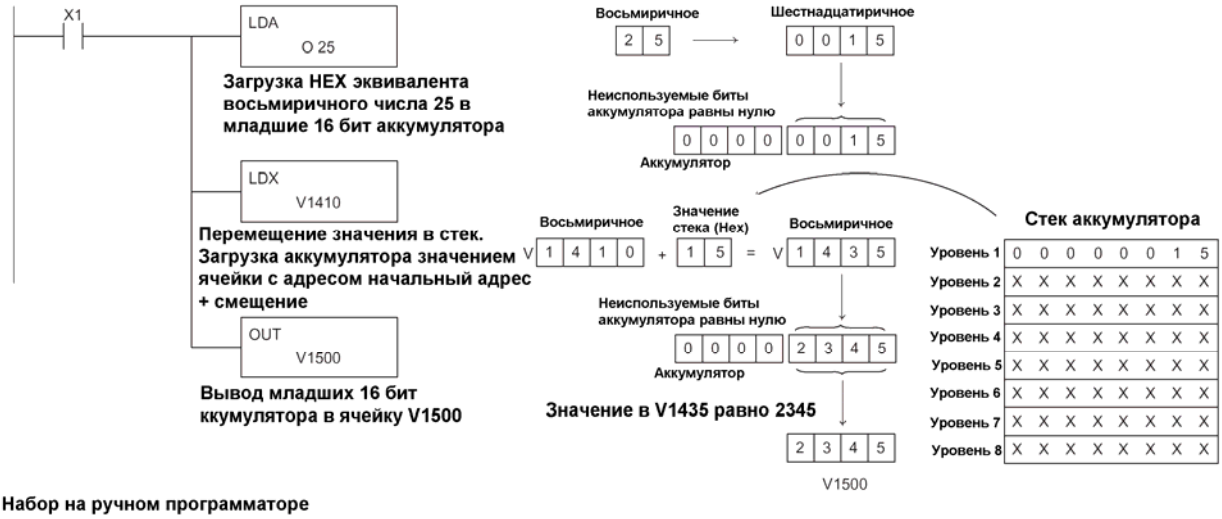
Эта инструкция интерпретирует значение в первой ячейке стека как шестнадцатеричное. Значение смещения адреса (исходный адрес + смещение) должно быть загружено в младшие 16 бит аккумулятора. Старшие 16 бит аккумулятора установлены в 0.

Полезный совет: Команду Load Address можно использовать для конвертирования восьмиричного адреса в шестнадцатиричный адрес и загрузки значения ячейки в аккумулятор.

Тип данных операнда		Диапазон
		aaa
V-память	**V**	Смотри карту памяти
Указатель	**P**	Смотри карту памяти

ПРИМЕЧАНИЕ: Две последовательных команды Load поместят значение первой команды Load в стек аккумулятора.

В следующем примере, когда X1 включен, шестнадцатиричный эквивалент восьмиричного числа 25 будет загружен в аккумулятор (это значение будет перемещено в стек при выполнении команды Load Accumulator Indexed). К адресу ячейки V-памяти V1410 добавляется значение из первого уровня стека и значение этой ячейки (V1435 = 2345) загружается в младшие 16 бит аккумулятора с использованием команды Load Accumulator Indexed. Значение младших 16 бит аккумулятора выводятся в ячейку V1500 с использованием команды Out.



Набор на ручном программаторе

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	A 0	→	C 2	F 5	ENT	
SHFT	L ANDST	D 3	X SET	→	B 1	E 4	B 1	A 0	ENT
GX OUT	→	PREV	PREV	PREV	B 1	F 5	A 0	A 0	ENT

Load Accumulator Indexed from Data Constants (LDSX)

Команда Load Accumulator Indexed from Data Constants является 16-битной. Команда определяет Область меток данных Data Label Area (DLBL) где хранятся числовые или ASCII константы. Значение будет загружено в младшие 16 бит.

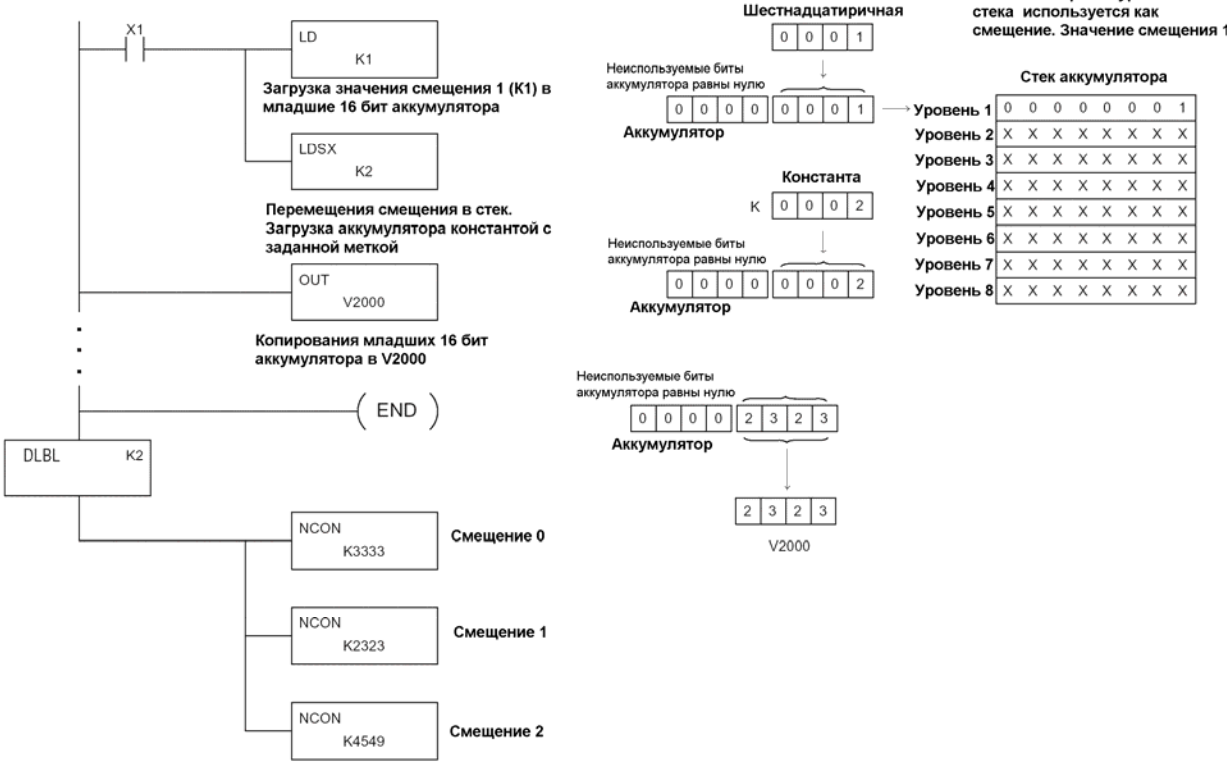
Команда LDSX использует значение в ячейки первого уровня стека аккумулятора как смешение, чтобы определить какую числовую или ASCII константу в пределах Области меток данных загрузить в аккумулятор. Команда LDSX Интерпретирует значение в первом уровне стека аккумулятора как шестнадцатиричное значение.

Полезный совет: Команду Load Address можно использовать для конвертирования восьмиричного адреса в шестнадцатиричный адрес и загрузки значения ячейки в аккумулятор.

Тип данных операнда		Диапазон
		aaa
Константа	**K**	1-FFFF

ПРИМЕЧАНИЕ: Две последовательных команды Load поместят значение первой команды Load в стек аккумулятора.

В следующем примере, когда X1 включен, смещение 1 загружается в аккумулятор. Это значение будет перемещено в стек при выполнении LDSX. Команда LDSX определяет метку данных (DLBL K2), где находится числовая константа(ы) в программе и загружает значение константы, с указанным в стеке смещением в младшие 16 бит аккумулятора.

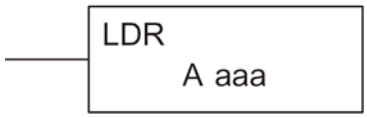


Набор на ручном программаторе

$ STR	→	B 1	ENT							
SHFT	L ANDST	D 3	→	SHFT	K JMP	B 1	ENT			
SHFT	L ANDST	D 3	S RST	X SET	→	C 2	ENT			
SHFT	E 4	N TMR	D 3	ENT						
SHFT	D 3	L ANDST	B 1	L ANDST	→	C 2	ENT			
SHFT	N TMR	C 2	O INST#	N TMR	→	D 3	D 3	D 3	D 3	ENT
SHFT	N TMR	C 2	O INST#	N TMR	→	C 2	D 3	C 2	D 3	ENT
SHFT	N TMR	C 2	O INST#	N TMR	→	E 4	F 5	E 4	J 9	ENT
GX OUT	→	SHFT	V AND	C 2	A 0	A 0	A 0	ENT		

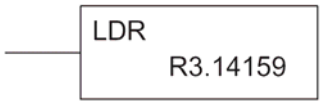
Load Real Number (LDR)

Команда Load Real Number загружает вещественное число, содержащееся в двух последовательных ячейках V-памяти, или в 8-разрядной константе аккумулятора.



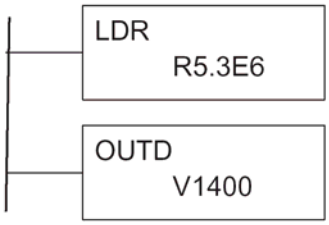
Тип данных операнда		Диапазон
	A	aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Вещественная константа	R	-3.402823E+038 ÷ +3.402823E+038

DirectSOFT32 разрешает Вам непосредственно вводить вещественные числа, пользуясь индексом “R”, при вводе вещественного числа. Вы можете вводить константу такую как ПИ, показанную в примере справа. Чтобы ввести отрицательные числа, используйте знак минус (-) после “R”.



Для очень больших или очень малые чисел, Вы можете использовать экспоненциальное представление. Число справа - 5.3 миллиона. Команда OUTD сохраняет это число в V1400 и V1401.

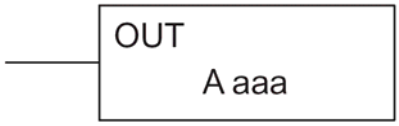
Эти вещественные числа сохраняются в 32-разрядном формате IEEE с плавающей запятой, так что они занимают две ячейки V-памяти, независимо от того, насколько большое или малое число сохраняется! Если Вы просматриваете сохраненное вещественное число в шестнадцатеричном, двоичном, или четном BCD, показанное число будет очень трудно декодировать. Аналогично всем другим типам чисел, Вы должны следить за размещением вещественного числа в памяти, поскольку они могут считываться только соответствующими командами рассматриваемыми позже.



В предыдущем примере, приведенном выше, вещественное число сохранено в V1400 и V1401. Предположите, что теперь нам требуется извлечь то число. Используйте Load Real с типом данных V, как показано справа. После этого Вы можете выполнять математические действия с вещественными числами или преобразовать это в двоичное число.

Out (OUT)

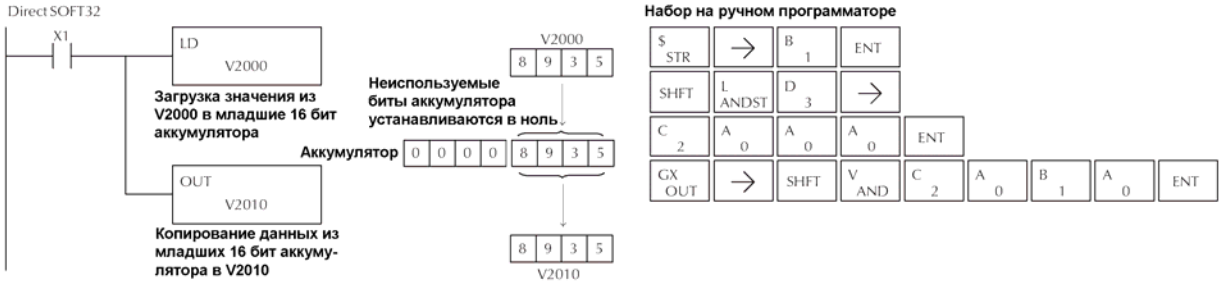
Команда Out — это 16-битная команда, которая копирует значение из младших 16 бит аккумулятора в заданную ячейку V-памяти (Aaaa).



Тип данных операнда		Диапазон
	A	aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти

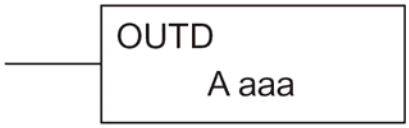
Флаги	**Описание**
SP53	«1», если ЦПУ не может решить логику

В следующем примере, когда X1 включен, значение в V2000 будет загружено в младшие 16 бит аккумулятора, используя команду Load. Значение из младших 16 бит аккумулятора копируется в V2010 командой Out.



Out Double (OUTD)

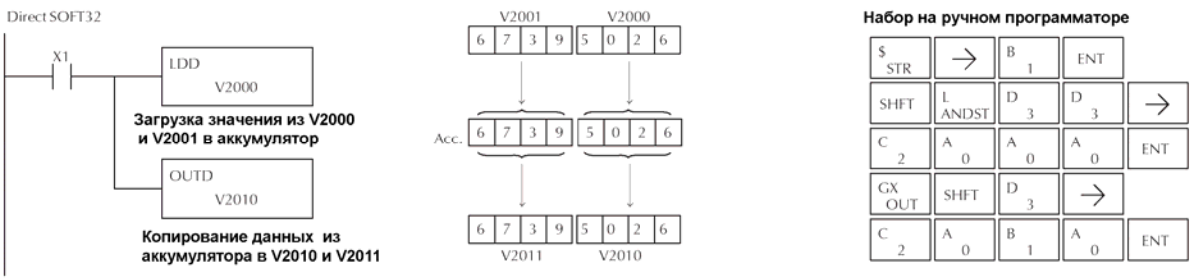
Команда Out Double — это 32-битная команда, которая копирует значение из аккумулятора в две последовательных ячейки V-памяти с указанной стартовой ячейки (Aaaa).



Тип данных операнда		Диапазон
	A	aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти

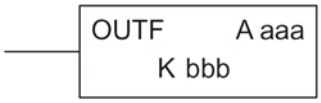
Флаги	**Описание**
SP53	«1», если ЦПУ не смогло решить логическую схему

В следующем примере, когда X1 включен, 32 разрядное значение в V2000 и V2001 будет загружено в аккумулятор, используя команду Load Double. Значение в аккумуляторе выводится в V2010 и V2011, используя команду Out Double.



Out Formatted (OUTF)

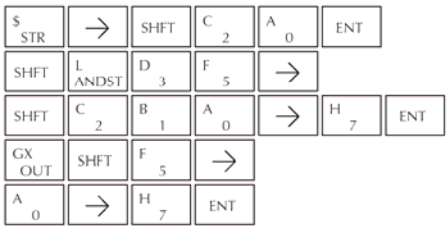
Команда Out Formatted выводит 1-32 бит из аккумулятора в указанные дискретные ячейки памяти. Для вывода команды требуется определить стартовую ячейку (Aaaa) для адреса и количество бит (Kbbb).



Тип данных операнда		DL06 Диапазон	
	A	aaa	bbb
Входы	X	0-777	--
Выходы	Y	0-777	--
Реле управления	C	0-1777	--
Константа	K	--	1-32

В следующем примере, когда C0 включен, двоичное значение в C10-C16 (7 бит) будет загружено в аккумулятор, используя команду Load Formatted. Младшие 7 бит аккумулятора выводятся на Y0-Y6, используя команду Out Formatted.

Набор на ручном программаторе



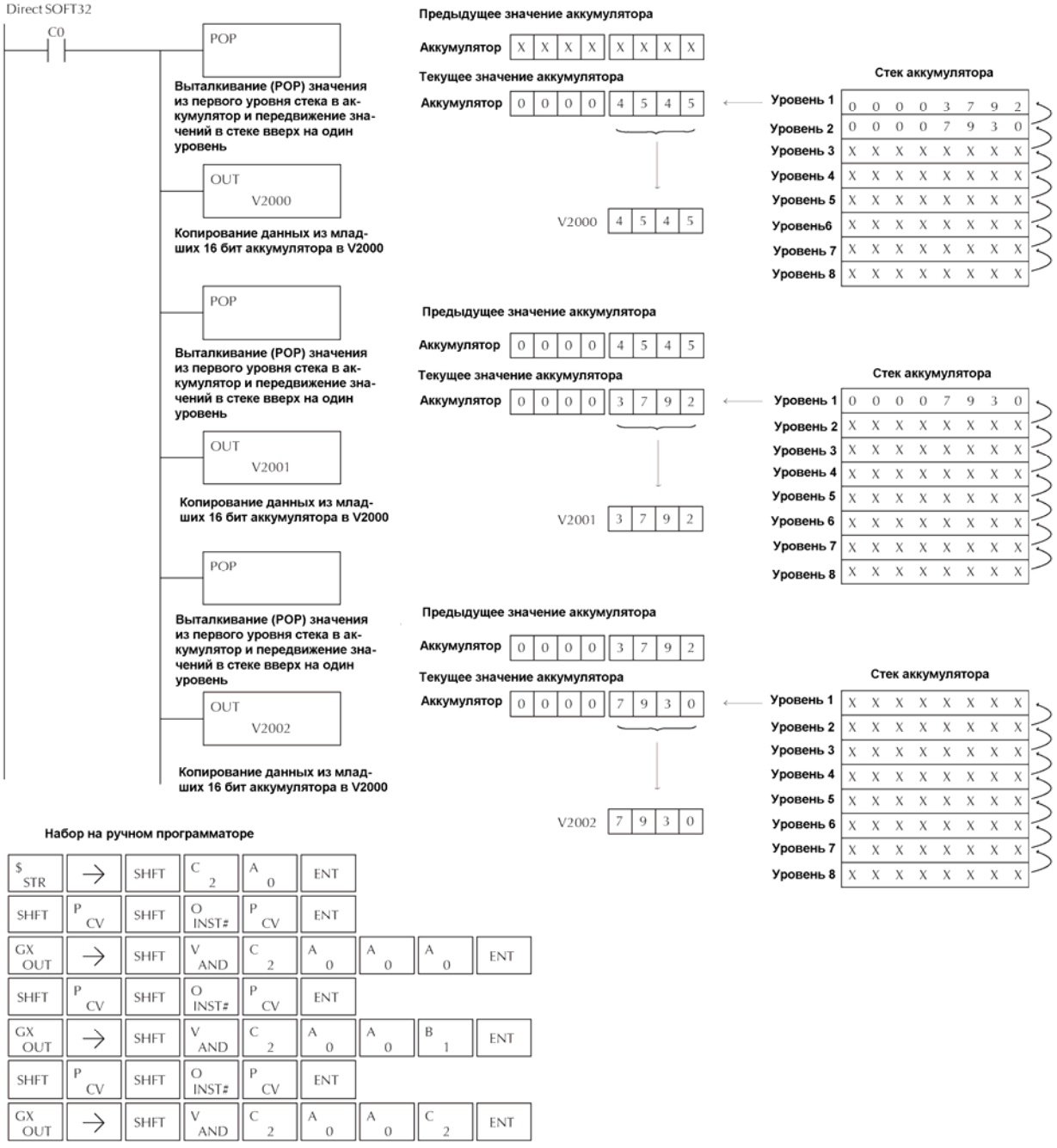
Pop (POP)

Команда Pop перемещает значение с первого уровня стека аккумулятора (32 бита) в аккумулятор и сдвигает каждое значение в стеке вверх на один уровень.

Флаги	Описание
SP63	«1», когда значение, загруженное в аккумулятор любой командой, является нулем

Продолжение описание команды Pop

В следующем примере, когда C0 включен, значение 4545, которое было на вершине стека, перемещается в аккумулятор, используя команду Pop. Значение выводится в V2000, используя команду Out. Следующая команда Pop перемещает значение 3792 в аккумулятор и выводит значение в V2001. Последняя команда Pop перемещает значение 7930 в аккумулятор и выводит значение в V2002. Пожалуйста, обратите внимание, если значение в стеке было бы больше, чем 16 бит (4 знака), то использовалась бы команда Out Double и для каждой команды Out Double должны быть назначены 2 ячейки V-памяти.



Набор на ручном программаторе

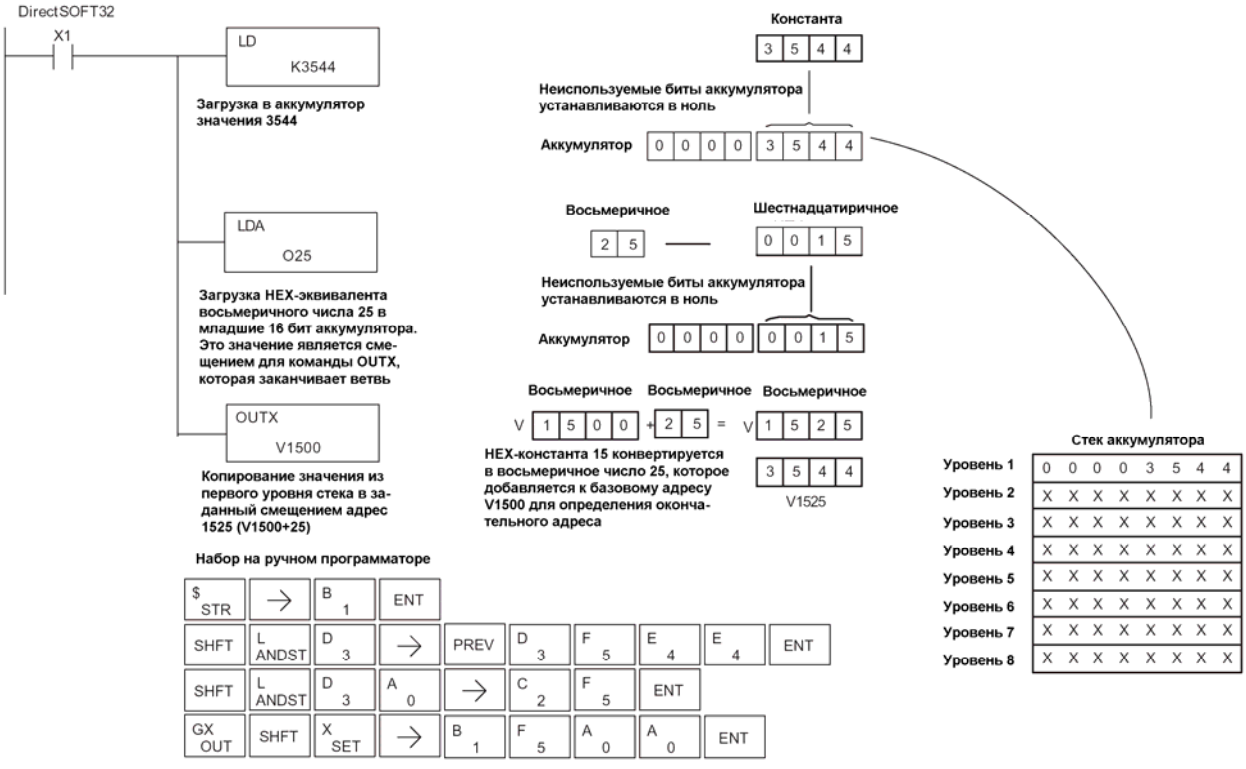
$ STR	→	SHFT	C 2	A 0	ENT			
SHFT	P CV	SHFT	O INST#	P CV	ENT			
GX OUT	→	SHFT	V AND	C 2	A 0	A 0	A 0	ENT
SHFT	P CV	SHFT	O INST#	P CV	ENT			
GX OUT	→	SHFT	V AND	C 2	A 0	A 0	B 1	ENT
SHFT	P CV	SHFT	O INST#	P CV	ENT			
GX OUT	→	SHFT	V AND	C 2	A 0	A 0	C 2	ENT

Out Indexed (OUTX)

Команда Out Indexed – это 16-разрядная команда. Она копирует 16 бит или 4-х-разрядное значение из первого уровня стека аккумулятора в ячейку памяти с адресом – исходный адрес + смещение. Эта команда интерпретирует значение смещения как шестнадцатиричное число. Старшие 16 бит аккумулятора устанавливаются в ноль.

Тип данных операнда		Диапазон DL06
	A	aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти

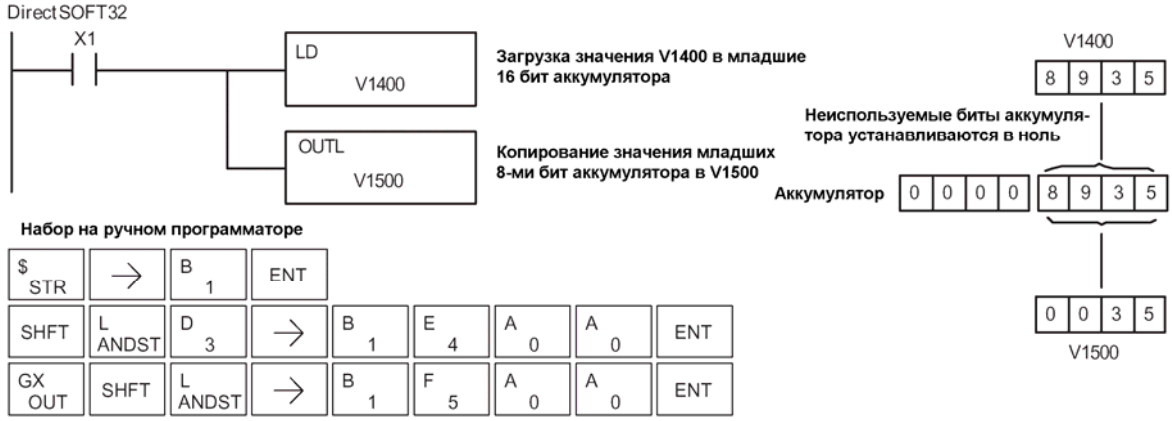
В следующем примере, когда X1 включен, значение константы 3544 загружено в аккумулятор. Это - значение, которое будет выводиться с заданным смещением в ячейку V-памяти (V1525). Значение 3544 будет помещено в стек, когда команда Load Address выполнится. Помните, что две последовательных команды Load помещают значение первой команды Load в стек. Команда Load Address преобразовывает восьмеричное число 25 в шестнадцатиричное число 15 и помещает значение в аккумулятор. Команда Out Indexed выводит значение 3544, которое находится в первом уровне стека аккумулятора в V1525.



Out Least (OUTL)

Команда Out Least копирует значение в младшие восемь бит аккумулятора в заданные младшие восемь бит ячейки V-памяти (то есть, копируется младший байт младшего слова аккумулятора).

В следующем примере, когда X1 включен, значение в V1400 будет загружено в младшие 16 битов аккумулятора, используя команду Load. Значение младших 8 битов аккумулятора копируется в V1500, используя команду Out Least.

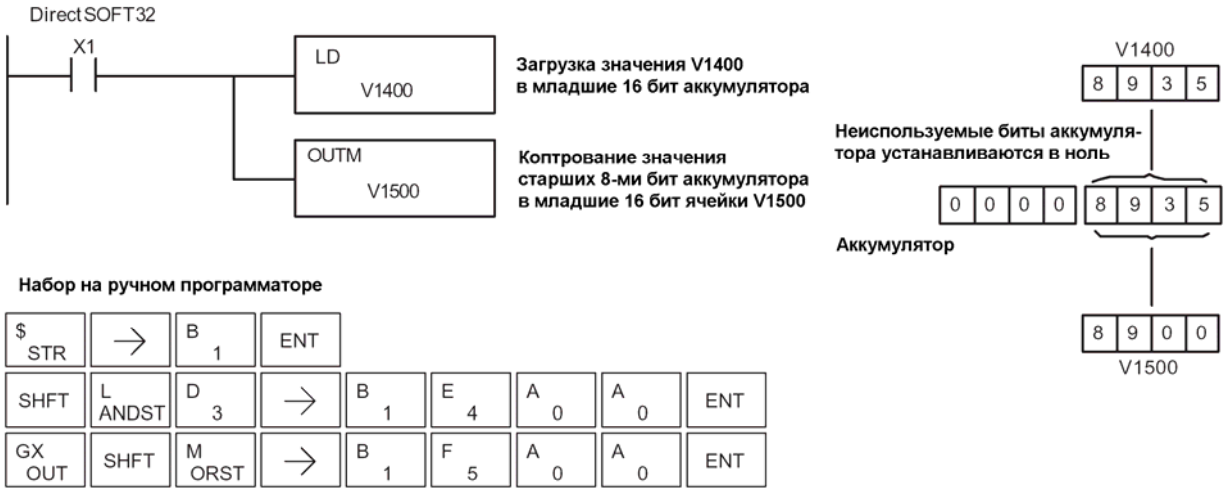


Out Most (OUTM)

Команда Out Most копирует значение старших 8-ми битов из младших шестнадцати битов аккумулятора в старшие восемь бит заданной ячейки V-памяти (то есть, копируется старший байт младшего слова акумулятора).

Тип данных операнда		Диапазон DL06
	A	aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти

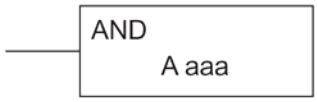
В следующем примере, когда X1 включен, значение в V1400 будет загружено в младшие 16 битов аккумулятора, используя команду Load. Значение в старших 8 бит аккумулятора копируется в младшие 16 бит V1500, используя команду Out Most.



Логические команды аккумулятора

And (AND)

Команда And — это 16-битная 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 DL06
	A	ааа
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти

Флаги	Описание
SP63	«1», если результат в аккумуляторе является нулем
SP70	«1», когда значение, загруженное в аккумулятор любой командой, является нулем

ПРИМЕЧАНИЕ: Флаги состояния доступны только до тех пор, пока не будет выполнена другая команда, использующая те же самые флаги

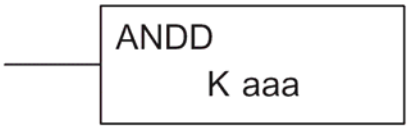
В следующем примере, когда X1 включен, значение в V2000 будет загружено в аккумулятор, используя команду Load. Выполняется операция логического И над значением из аккумулятора и значением из V2006. Значение из младших 16 бит аккумулятора выводится в V2010, используя команду Out.

Набор на ручном программаторе

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→	C 2	A 0	A 0	A 0	ENT
V AND	→	SHFT	V AND	C 2	A 0	A 0	G 6	ENT
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT

And Double (ANDD)

Команда And Double — это 32-битная команда, которая выполняет операцию логического И над значением из аккумулятора вместе с двумя последовательными ячейками V-памяти или 8-ми разрядным (максимум) значением константы (Аааа). Результат находится в аккумуляторе. Дискретные флаги состояния указывают, является ли результат команды And Double нулем или отрицательным числом (старший бит включен)

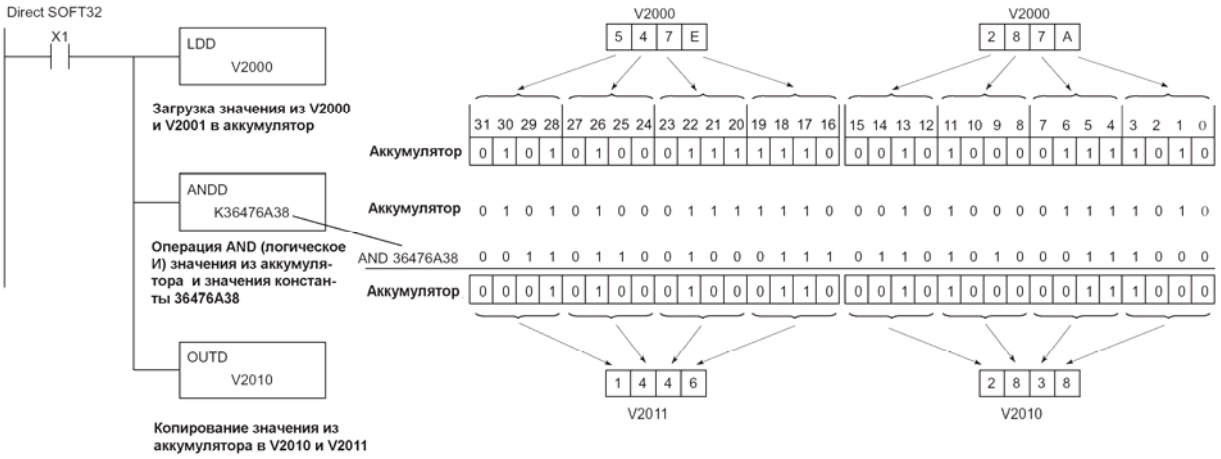


Тип данных операнда		Диапазон DL06
		aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-FFFFFFFF

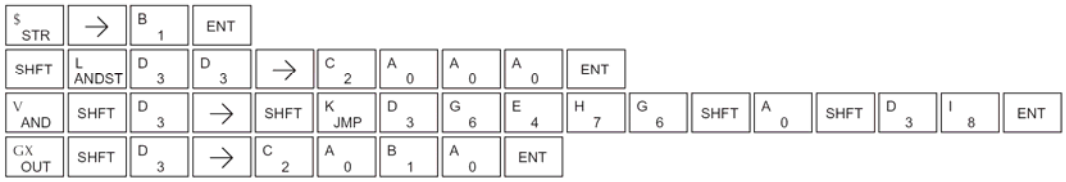
Флаги	Описание
SP63	«1», если результат в аккумуляторе является нулем
SP70	«1», если результат в аккумуляторе является отрицательным

ПРИМЕЧАНИЕ: Флаги состояния доступны только до тех пор, пока не будет выполнена другая команда, использующая те же самые флаги

В следующем примере, когда X1 включен, значение в V2000 и V2001 будет загружено в аккумулятор, используя команду Load Double. Выполняется операция логического И над значением из аккумулятора и 36476A38, используя команду And Double. Значение из аккумулятора выводится в V2010 и V2011 используя команду Out Double..

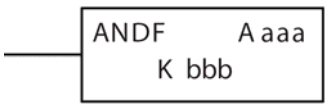


Набор на ручном программаторе



And Formatted (ANDF)

Команда And Formatted – это команда, которая выполняет операцию логического И над двоичным значением аккумулятора и заданным диапазоном бит памяти (1-32). Команда требует адрес исходной ячейки (Aaaa) и числа бит (Kbbb), чтобы выполнить операцию логического И. Дискретные флаги состояния указывают, является ли результат команды And Double нулем или отрицательным числом (старший бит=1).

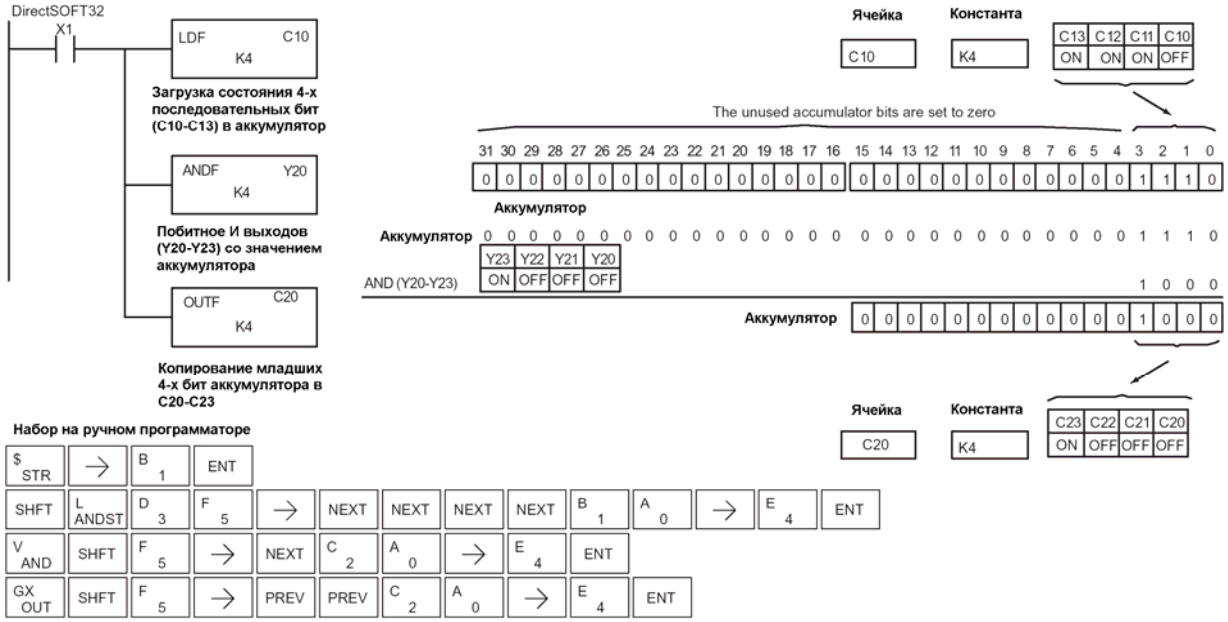


Тип данных операнда		Диапазон DL06	
	B	aaa	bbb
Входы	X	0-777	-
Выходы	Y	0-777	-
Реле управления	C	0-1777	-
Бит стадии	S	0-1777	-
Бит таймера	T	0-377	-
Бит счетчика	CT	177	-
Специальные реле	SP	0-777	-
Константа	K	-	1-32

Флаги	Описание
SP63	«1», если результат в аккумуляторе является нулем
SP70	«1», если результат в аккумуляторе является отрицательным

ПРИМЕЧАНИЕ: Флаги состояния доступны только до тех пор, пока не другая команда не использует те же самые флаги

В следующем примере, когда X1 включен, команда Load Formatted загружает C10–C13 (4 бита) в аккумулятор. . Выполняется операция логического И содержимого аккумулятора вместе с конфигурацией битов из Y20–Y23, используя команду And Formatted. Команда Out Formatted выводит 4 младших бита аккумулятора в C20-C23.



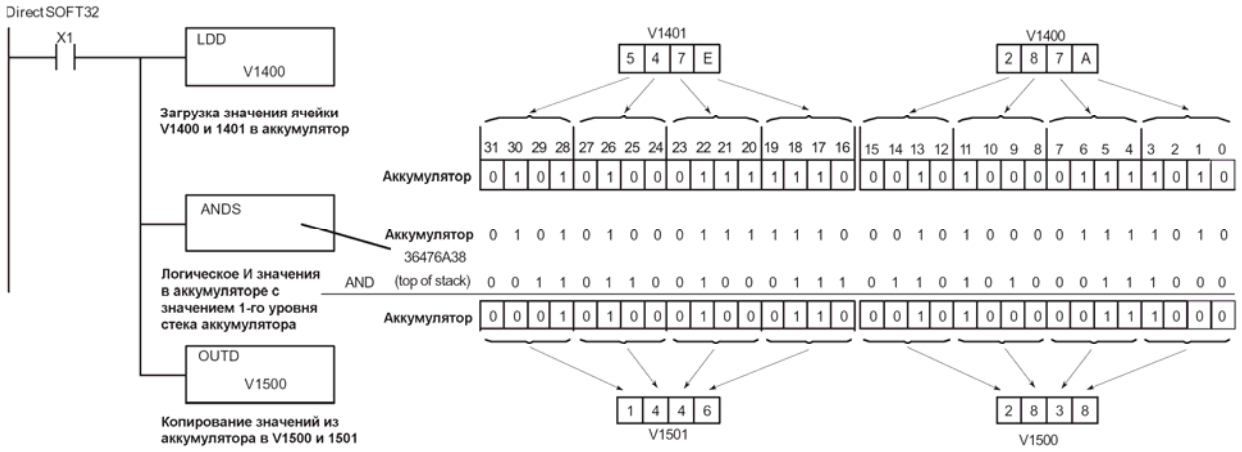
And with Stack (ANDS)

Команда And with Stack - это 32-битная команда, которая выполняет операцию логического И над значением из аккумулятора вместе с значением, расположенным на первом уровне стека аккумулятора. Результат находится в аккумуляторе.Удаляется из стека значение , расположенным на первом уровне стека аккумулятора, и все значения перемещаются на один уровень вверх. Дискретные флаги состояния указывают, является ли результат команды And with Stack нулем или отрицательным числом (старший бит =1)

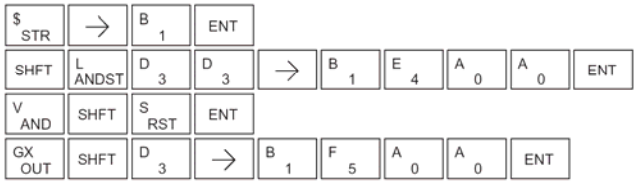
Флаги	Описание
SP63	«1», если результат в аккумуляторе является нулем
SP70	«1», если результат в аккумуляторе является отрицательным

ПРИМЕЧАНИЕ: Флаги состояния доступны только до тех пор, пока не другая команда не использует те же самые флаги

В следующем примере, когда X1 включен, выполнится операция логического И двоичного значения в аккумуляторе и двоичного значения на первом уровне стека аккумулятора. Результат находится в аккумуляторе. Затем 32 разрядное значение выводится в V1500 и V1501

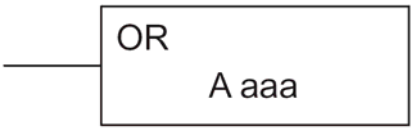


Набор на ручном программаторе



Or (OR)

Команда Or — это 16-битная команда, которая выполняет операцию логического ИЛИ 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 Or нулем.



Тип данных операнда	Диапазон DL06
	aaa
V-память V	Смотри карту памяти
Указатель P	Смотри карту памяти

Флаги	Описание
SP63	«1», если результат в аккумуляторе является нулем
SP70	«1», если результат в аккумуляторе является отрицательным

ПРИМЕЧАНИЕ: Флаги состояния доступны только до тех пор, пока не другая команда не использует те же самые флаги

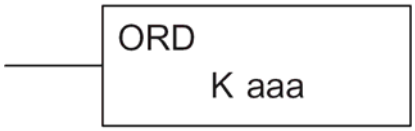
В следующем примере, когда X1 включен, значение в V2000 будет загружено в аккумулятор командой Load. Выполняется операция логического ИЛИ над значением из аккумулятора и значением из V2006, используя команду Or. Значение младших 16 бит аккумулятора выводится в V2010 командой Out.

Набор на ручном программаторе

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→	C 2	A 0	A 0	A 0	ENT
Q OR	→	SHFT	V AND	C 2	A 0	A 0	G 6	ENT
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT

Or Double (ORD)

Команда Or Double — это 32-битная команда, которая выполняет операцию логического ИЛИ над значением из аккумулятора и константой (Аааа), которая является или двумя последовательными ячейками V-памяти, или 8-ми разрядной (максимум) константой. Результат находится в аккумуляторе. Дискретные флаги состояния указывают, является ли результат команды Or Double нулем или отрицательным числом (старший бит включен).

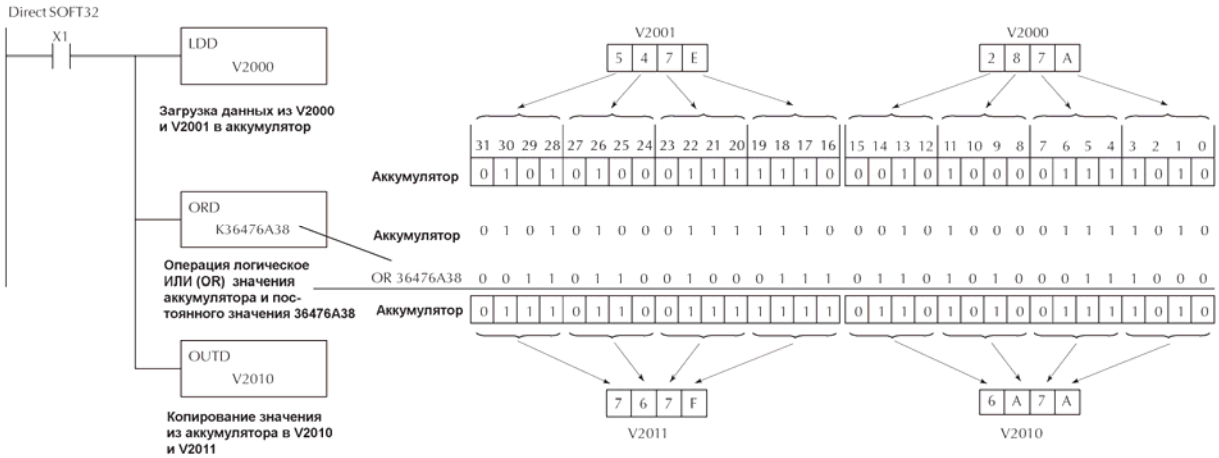


Тип данных операнда		Диапазон DL06
		aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-FFFFFFFF

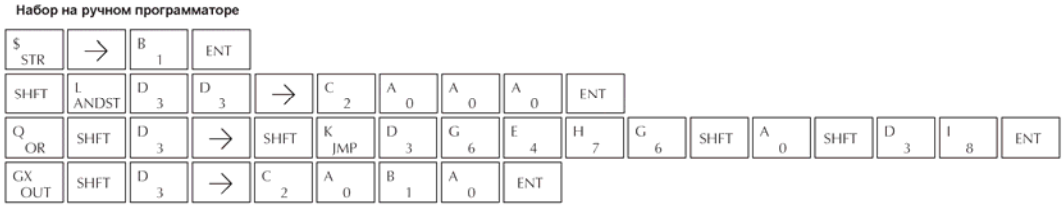
Флаги	Описание
SP63	«1», если результат в аккумуляторе является нулем
SP70	«1», если результат в аккумуляторе является отрицательным

ПРИМЕЧАНИЕ: Флаги состояния доступны только до тех пор, пока не другая команда не использует те же самые флаги

В следующем примере, когда X1 включен, значение в V2000 и V2001 будет загружено в аккумулятор командой Load Double. Выполняется операция логического ИЛИ над значением из аккумулятора и значением 36476A38 используя команду Or Double. Значение из аккумулятора выводится в V2010 и V2011, используя команду Out Double.

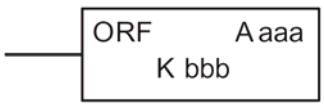


Набор на ручном программаторе



Or Formatted (ORF)

Команда OR Formatted – это команда, которая выполняет операцию логического ИЛИ над двоичным значением аккумулятора и заданным диапазоном бит памяти (1-32). Команда требует значение исходной ячейки (Aaaa) и числа бит (Kbbb), чтобы выполнить сравнение. Дискретные флаги состояния указывают, является ли результат нулем или отрицательным числом (старший бит = 1).

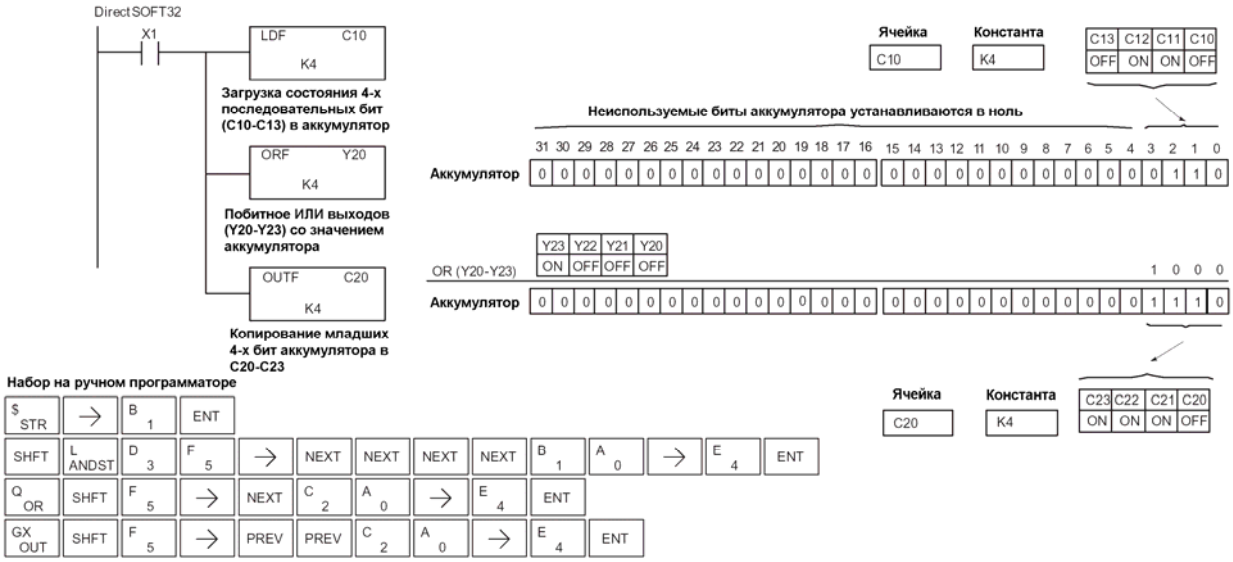


Тип данных операнда		Диапазон DL06	
	A/B	aaa	bbb
Входы	X	0-777	-
Выходы	Y	0-777	-
Реле управления	C	0-1777	-
Бит стадии	S	0-1777	-
Бит таймера	T	0-377	-
Бит счетчика	CT	177	-
Специальное реле	SP	0-777	-
Константа	K	-	1-32

Флаги	Описание
SP63	«1», если результат в аккумуляторе является нулем
SP70	«1», если результат в аккумуляторе является отрицательным

ПРИМЕЧАНИЕ: Флаги состояния доступны только до тех пор, пока не другая команда не использует те же самые флаги

В следующем примере, когда X1 включен, команда Load Formatted загружает C10–C13 (4 бита) в аккумулятор Команда OR Formatted выполняется операцию логического ИЛИ над содержимым аккумулятора и набором бит Y20–Y23. Команда Out Formatted выводит занчение 4 младших бита аккумулятора в биты C20–C23.

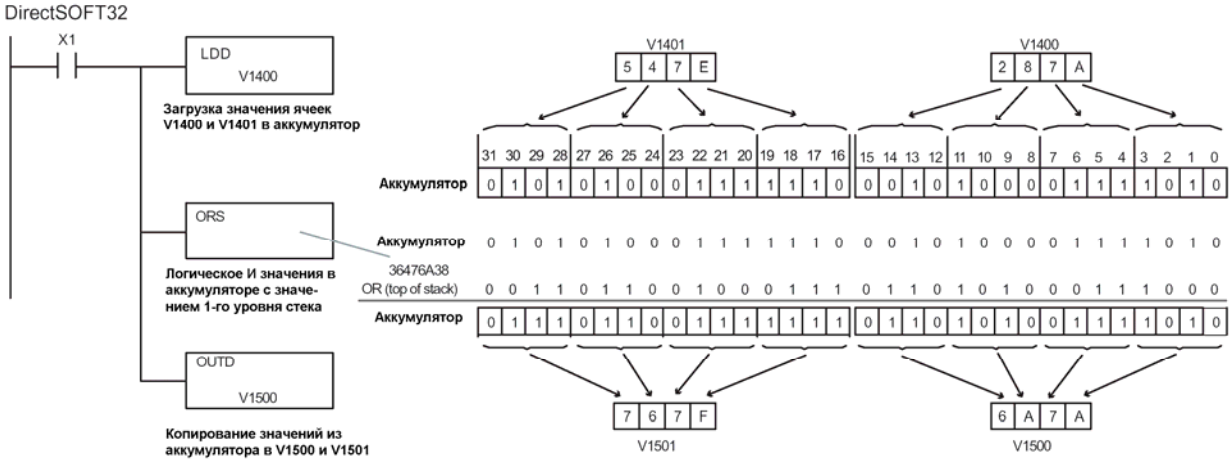


Or with Stack (ORS)

Команда OR with Stack - это 32-битная команда, которая выполняет операцию логического ИЛИ над значением из аккумулятора и значением первого уровня стека аккумулятора. Результат находится в аккумуляторе. Удаляется из стека значение, расположенное на первом уровне стека аккумулятора, и все значения перемещаются на один уровень вверх. Дискретные флаги состояния указывают, является ли результат команды OR with Stack нулем или отрицательным числом (старший бит =1)

Флаги	Описание
SP63	«1», если результат в аккумуляторе является нулем
SP70	«1», если результат в аккумуляторе является отрицательным

В следующем примере, когда X1 включен, будет выполнятся операция логического ИЛИ с двоичным значением в аккумуляторе и двоичным значением на первом уровне стека аккумулятора. Результат находится в аккумуляторе.

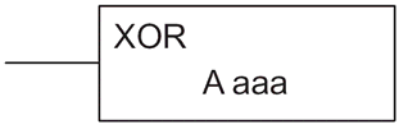


Набор на ручном программаторе

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT
Q OR	SHFT	S RST	ENT						
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT	

Exclusive Or (XOR)

Команда Exclusive Or — это 16-битная команда, которая выполняет операцию исключающего ИЛИ над значением, расположенным в нижних 16 битах аккумулятора, и значением в заданной ячейке V-памяти (Аааа). Результат находится в аккумуляторе. Дискретный флаг состояния указывает, является ли результат команды XOR нулем.



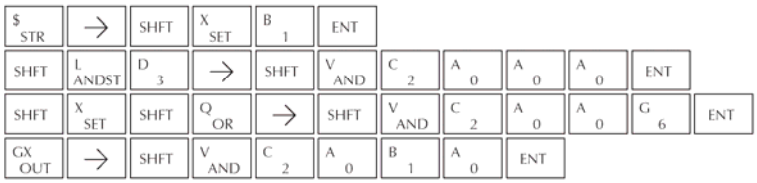
Тип данных операнда		Диапазон DL06
		aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти

Флаги	Описание
SP63	«1», если результат в аккумуляторе является нулем
SP70	«1», если результат в аккумуляторе является отрицательным

ПРИМЕЧАНИЕ: Флаги состояния доступны только до тех пор, пока не другая команда не использует те же самые флаги

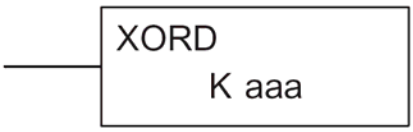
В следующем примере, когда X1 включен, значение в V2000 будет загружено в аккумулятор командой Load. Выполняется операция исключающего ИЛИ над значением из аккумулятора и значением из V2006, используя команду Exclusive Or. Значение младших 16 бит аккумулятора выводится в V2010 командой Out.

Набор на ручном программаторе



Exclusive Or Double (XORD)

Команда Exclusive Or Double — это 32-битная команда, которая выполняет операцию исключающего ИЛИ над значением из аккумулятора и константой (Аааа), которая является или двумя последовательными ячейками V-памяти, или 8-ми разрядной константой (максимум). Результат находится в аккумуляторе. Дискретные флаги состояния указывают, является ли результат команды Exclusive Or Double нулем или отрицательным числом (старший бит включен).

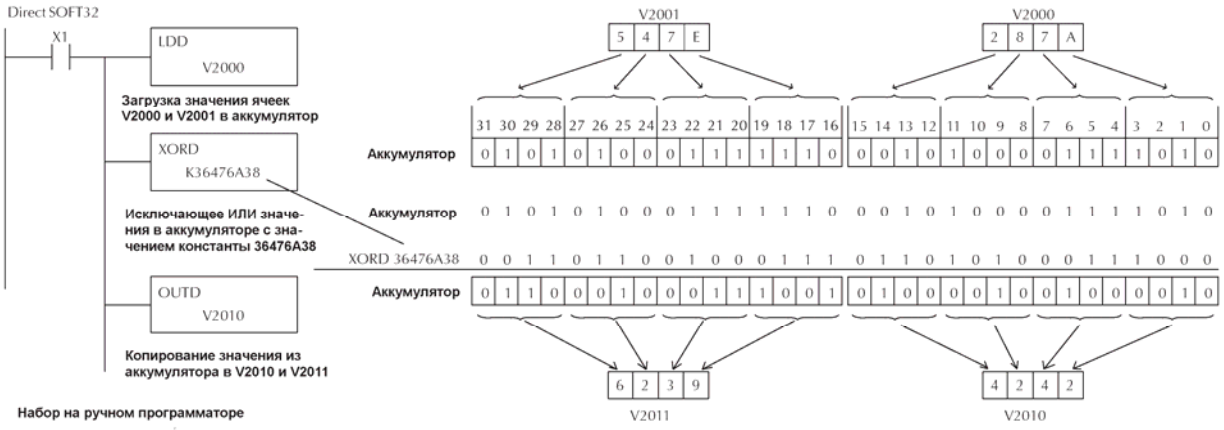


Тип данных операнда		Диапазон DL06
	A	aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-FFFFFFFF

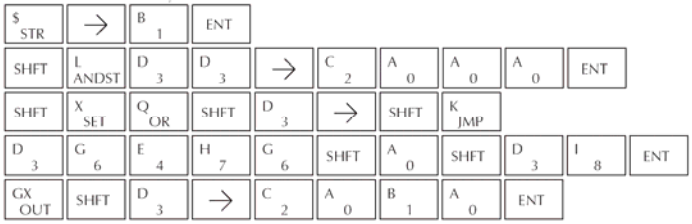
Флаги	Описание
SP63	«1», если результат в аккумуляторе является нулем
SP70	«1», если результат в аккумуляторе является отрицательным

ПРИМЕЧАНИЕ: Флаги состояния доступны только до тех пор, пока не другая команда не использует те же самые флаги

В следующем примере, когда X1 включен, значение в V2000 и V2001 будет загружено в аккумулятор командой Load Double. Выполняется операция исключающего ИЛИ над значением из аккумулятора и значением 36476A38, используя команду Exclusive Or Double. Значение из аккумулятора выводится в V2010 и V2011, используя команду Out Double.



Набор на ручном программаторе



Exclusive Or Formatted (XORF)

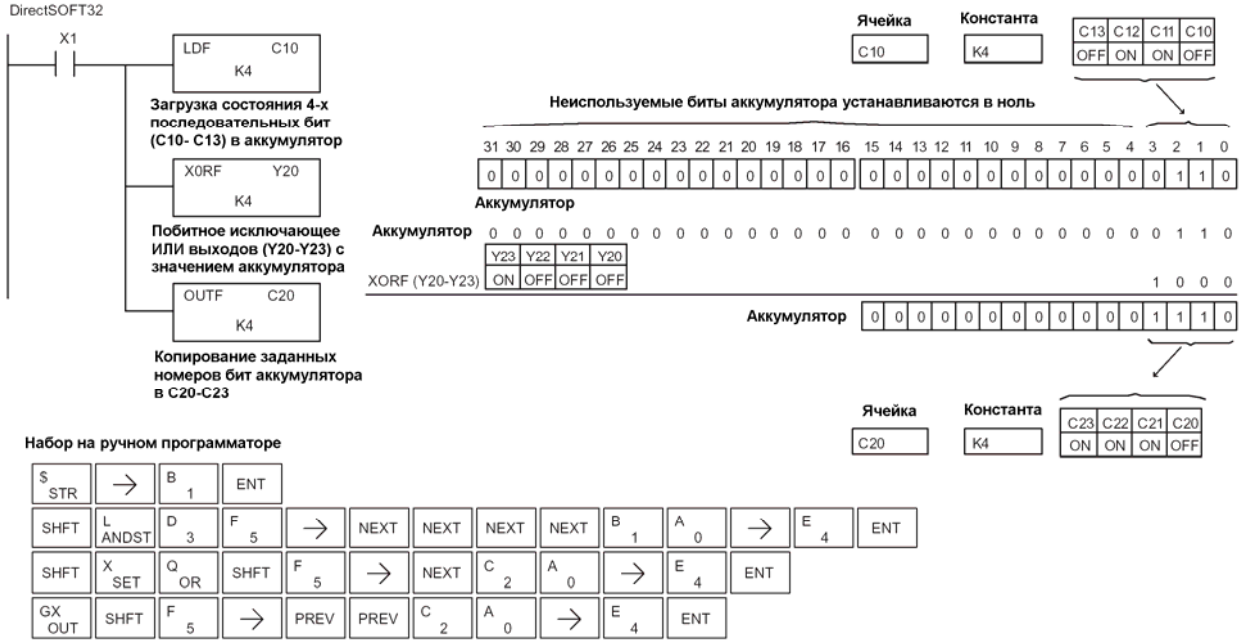
Команда Exclusive OR Formatted – это команда, которая выполняет операцию исключающего ИЛИ над двоичным значением аккумулятора и заданным диапазоном бит дискретной памяти (1-32). Команда требует значение адреса исходной ячейки (Aaaa) и числа бит (Kbbb), чтобы выполнить исключающее ИЛИ. Дискретные флаги состояния указывают, является ли результат команды Exclusive OR Formatted нулем или отрицательным числом (старший бит=1).

Тип данных операнда		Диапазон DL06	
	A/B	aaa	bbb
Входы	X	0-777	-
Выходы	Y	0-777	-
Реле управления	C	0-1777	-
Бит стадии	S	0-1777	-
Бит таймера	T	0-377	-
Бит счетчика	CT	177	-
Специальное реле	SP	0-777	-
Константа	K	-	1-32

Флаги	Описание
SP63	«1», если результат в аккумуляторе является нулем
SP70	«1», если результат в аккумуляторе является отрицательным

ПРИМЕЧАНИЕ: Флаги состояния доступны только до тех пор, пока не другая команда не использует те же самые флаги

В следующем примере, когда X1 включен, конфигурация битов C10–C13 (4 бита) будет загружена в аккумулятор, используя команду Load Formatted. Выполняется логическая операция исключающего ИЛИ над содержимым аккумулятора и набором битов (bit pattern) из Y20–Y23, используя команду Exclusive OR Formatted. Значение младших 4 бит аккумулятора выводится в C20-C23, используя команду Out Formatted.



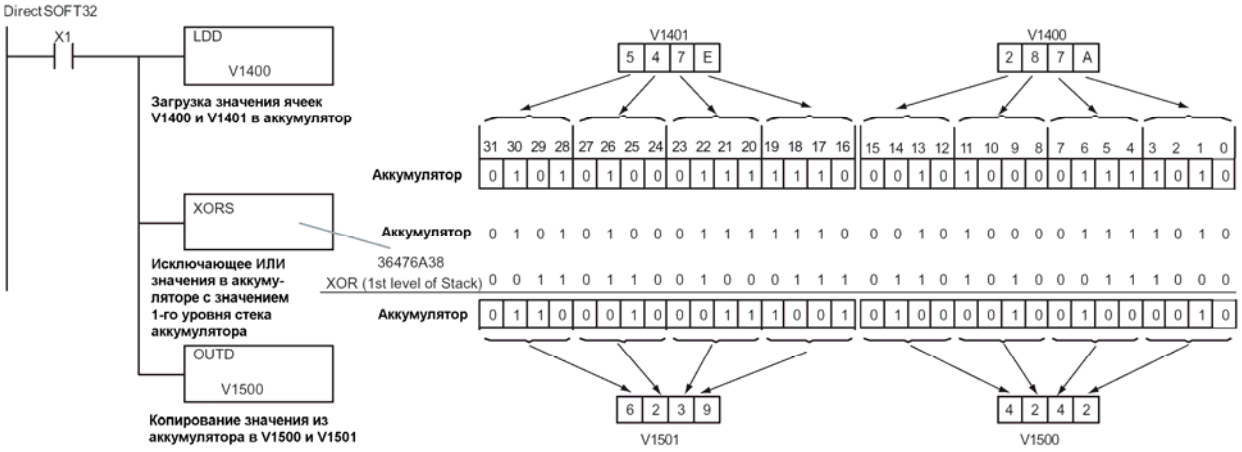
Exclusive Or with Stack (XORS)

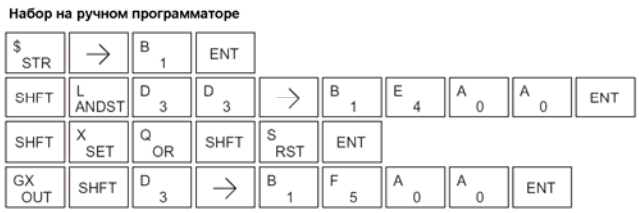
Команда Exclusive OR with Stack - это 32-битная команда, которая выполняет операцию исключающего ИЛИ над значением из аккумулятора и значением первого уровня стека аккумулятора. Результат находится в аккумуляторе. Из стека удаляется значение, расположенное на первом уровне стека аккумулятора, и все значения перемещаются на один уровень вверх. Дискретные флаги состояния указывают, является ли результат команды Exclusive OR with Stack нулем или отрицательным числом (старший бит =1)

ПРИМЕЧАНИЕ: Флаги состояния доступны только до тех пор, пока не другая команда не использует те же самые флаги

Флаги	Описание
SP63	«1», если результат в аккумуляторе является нулем
SP70	«1», если результат в аккумуляторе является отрицательным

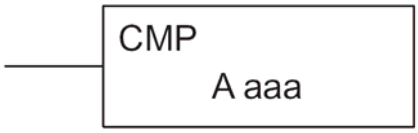
В следующем примере, когда X1 включен, выполняется операция исключающего ИЛИ над двоичным значением аккумулятора и двоичным значением первого уровня стека аккумулятора. Результат находится в аккумуляторе.





Compare (CMP)

Команда Compare — это 16-битная команда, которая сравнивает значение из младших 16 бит аккумулятора со значением в определенной ячейке V-памяти (Aaaa). Соответствующий флаг состояния будет включен, показывая результат сравнения.



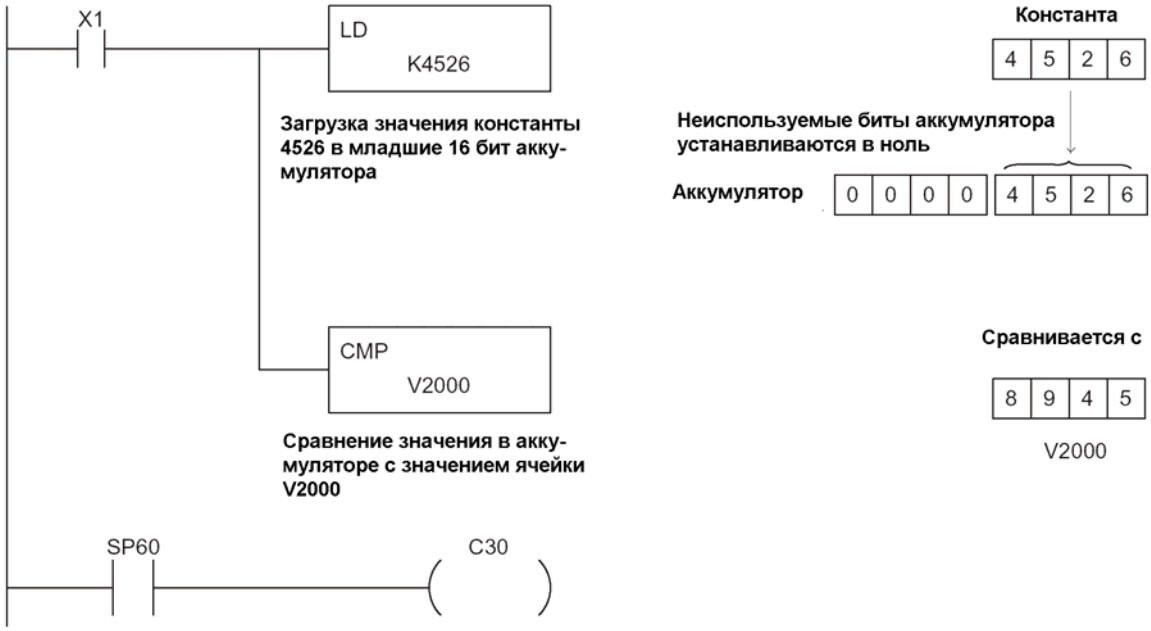
Тип данных операнда		Диапазон DL06
	A	aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти

Флаги	Описание
SP60	«1», когда значение в аккумуляторе меньше чем значение в команде
SP61	«1», когда значение в аккумуляторе равно значению в команде
SP62	«1», когда значение в аккумуляторе больше чем значение в команде

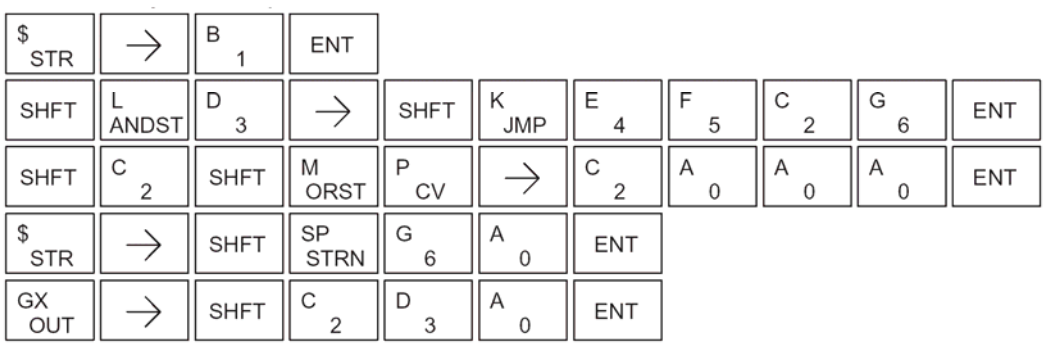
ПРИМЕЧАНИЕ: Флаги состояния доступны только до тех пор, пока не другая команда не использует те же самые флаги

В следующем примере, когда X1 включен, константа 4526 будет загружена в младшие 16 бит аккумулятора, командой Load. Значение из аккумулятора сравнивается с значением в V2000, используя команду Compare. Соответствующий дискретный флаг состояния «1», показывая результат сравнения. В этом примере, если значение в аккумуляторе меньше, чем значение, определенное в команде Compare, то SP60 включит C30.

Direct SOFT32

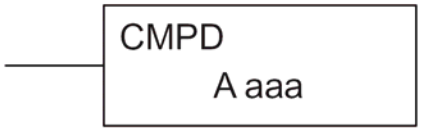


Набор на ручном программаторе



Compare Double (CMPD)

Команда Compare Double — это 32-битная команда, которая сравнивает значение из аккумулятора со значением (Aaaa), которое является или двумя последовательными ячейками V-памяти или 8-ми разрядной (максимум) константой. Соответствующий флаг состояния будет включен, показывая результат сравнения.

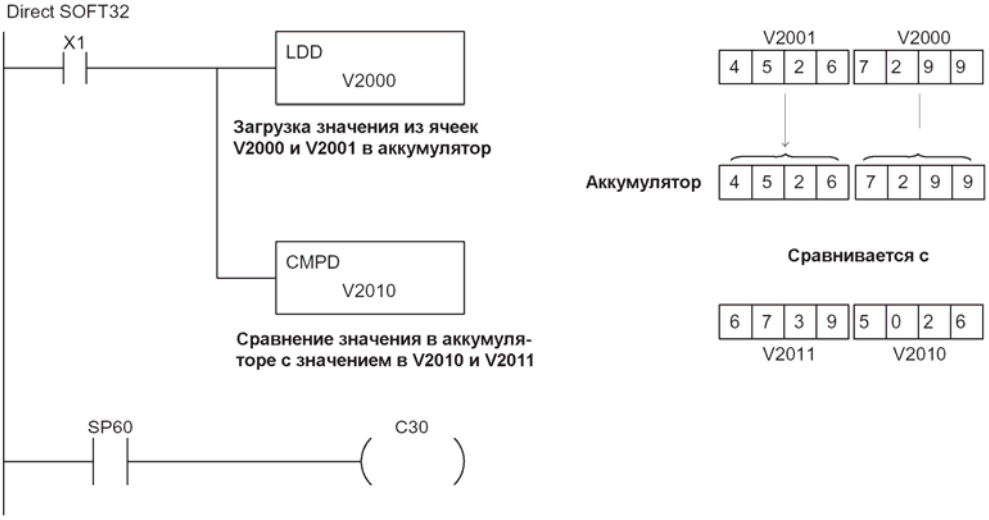


Тип данных операнда		Диапазон DL06
	A	aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-FFFFFFFF

Флаги	Описание
SP60	«1», когда значение в аккумуляторе меньше чем значение в команде
SP61	«1», когда значение в аккумуляторе равно значению в команде
SP62	«1», когда значение в аккумуляторе больше чем значение в команде

ПРИМЕЧАНИЕ: Флаги состояния доступны только до тех пор, пока не другая команда не использует те же самые флаги

В следующем примере, когда X1 включен, значение из V2000 и V2001 будет загружено в аккумулятор, используя команду Load Double. Значение из аккумулятора сравнивается с значением в V2010 и V2011, используя команду CMPD. Соответствующий дискретный флаг состояния «1», показывая результат сравнения. В этом примере, если значение в аккумуляторе меньше, чем значение, определенное в команде Compare, то SP60 включит C30.

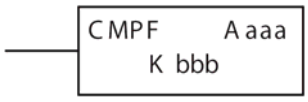


Набор на ручном программаторе

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	C 2	SHFT	M ORST	P CV	D 3	→	C 2	A 0	B 1	A 0	ENT
$ STR	→	SHFT	SP STRN	G 6	A 0	ENT					
GX OUT	→	SHFT	C 2	D 3	A 0	ENT					

Compare Formatted (CMPF)

Команда Compare Formatted сравнивает значение в аккумуляторе с заданным числом дискретных ячеек (1-32). Команда требует исходной ячейки (Aaaa) и числа бит (Kbbb), чтобы сравнивать. Соответствующий флаг состояния будет включен, указывая результат сравнения.

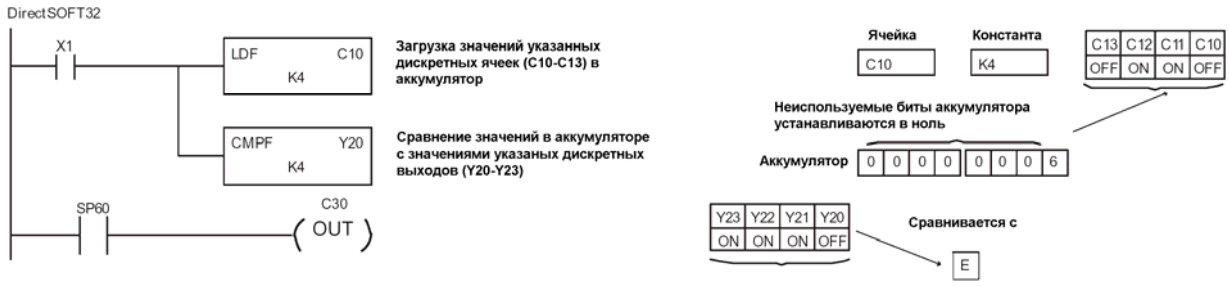


Тип данных операнда		Диапазон DL06	
	A/B	aaa	bbb
Входы	X	0-777	-
Выходы	Y	0-777	-
Реле управления	C	0-1777	-
Бит стадии	S	0-1777	-
Бит таймера	T	0-377	-
Бит счетчика	CT	177	-
Специальное реле	SP	0-777	-
Константа	K	-	1-32

Флаги	Описание
SP60	«1», когда значение в аккумуляторе меньше чем значение в команде
SP61	«1», когда значение в аккумуляторе равно значению в команде
SP62	«1», когда значение в аккумуляторе больше чем значение в команде

ПРИМЕЧАНИЕ: Флаги состояния доступны только до тех пор, пока не другая команда не использует те же самые флаги

В следующем примере, когда X1 включен, двоичное значение (6) из C10–C13 будет загружено в аккумулятор, используя команду Load Formatted. Команда CMPF сравнивает значение в аккумуляторе и значение в Y20-Y23 (шестнадцатеричное число E). Соответствующий дискретный флаг состояния будет включен, показывая результат сравнения. В этом примере, если значение в аккумуляторе – меньше, чем значение, определенное в команде Compare, SP60 включит C30.



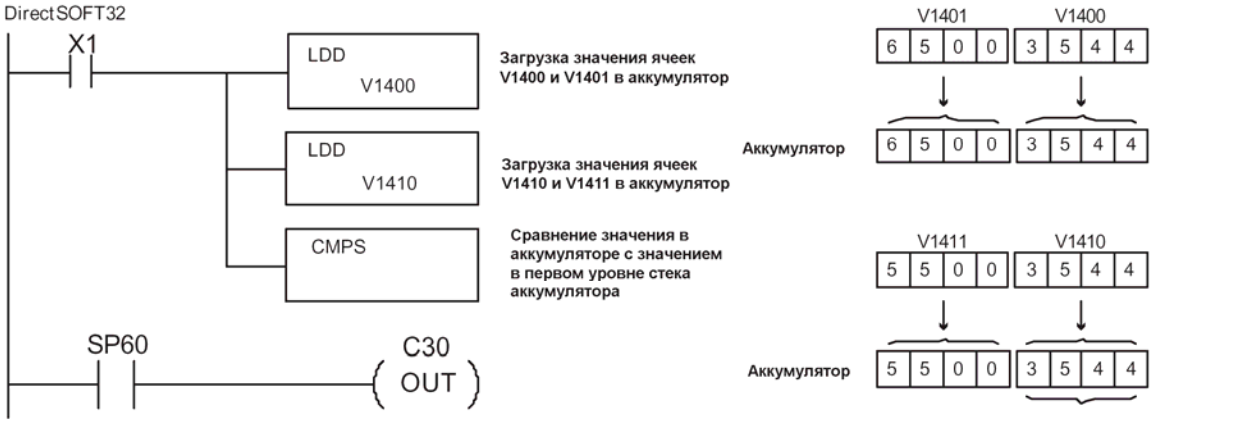
Compare with Stack (CMPS)

Команда Compare with Stack - 32-битная команда, которая сравнивает значение в аккумуляторе со значением в первом уровне стека аккумулятора.
Соответствующий флаг состояния будет включен, указывая результат сравнения. Команда не влияет на значение в аккумуляторе.

Флаги	Описание
SP60	«1», когда значение в аккумуляторе меньше чем значение в команде
SP61	«1», когда значение в аккумуляторе равно значению в команде
SP62	«1», когда значение в аккумуляторе больше чем значение в команде

ПРИМЕЧАНИЕ: Флаги состояния доступны только до тех пор, пока не будет выполнена другая команда, использующая те же самые флаги

В следующем примере, когда X1 включен, значение из V1400 и V1401 загружено в аккумулятор, используя команду Load Double. Значение из V1410 и V1411 загружено в аккумулятор, используя команду Load Double. Значение, которое было загружено в аккумулятор из V1400 и V1401, было помещено на вершину стека, когда была выполнена вторая команда Load. Значение в аккумуляторе сравнивается с значением в первом уровне стека аккумулятора, используя команду CMPS. Соответствующий дискретный флаг состояния будет включен, показывая результат сравнения. В этом примере, если значение в аккумуляторе меньше, чем значение в стеке, SP60 включит C30

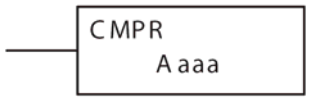


Набор на ручном программаторе

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	B 1	A 0	ENT
SHFT	C 2	SHFT	M ORST	P CV	S RST	ENT			
$ STR	PREV	G 6	A 0	ENT					
GX OUT	→	NEXT	NEXT	NEXT	SHFT	C 2	D 3	A 0	ENT

Compare Real Number (CMPR)

Команда Compare Real Number сравнивает значение вещественного числа в аккумуляторе с двумя последовательными ячейками V-памяти,. Соответствующий флаг состояния будет включен, указывая результат сравнения. Оба сравниваемых числа - длиной 32 бита.

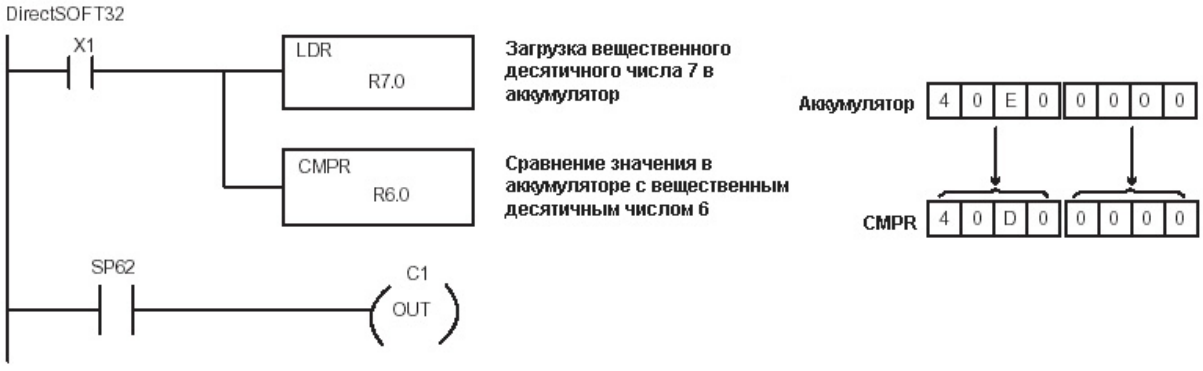


Тип данных операнда		Диапазон
	A	aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	R	-3.402823E+038 - +3.402823E+038

Флаги	Описание
SP60	«1», когда значение в аккумуляторе меньше чем значение в команде
SP61	«1», когда значение в аккумуляторе равно значению в команде
SP62	«1», когда значение в аккумуляторе больше чем значение в команде
SP71	«1» в любое время, когда определено, что указатель ссылается на недопустимую ячейку V-памяти
SP75	«1», когда ожидается BCD число, а приходит число в другом формате.

ПРИМЕЧАНИЕ: Флаги состояния доступны только до тех пор, пока не другая команда не использует те же самые флаги

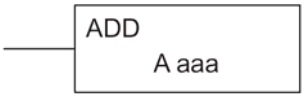
В следующем примере, когда X1 включен, команда LDR загружает вещественное десятичное число 7 в аккумулятор. Команда CMPR сравнивает значение в аккумуляторе с вещественным десятичным числом 6. Поскольку 7 > 6, то соответствующий дискретный флаг включается (специальное реле SP62), включая управляющее реле C1.



Математические команды

Add (ADD)

Add — 16-битная команда, которая складывает BCD значение в аккумуляторе с BCD значением в ячейке V-памяти (Аааа).(Вы не можете использовать константу как параметр). Результат находится в аккумуляторе.



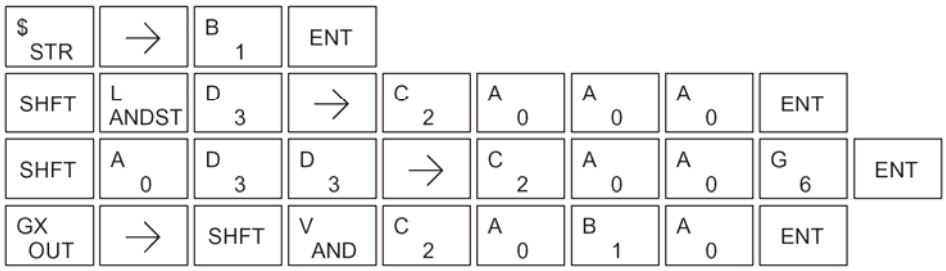
Тип данных операнда		Диапазон DL06
	A	aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP66	«1», когда 16-битовая команда сложения приводит к переносу
SP67	«1», когда 32-битовая команда сложения приводит к переносу
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD числом

ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

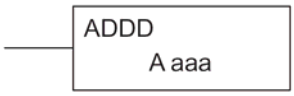
В следующем примере, когда X1 включен, значение в V2000 будет загружено в аккумулятор командой Load. Значение в младших 16 битах аккумулятора складывается со значением в V2006 командой Add. Значение в аккумуляторе копируется в V2010 командой Out.

Набор на ручном программаторе



Add Double (ADDD)

Add Double — 32-битная команда, которая складывает BCD значение в аккумуляторе с BCD значением (Aaaa), которое является или двумя последовательными ячейками V-памяти или BCD константой (8 знаков максимум). Результат находится в аккумуляторе.

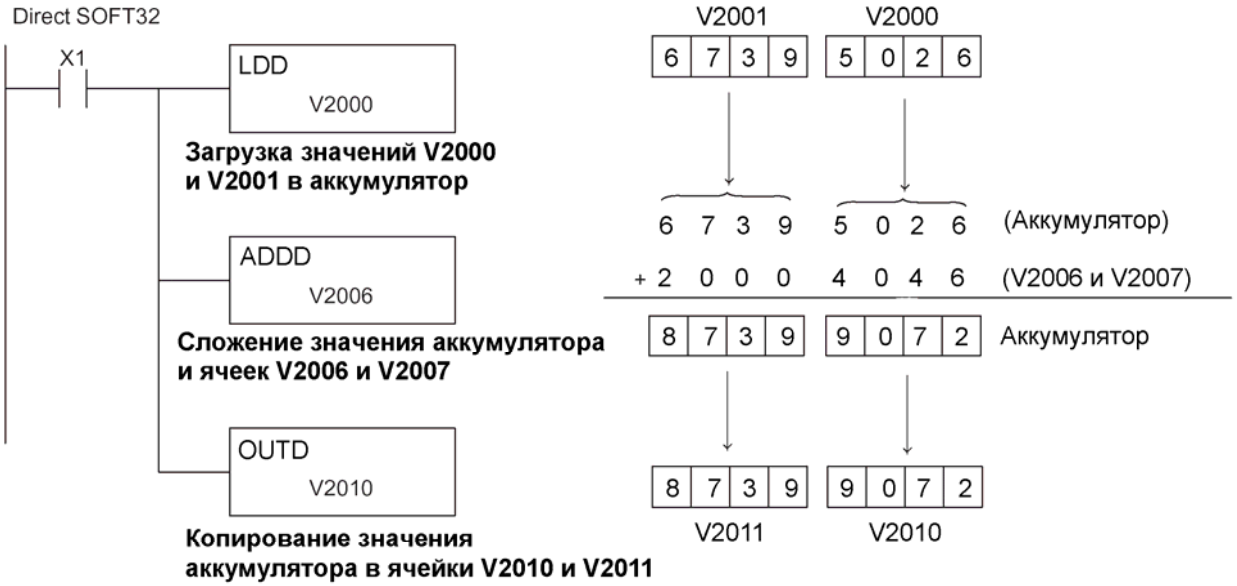


Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-99999999

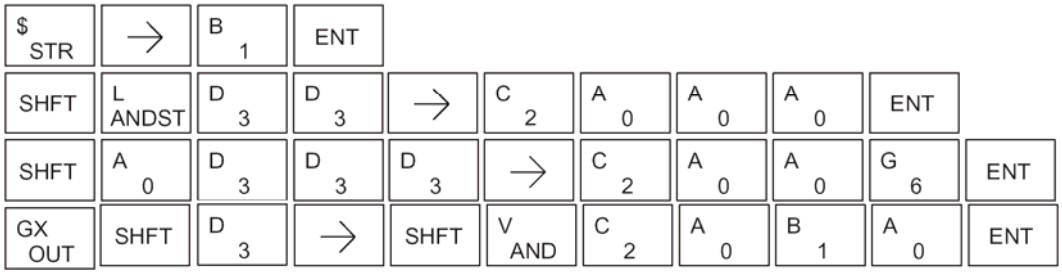
Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP66	«1», когда 16-битовая команда сложения приводит к переносу
SP67	«1», когда 32-битовая команда сложения приводит к переносу
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD 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 V2000 и V2001 будет загружено в аккумулятор командой Load Double. Значение в аккумуляторе складывается со значением в V2006 и V2007 командой Add Double. Значение в аккумуляторе копируется в V2010 и V2011 командой Out Double.

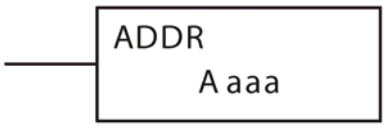


Набор на ручном программаторе



Add Real (ADDR)

Команда Add Real складывает вещественное число в аккумуляторе с вещественной константой или вещественным значением в двух ячейках последовательных V-памяти. Результат записывается в аккумулятор. Оба числа должны быть представлены в формате вещественного числа IEEE с плавающей точкой.

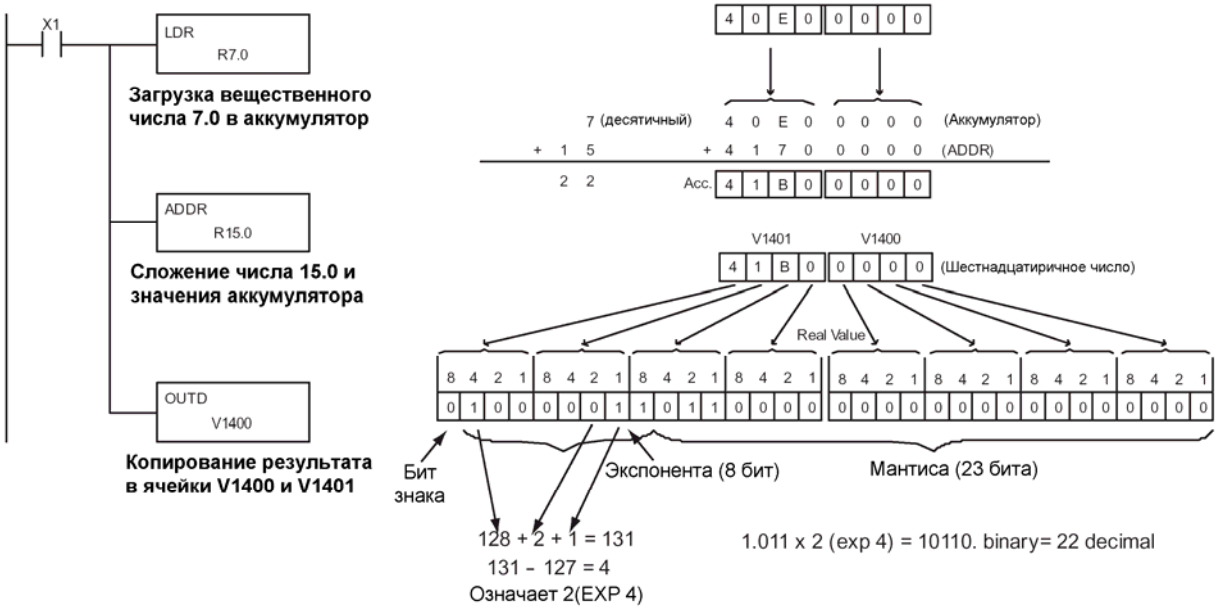


Тип данных операнда		Диапазон
	A	aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	R	-3.402823E+038 - +3.402823E+038

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP71	«1» в любое время, когда определено, что указатель ссылается на недопустимую ячейку V-памяти
SP72	«1» в любое время, когда значение в аккумуляторе - недопустимое число с плавающей запятой
SP73	«1», когда сложение или вычитание со знаком приводит к некорректному биту знака.
SP74	«1» в любое время, когда математическая операция с плавающей запятой приводит к ошибке переполнения.
SP75	«1», когда ожидается BCD число, а приходит число в другом формате.

ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

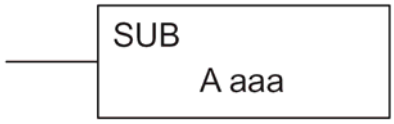
DirectSOFT32



ПРИМЕЧАНИЕ. Текущая версия ручного программатора не поддерживает ввод вещественных чисел с автоматическим преобразованием к 32-разрядному формату IEEE. Необходимо использовать DirectSOFT32.

Subtract (SUB)

Subtract — 16-битная команда, которая вычитает BCD значение (Aaaa) в ячейке V-памяти из BCD значения в младших 16 битах аккумулятора. Результат находится в аккумуляторе.

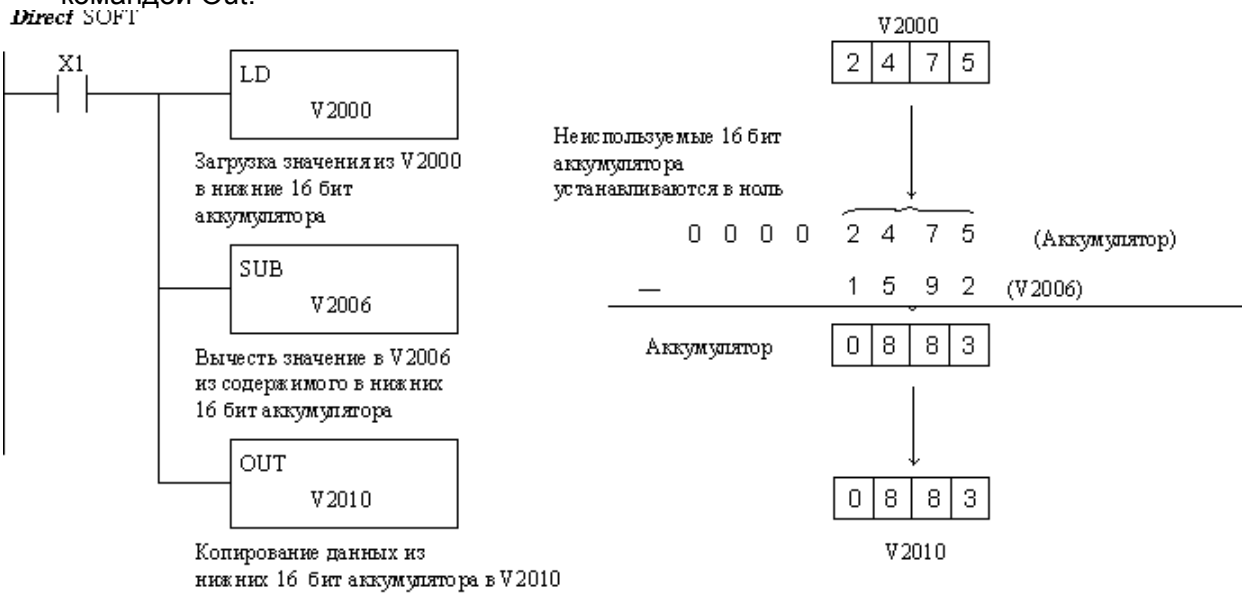


Тип данных операнда		Диапазон DL06
	A	aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти

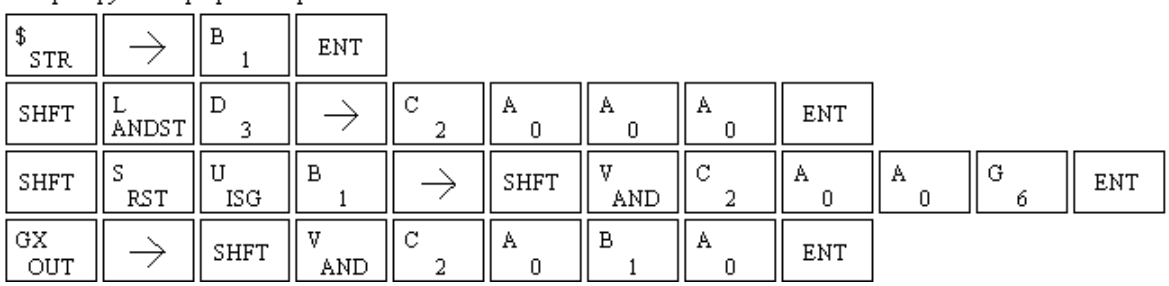
Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP64	«1», когда 16-битовая команда вычитания приводит к заимствованию
SP65	«1», когда 32-битовая команда вычитания приводит к заимствованию
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD 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 V2000 будет загружено в аккумулятор командой Load. Значение в V2006 вычитается из значения в аккумуляторе командой Subtract. Значение в аккумуляторе копируется в V2010 командой Out.

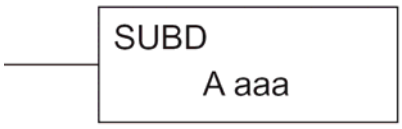


Набор на ручном программаторе



Subtract Double (SUBD)

Subtract Double — 32-битная команда, которая вычитает BCD значение (Aaaa), которое является или двумя последовательными ячейками V-памяти или константой (8 знаков максимум), из BCD значения в аккумуляторе. Результат находится в аккумуляторе.

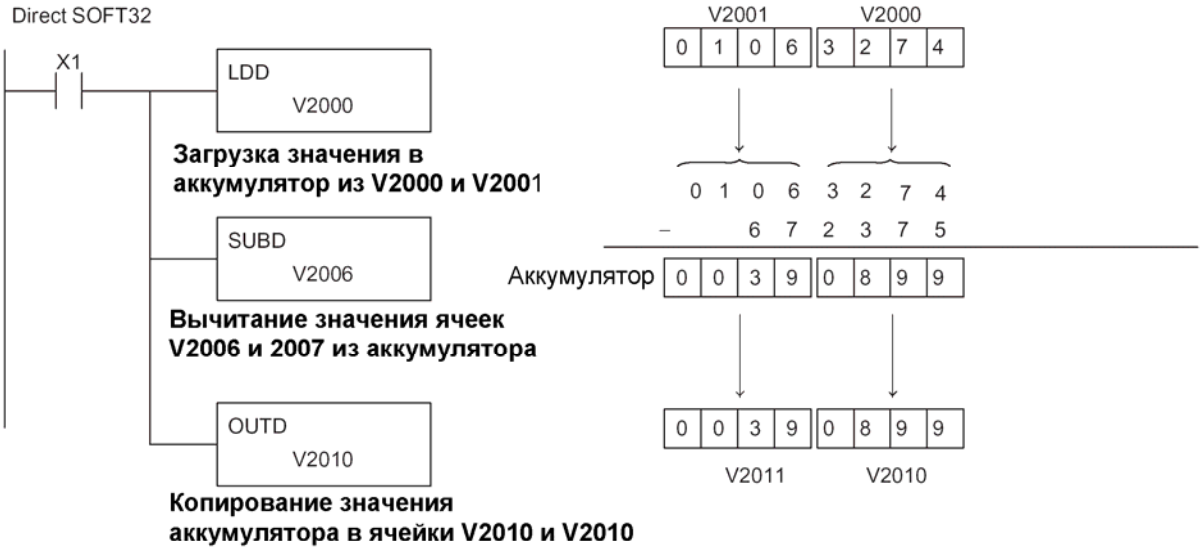


Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-99999999

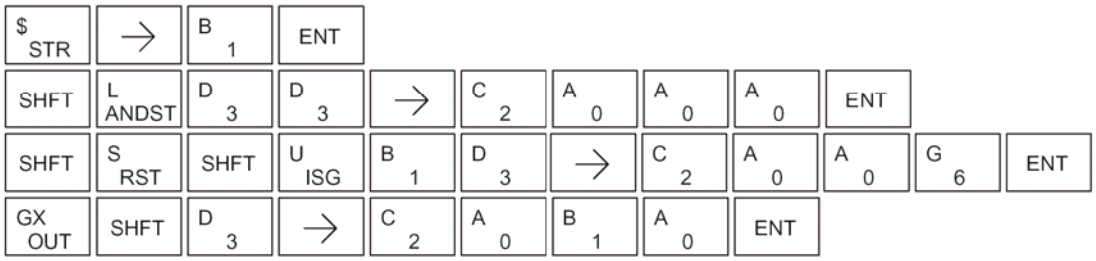
Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP64	«1», когда 16-битовая команда вычитания приводит к заимствованию
SP65	«1», когда 32-битовая команда вычитания приводит к заимствованию
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD 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 V2000 и V2001 будет загружено в аккумулятор командой Load Double. Значение в V2006 и V2007 вычитается из значения в аккумуляторе. Значение в аккумуляторе копируется в V2010 и V2011 командой Out Double.

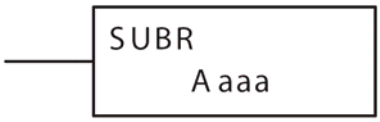


Набор на ручном программаторе



Subtract Real (SUBR)

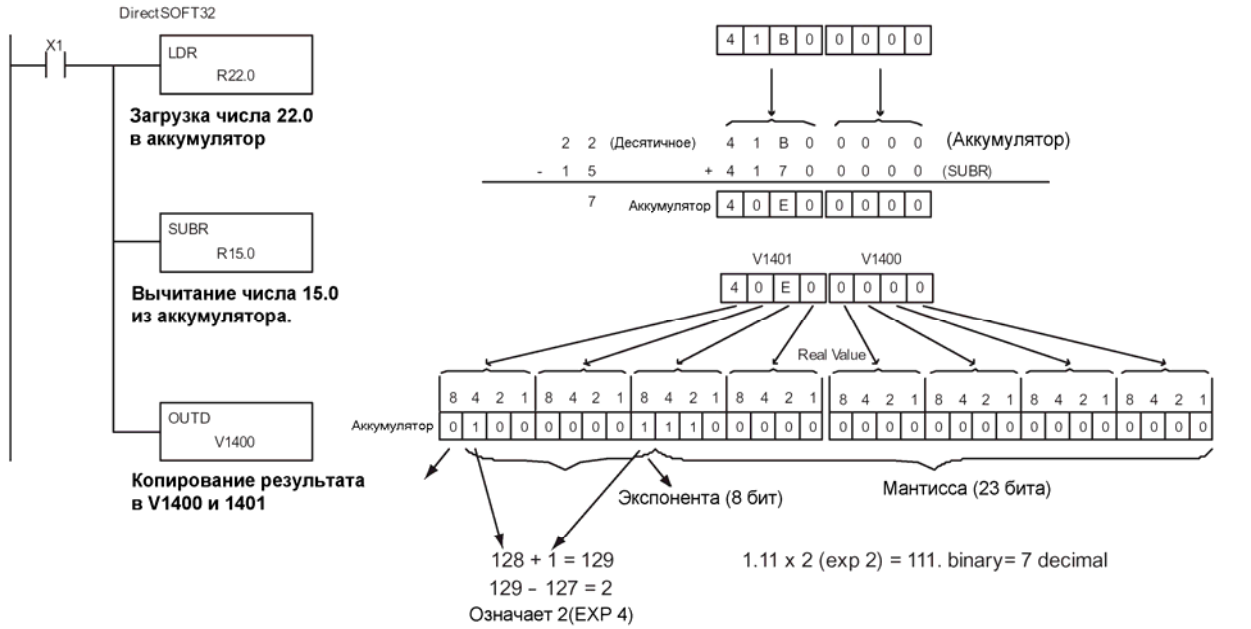
Команда Subtract Real вычитает из аккумулятора вещественное число, которое является или константой, или двумя последовательными ячейками V-памяти. Результат находится в аккумуляторе. Оба числа должны быть в формате IEEE, с плавающей запятой.



Тип данных операнда		Диапазон
	A	aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	R	-3.402823E+038 - +3.402823E+038

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP71	«1» в любое время, когда определено, что указатель ссылается на недопустимую ячейку V-памяти
SP72	«1» в любое время, когда значение в аккумуляторе - недопустимое число с плавающей запятой
SP73	«1», когда сложение или вычитание со знаком приводит к некорректному биту знака.
SP74	«1» в любое время, когда математическая операция с плавающей запятой приводит к ошибке переполнения.
SP75	«1», когда ожидается BCD число, а приходит число в другом формате.

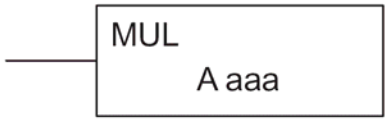
ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.



ПРИМЕЧАНИЕ. Текущая версия ручного программатора не поддерживает ввод вещественных чисел с автоматическим преобразованием к 32-разрядному формату IEEE. Необходимо использовать DirectSOFT32.

Multiply (MUL)

Multiply — 16-битная команда, которая умножает BCD значение (Аааа), которое является или ячейкой V-памяти или константой (4 знака максимум), на BCD значение в нижних 16 битах аккумулятора. Результат может иметь до 8 знаков и находится в аккумуляторе.

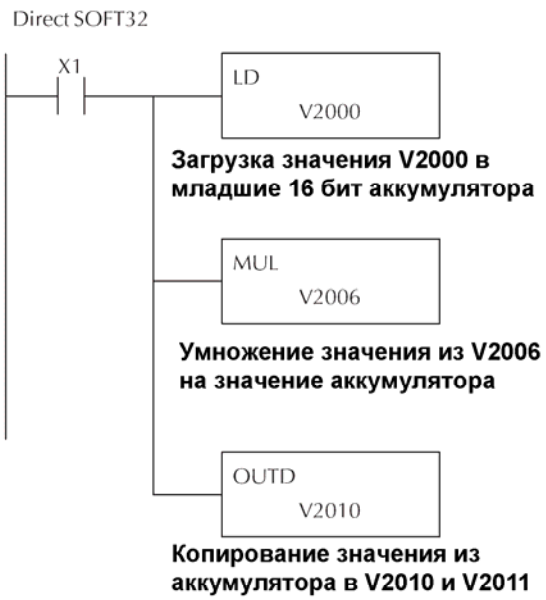


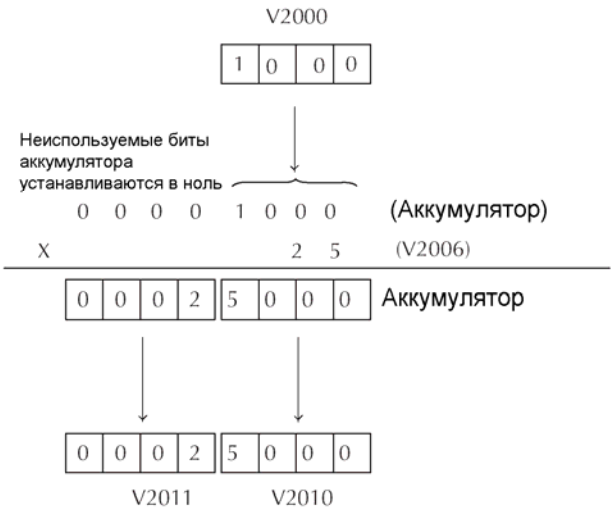
Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-9999

Флаги	**Описание**
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD числом

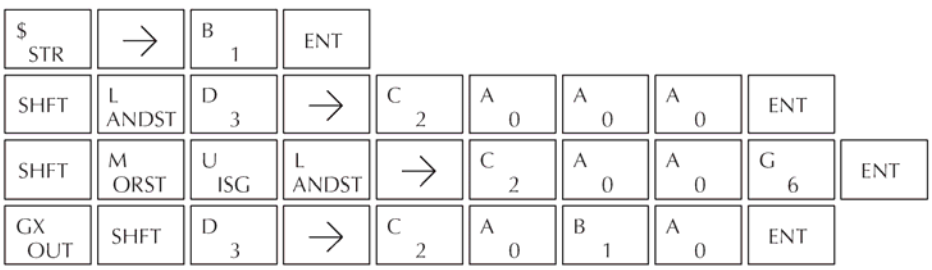
ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X1 включен, значение в V2000 будет загружено в аккумулятор командой Load. Значение в V2006 умножается на значение в аккумуляторе. Значение в аккумуляторе копируется в V2010 и V2011 командой Out Double.



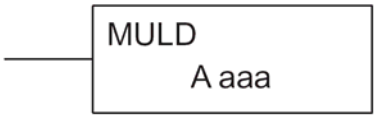


Набор на ручном программаторе



Multiply Double (MULD)

Multiply Double — 32-битная команда, которая умножает 8-знаковое BCD значение в аккумуляторе на 8-знаковое BCD значение в двух последовательных ячейках V-памяти, указанных в команде. Нижние 8 знаков результата находятся в аккумуляторе. Верхние знаки результата хранятся в стеке аккумулятора.



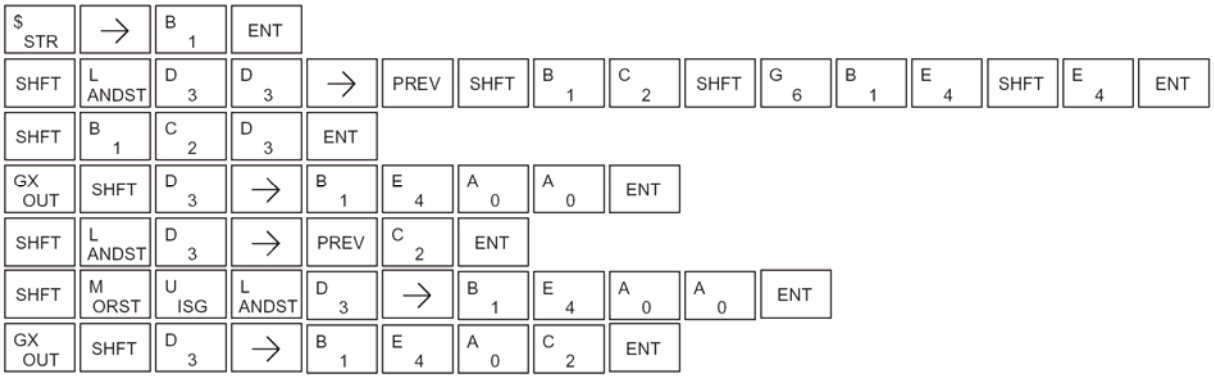
Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD числом

ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

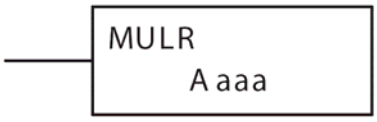
В следующем примере, когда X1 включен, шестнадцатиричная константа Kbc614e будет загружена в аккумулятор. Это число, преобразованное в BCD формат, равно «12345678». Эти числа хранятся в V1400 и V1401. После загрузки константы K2 в аккумулятор, мы умножаем ее на 12345678 и получаем 24691356.

Набор на ручном программаторе



Multiply Real (MULR)

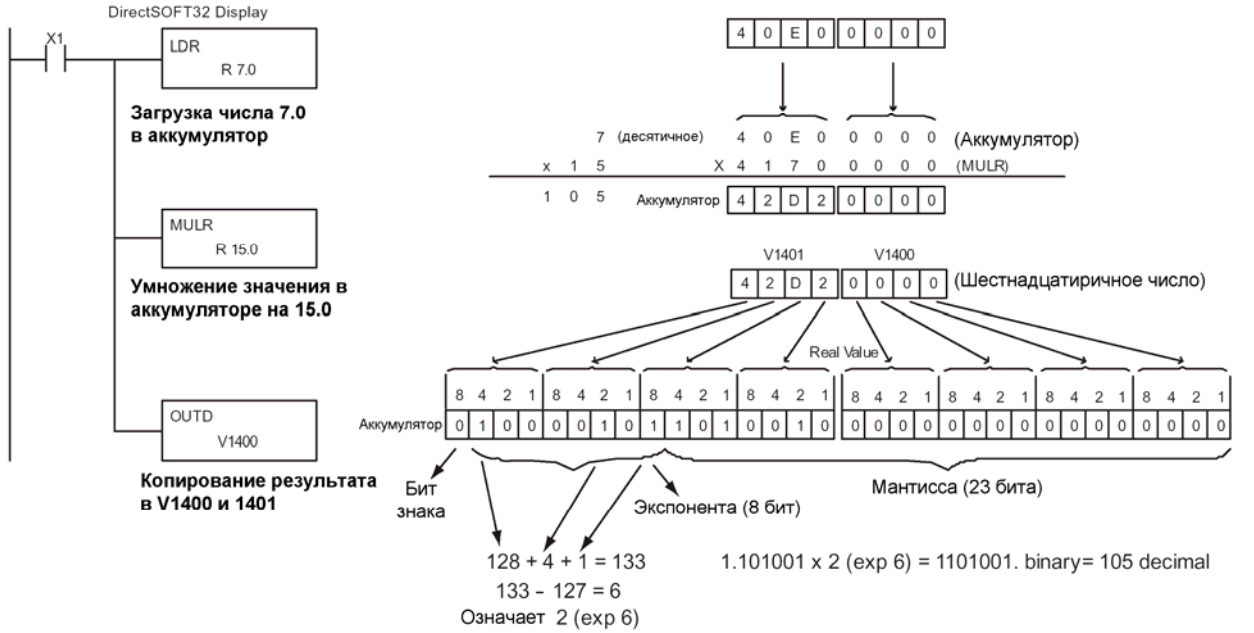
Команда Multiply Real умножает вещественное число в аккумуляторе на вещественную константу или число, находящееся в двух последовательных ячейках V-памяти. Результат сохраняется в аккумуляторе. Оба числа должны быть представлены в формате IEEE как числа с плавающей запятой.



Тип данных операнда		Диапазон
	A	aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	R	-3.402823E+038 -- +3.402823E+038

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP71	«1» в любое время, когда определено, что указатель ссылается на недопустимую ячейку V-памяти
SP72	«1» в любое время, когда значение в аккумуляторе - недопустимое число с плавающей запятой
SP73	«1», когда сложение или вычитание со знаком приводит к некорректному биту знака.
SP74	«1» в любое время, когда математическая операция с плавающей запятой приводит к ошибке переполнения.
SP75	«1», когда ожидается BCD число, а приходит число в другом формате.

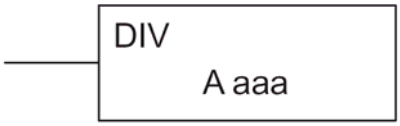
ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.



ПРИМЕЧАНИЕ. Текущая версия ручного программатора не поддерживает ввод вещественных чисел с автоматическим преобразованием к 32-разрядному формату IEEE. Необходимо использовать DirectSOFT32.

Divide (DIV)

Divide — 16-битная команда, которая делит BCD значение в аккумуляторе на BCD значение (Aaaa), которое является или ячейкой V-памяти или константой (4 знака максимум). Целая часть результата находится в аккумуляторе, а остаток — в первой ячейке стека.

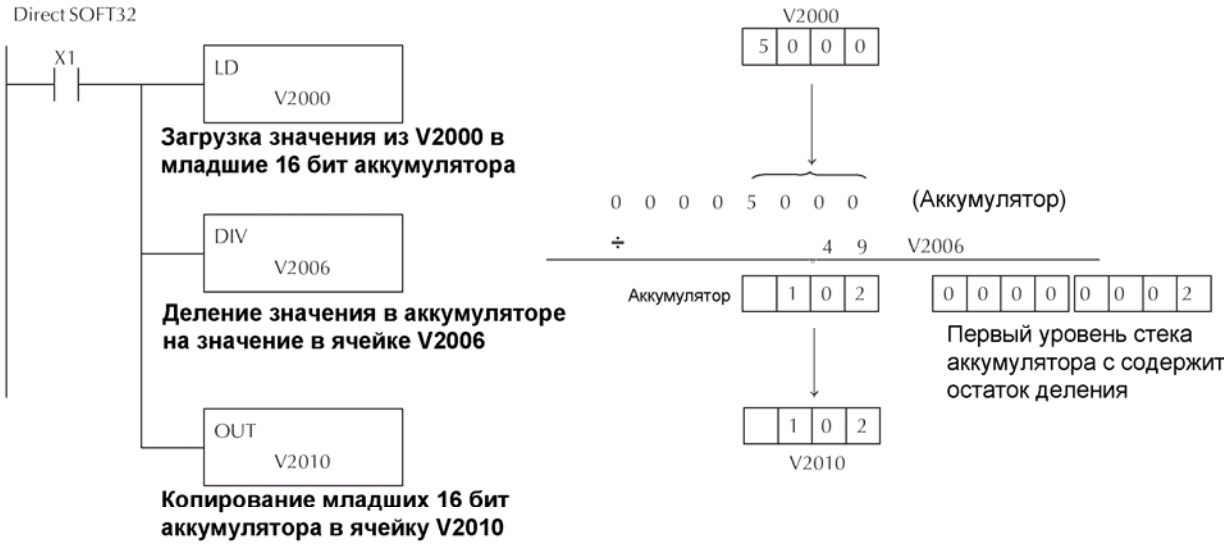


Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-9999

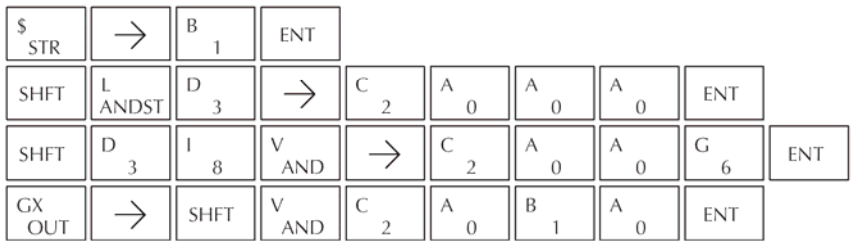
Флаги	Описание
SP53	«1», когда значение операнда больше, чем значение, с которым может работать аккумулятор
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD 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 V2000 будет загружено в аккумулятор командой Load. Значение в аккумуляторе будет поделено на значение в V2006 командой Divide. Значение в аккумуляторе копируется в V2010 командой Out.

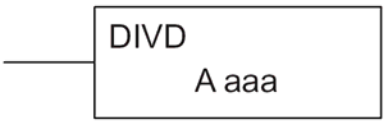


Набор на ручном программаторе



Divide Double (DIVD)

Divide Double — 32-битная команда, которая делит BCD значение в аккумуляторе на BCD 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.

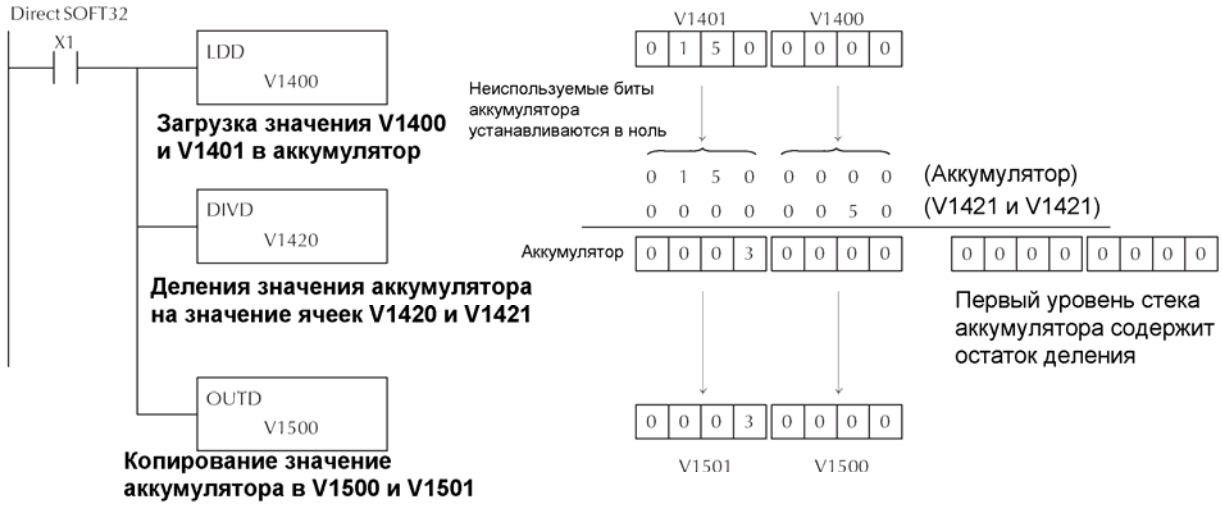


Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти

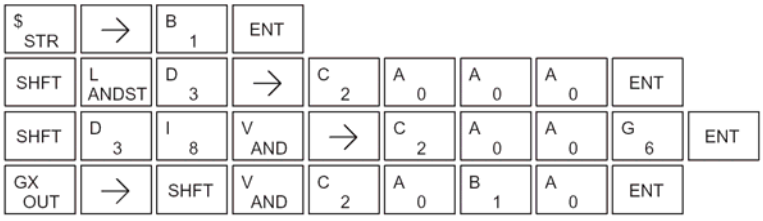
Флаги	Описание
SP53	«1», когда значение операнда больше, чем значение, с которым может работать аккумулятор
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD 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 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 программаторет



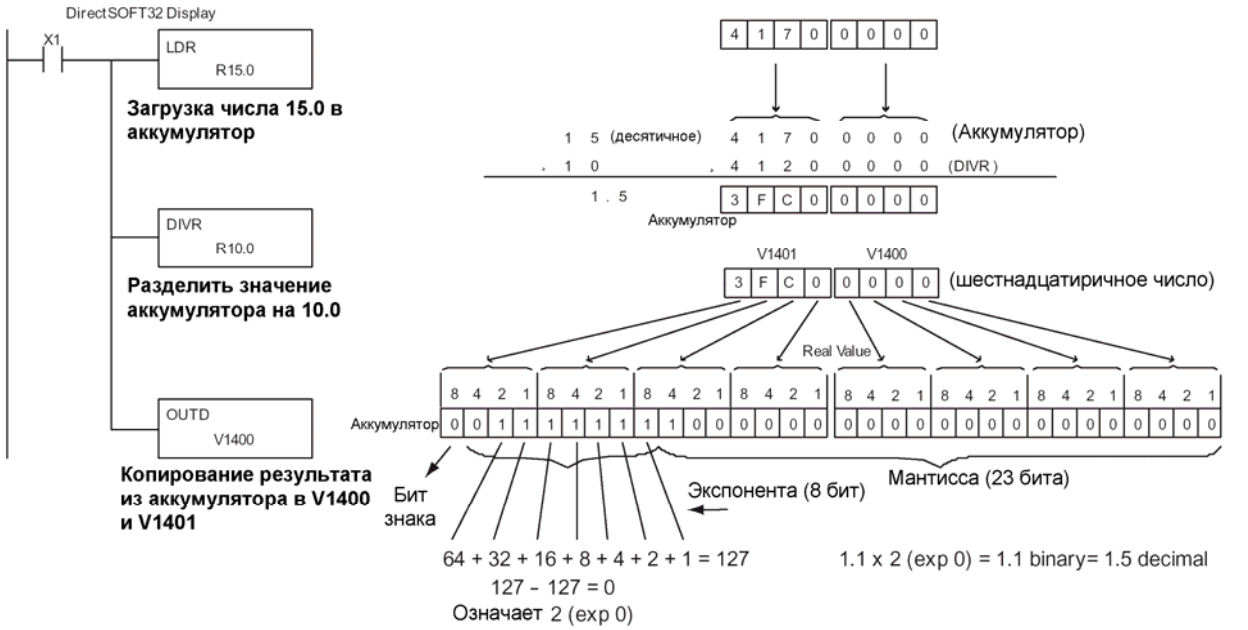
Divide Real (DIVR)

Команда Divide Real делит вещественное число в аккумуляторе на вещественную константу или число, находящееся в двух последовательных ячейках V-памяти. Результат сохраняется в аккумуляторе. Оба числа должны быть представлены в формате IEEE как числа с плавающей запятой.

Тип данных операнда		Диапазон
	A	aaa
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	R	-3.402823E+038 -- +3.402823E+038

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP71	«1» в любое время, когда определено, что указатель ссылается на недопустимую ячейку V-памяти
SP72	«1» в любое время, когда значение в аккумуляторе - недопустимое число с плавающей запятой
SP73	«1», когда сложение или вычитание со знаком приводит к некорректному биту знака.
SP74	«1» в любое время, когда математическая операция с плавающей запятой приводит к ошибке переполнения.
SP75	«1», когда ожидается BCD число, а приходит число в другом формате.

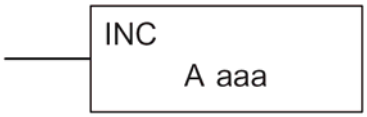
ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.



ПРИМЕЧАНИЕ. Текущая версия ручного программатора не поддерживает ввод вещественных чисел с автоматическим преобразованием к 32-разрядному формату IEEE. Вы должны использовать DirectSOFT32 для этой возможности..

Increment (INC)

Команда Increment увеличивает BCD значение в определенной ячейке V-памяти на «1» каждый раз при выполнении команды.



Decrement (DEC)

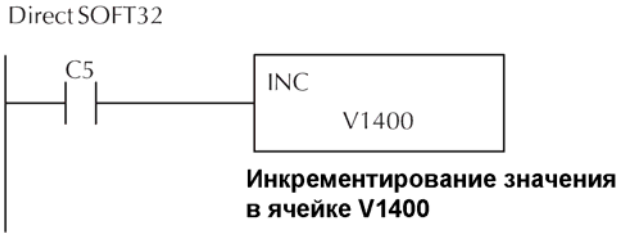
Команда Decrement уменьшает BCD значение в определенной ячейке V-памяти на «1» каждый раз при выполнении команды.

Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти

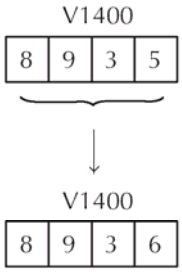
Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP75	«1», когда выполняется BCD команда с не-BCD числом

ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

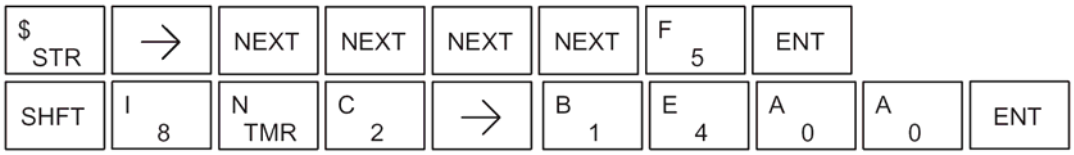
В следующем примере команды Increment, когда C5 включен, значение в V1400 увеличивается на единицу



Инкрементирование значения в ячейке V1400

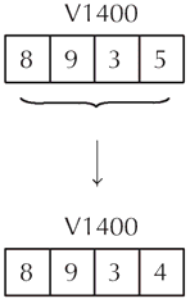


Набор на ручном программаторе

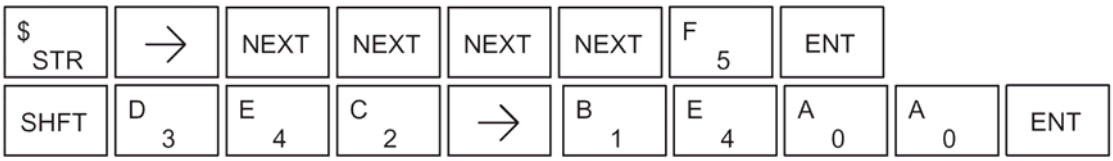


В следующем примере команды Decrement, когда C5 включен, значение в V1400 уменьшается на единицу.

Уменьшение значения ячейки V1400 на единицу



Набор на ручном программаторе



Add Binary (ADDB)

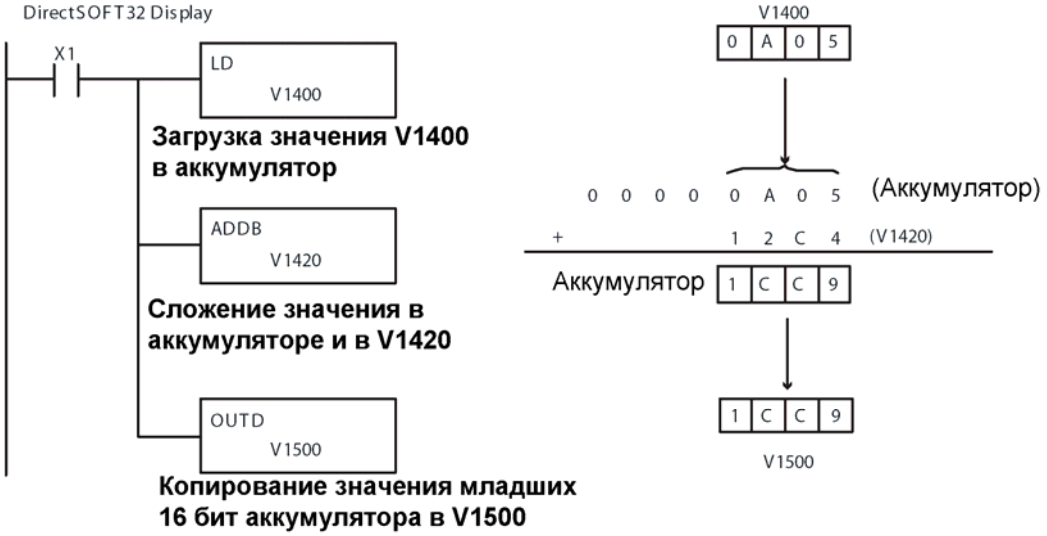
Add Binary — 16-битная команда, которая складывает двоичное значение в дополнительном коде без знака в младших 16 битах аккумулятора с двоичным значением в дополнительном коде без знака (Aaaa), которое является или ячейкой V-памяти или 16-битной константой. Результат может быть до 32 бит (без знака) и находится в аккумуляторе.

Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-FFFF

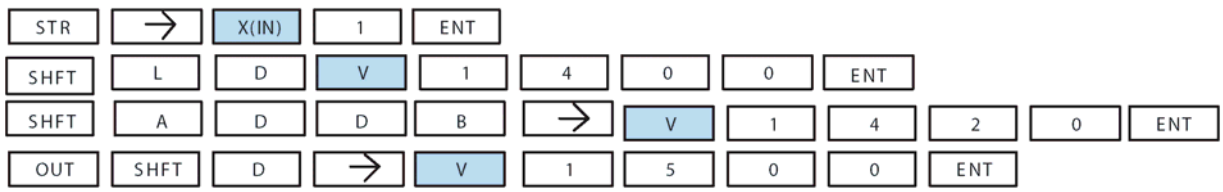
Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP66	«1», когда 16-битовая команда сложения приводит к переносу
SP67	«1», когда 32-битовая команда сложения приводит к переносу
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP73	«1», 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. Значение в аккумуляторе копируется в V1500 и V1501 командой Out.

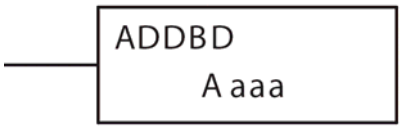


Набор на ручном программаторе



Add Binary Double (ADDBD)

Add Binary Double — 32-битная команда, которая складывает двоичное значение в дополнительном коде без знака в аккумуляторе с двоичным значением в дополнительном коде без знака (Aaaa), которое является или ячейкой V-памяти или 32-битной константой в дополнительном коде. Результат находится в аккумуляторе.

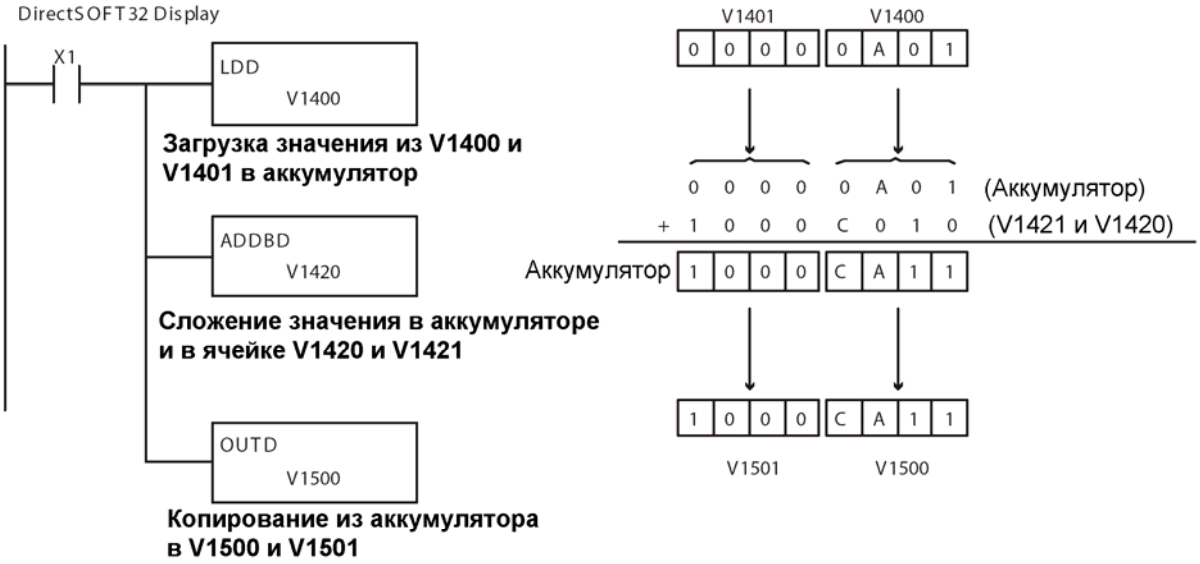


Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-FFFF

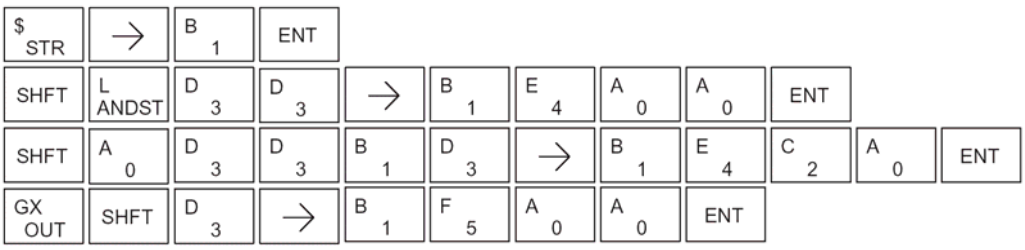
Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP66	«1», когда 16-битовая команда сложения приводит к переносу
SP67	«1», когда 32-битовая команда сложения приводит к переносу
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP73	«1», 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 из 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.

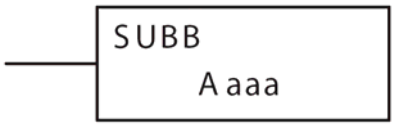


Набор на ручном программаторе



Subtract Binary (SUBB)

Subtract Binary — 16-битная команда, которая вычитает двоичное значение в дополнительном коде без знака (Aaaa), которое является или ячейкой V-памяти или 16-битным двухкомпонентным двоичным значением, из двоичного значения в дополнительном коде в аккумуляторе. Результат находится в аккумуляторе.

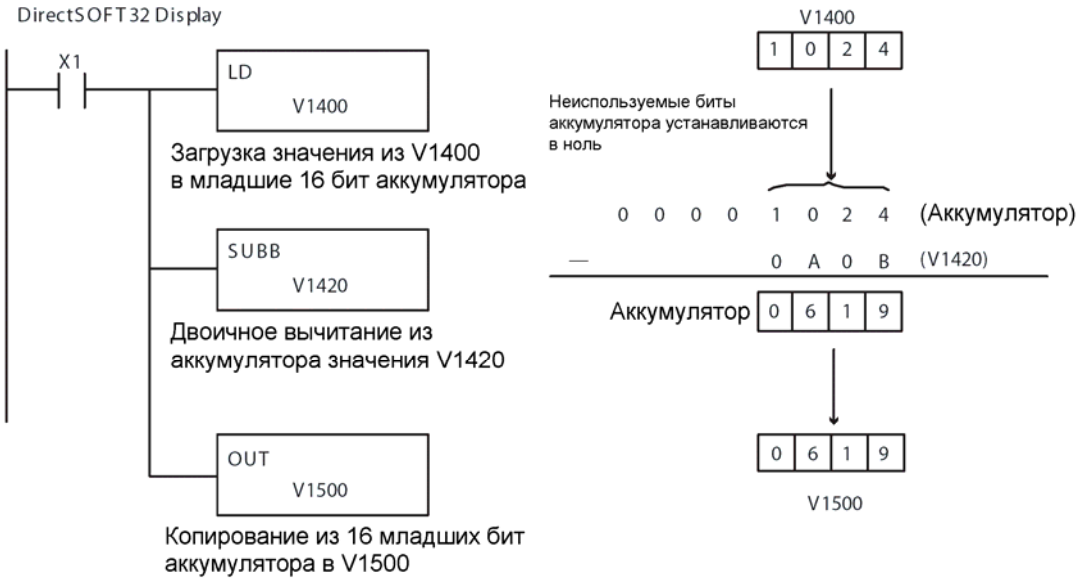


Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-FFFF

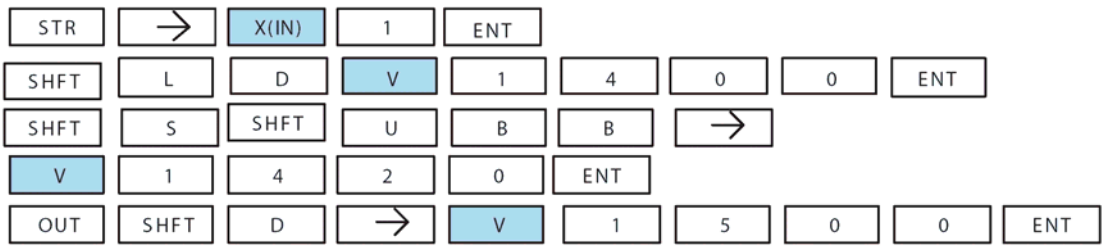
Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP64	«1», когда 16-битовая команда вычитания приводит к заимствованию
SP65	«1», когда 32-битовая команда вычитания приводит к заимствованию
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное

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.

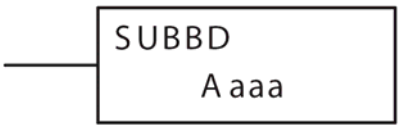


Набор на ручном программаторе



Subtract Binary Double (SUBBD)

Subtract Binary Double — 16-битная команда, которая вычитает двоичное значение в дополнительном коде без знака (Aaaa), которое является или ячейкой V-памяти или 32-битным двоичным значением, из двоичного значения в дополнительном коде в аккумуляторе. Результат находится в аккумуляторе.

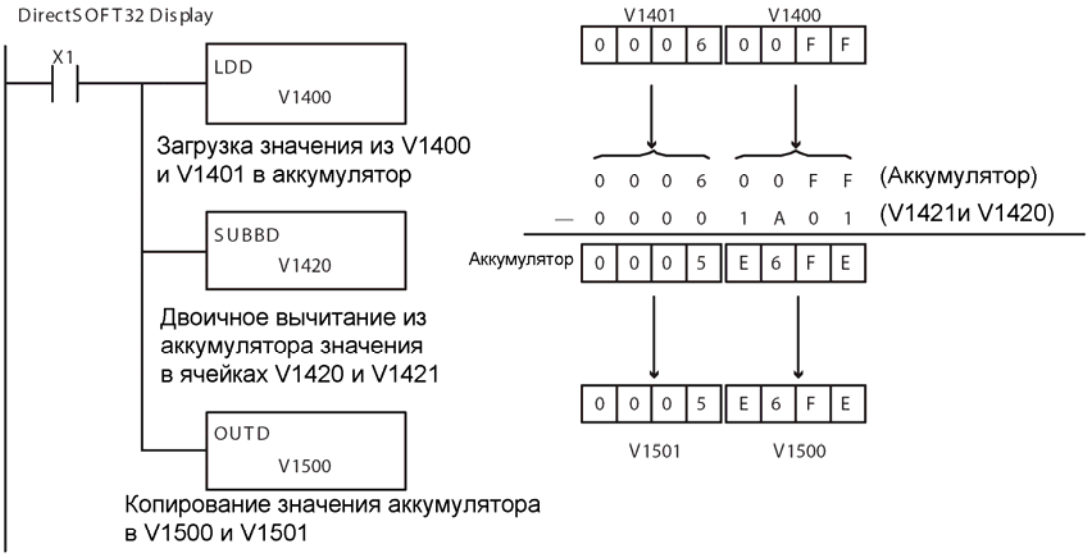


Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0 - FFFF FFFF

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP64	«1», когда 16-битовая команда вычитания приводит к заимствованию
SP65	«1», когда 32-битовая команда вычитания приводит к заимствованию
SP70	«1» 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 из V1400 и V1401 будет загружено в аккумулятор командой Load Double. Двоичное значение в V1420 и V1421 вычитается из двоичного значения в аккумуляторе командой Subtract Binary Double. Значение в аккумуляторе копируется в V1500 и V1501 командой Out Double.

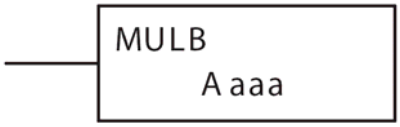


Набор на ручном программаторе

$ STR	→	B 1	ENT									
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT			
SHFT	S RST	SHFT	U ISG	B 1	B 1	D 3	→	B 1	E 4	C 2	A 0	ENT
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT				

Multiply Binary (MULB)

Multiply Binary — 16-битная команда умножения двоичного значения в дополнительном коде без знака (Aaaa), которое является или ячейкой V-памяти или 16-битной двоичной константой в дополнительном коде без знака, на 16-битное двоичное значение в аккумуляторе. Результат может быть до 32 бит и находится в аккумуляторе.

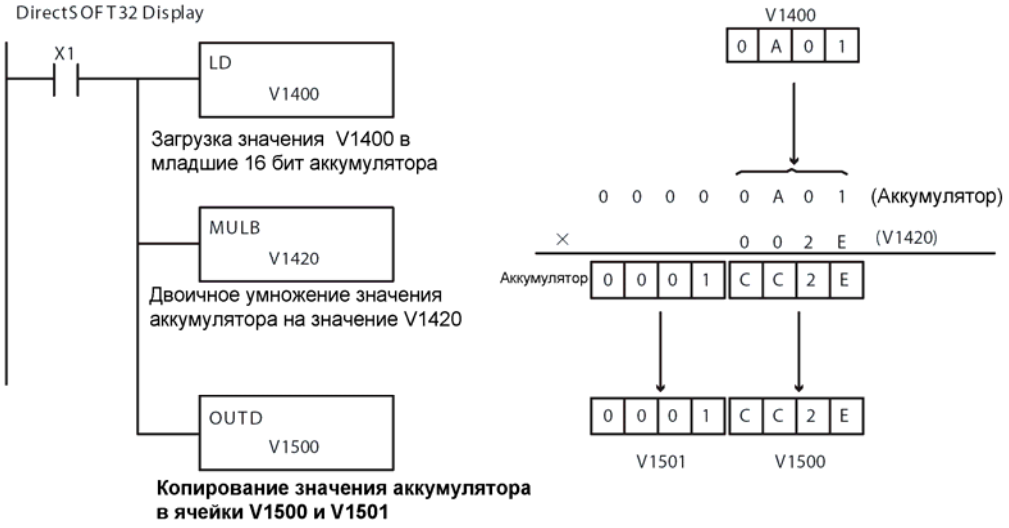


Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-FFFF

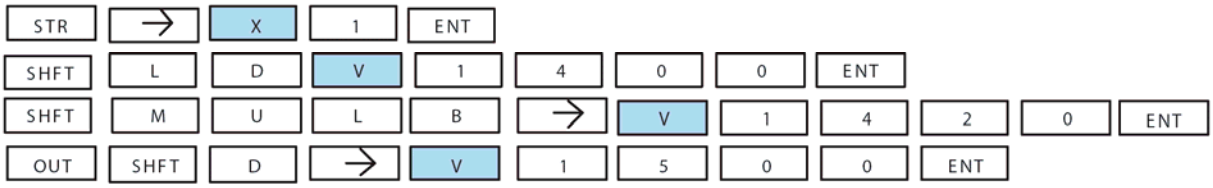
Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное

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 командой Out.

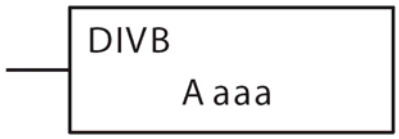


Набор на ручном программаторе



Divide Binary (DIVB)

Divide Binary — 16-битная команда, которая делит двоичное значение в дополнительном коде без знака в аккумуляторе на двоичное значение (Aaaa), которое является или ячейкой V-памяти или 16-битной двоичной константой в дополнительном коде без знака. Целая часть результата находится в аккумуляторе, а остаток от деления — в первом уровне стека.

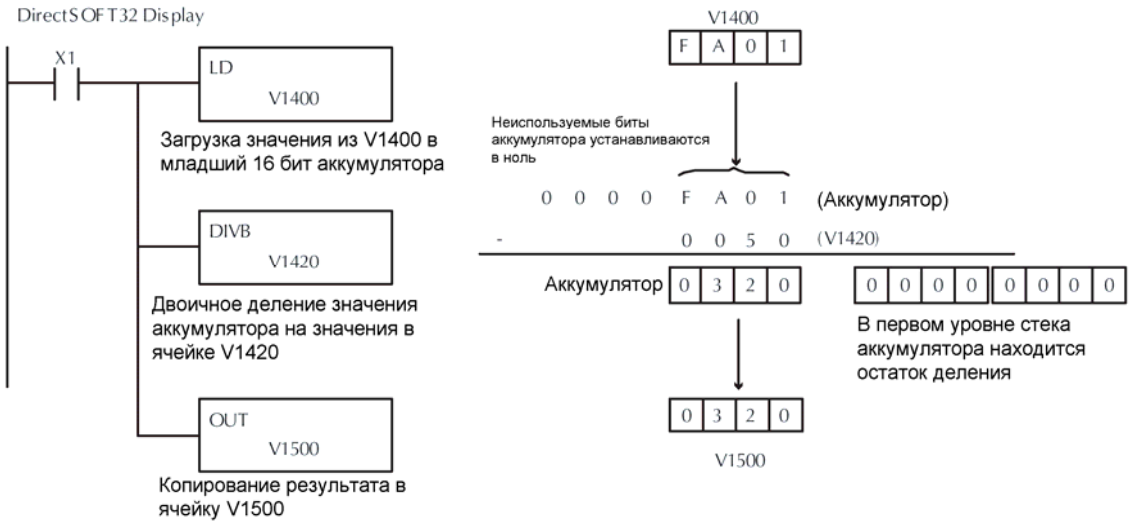


Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-FFFF

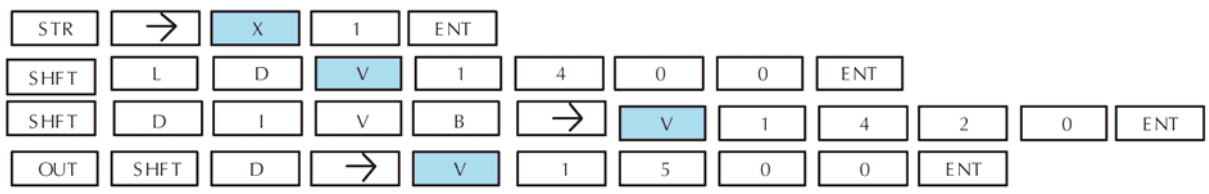
Флаги	Описание
SP53	«1», когда значение операнда больше, чем значение, с которым может работать аккумулятор
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное

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.

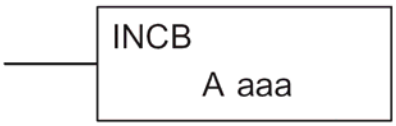


Набор на ручном программаторе



Increment Binary (INCB)

Команда Increment Binary увеличивает двоичное значение в определенной ячейке V-памяти на «1» каждый раз при выполнении команды.

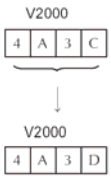


Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти

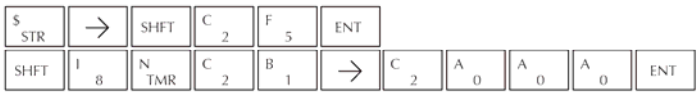
Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем

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 V2000 увеличивается на 1.



Набор на ручном программаторе



Decrement Binary (DECB)

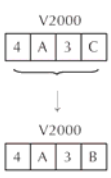
Команда Decrement Binary уменьшает двоичное значение в определенной ячейке V-памяти на «1» каждый раз при выполнении команды.

Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти

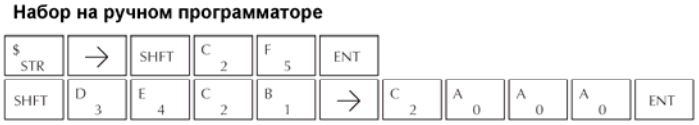
Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем

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 V2000 уменьшается на 1.



Набор на ручном программаторе



Add Formatted (ADDF)

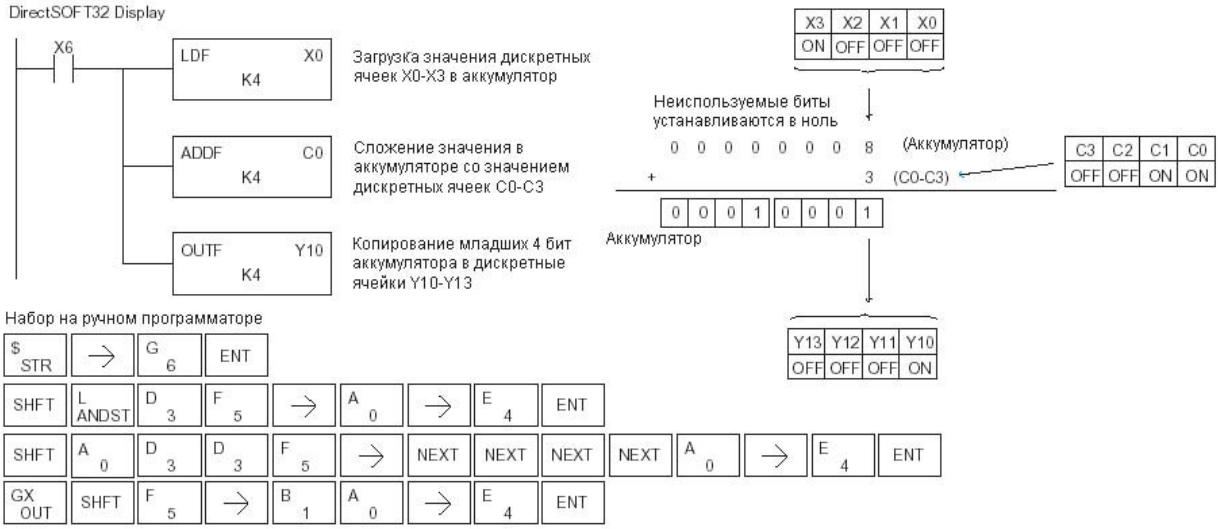
Add Formatted — 32-битная команда, которая складывает BCD значение в аккумуляторе с BCD значением (Aaaa), которое является диапазоном бит. Указанный диапазон (Kbbb) может быть от 1 до 32 последовательных бит. Результат находится в аккумуляторе.

Тип данных операнда		Диапазон DL06	
	A/B	aaa	bbb
Входы	X	0-777	-
Выходы	Y	0-777	-
Управляющие реле	C	0-1777	-
Биты стадий	S	0-1777	-
Биты таймера	T	0-377	-
Биты счетчика	CT	0-177	-
Специальные реле	SP	0-137, 320-717	-
Удаленный ввод/вывод	GX	0-3777	
Константа	K	-	1-32

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP66	«1», когда 16-битовая команда сложения приводит к переносу
SP67	«1», когда 32-битовая команда сложения приводит к переносу
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD 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, значение дискретных ячеек X0-X3 будет загружено в аккумулятор командой Load Formatted. Двоичное значение в аккумуляторе будет сложено с двоичным значением в ячейках C0–C3 командой Add Formatted. Значение в аккумуляторе выводится на выходы Y10–Y13 командой Out Formatted.



Subtract Formatted (SUBF)

Add Formatted — 32-битная команда, которая вычитает из BCD значения аккумулятора BCD значение (Aaaa), которое является диапазоном бит. Указанный диапазон (Kbbb) может быть от 1 до 32 последовательных бит. Результат находится в аккумуляторе.

Тип данных операнда		Диапазон DL06	
	A/B	aaa	bbb
Входы	X	0-777	-
Выходы	Y	0-777	-
Управляющие реле	C	0-1777	-
Биты стадий	S	0-1777	-
Биты таймера	T	0-377	-
Биты счетчика	CT	0-177	-
Специальные реле	SP	0-137, 320-717	-
Глобальный ввод/вывод	GX	0-3777	
Константа	K	-	1-32

Флаги	**Описание**
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP64	«1», когда 16-битовая команда вычитания приводит к заимствованию
SP65	«1», когда 32-битовая команда вычитания приводит к заимствованию
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD 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, значение дискретных ячеек X0-X3 будет загружено в аккумулятор командой Load Formatted. Из двоичного значения в аккумуляторе будет вычтено двоичное значение в ячейках C0–C3 командой Subtract Formatted. Значение в аккумуляторе выводится на выходы Y10–Y13 командой Out Formatted.

Multiply Formatted (MULF)

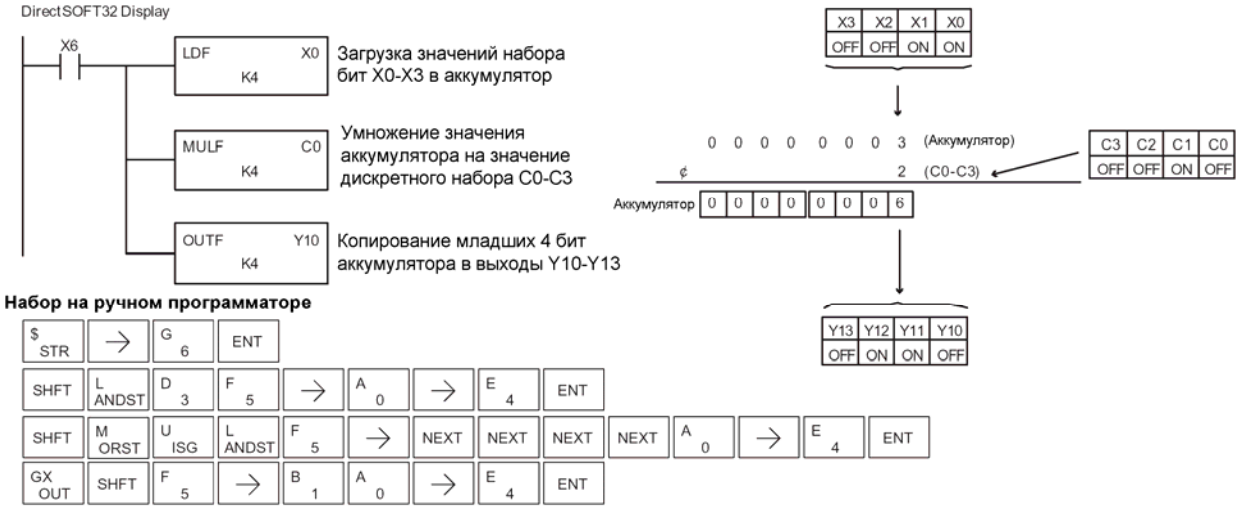
Multiply Formatted — 16-битная команда, которая умножает BCD значение аккумулятора на BCD значение (Aaaa), которое является диапазоном бит. Указанный диапазон (Kbbb) может быть от 1 до 16 последовательных бит. Результат находится в аккумуляторе.

Тип данных операнда		Диапазон DL06	
	A	aaa	bbb
Входы	X	0-777	-
Выходы	Y	0-777	-
Управляющие реле	C	0-1777	-
Биты стадий	S	0-1777	-
Биты таймера	T	0-377	-
Биты счетчика	CT	0-177	-
Специальные реле	SP	0-137, 320-717	-
Глобальный ввод/вывод	GX	0-3777	
Константа	K	-	1-16

Флаги	**Описание**
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD 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, значение дискретных ячеек X0-X3 будет загружено в аккумулятор командой Load Formatted. Двоичное значение в аккумуляторе будет умножено на двоичное значение в ячейках C0–C3 командой Multiply Formatted. Значение в аккумуляторе выводится на выходы Y10–Y13 командой Out Formatted.



Divide Formatted (DIVF)

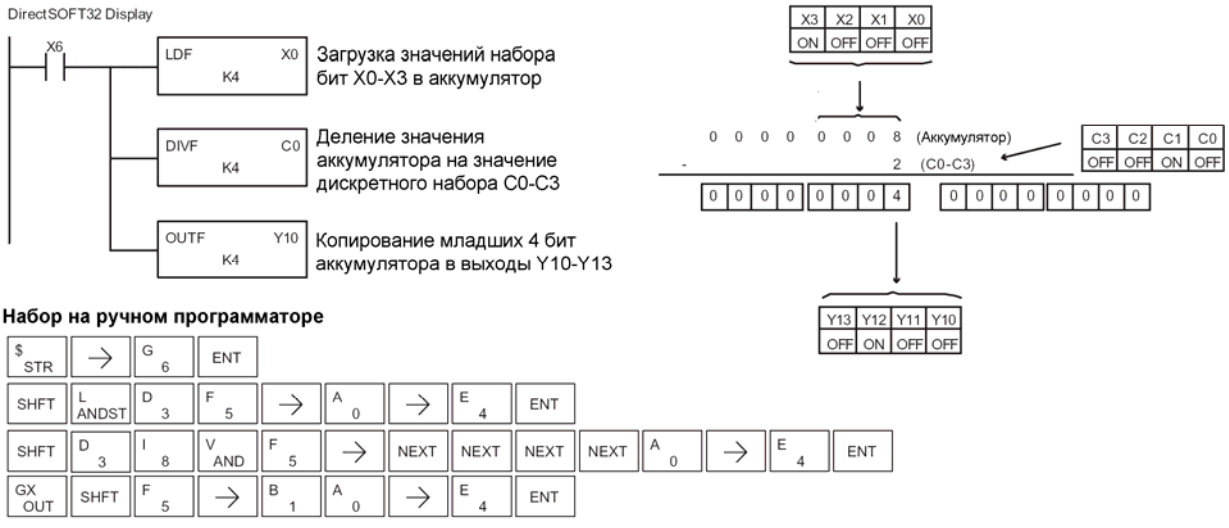
Divide Formatted — 16-битная команда, которая делит BCD значение аккумулятора на BCD значение (Aaaa), которое является диапазоном бит. Указанный диапазон (Kbbb) может быть от 1 до 16 последовательных бит. Результат находится в аккумуляторе.

Тип данных операнда		Диапазон DL06	
	A	aaa	bbb
Входы	X	0-777	-
Выходы	Y	0-777	-
Управляющие реле	C	0-1777	-
Биты стадий	S	0-1777	-
Биты таймера	T	0-377	-
Биты счетчика	CT	0-177	-
Специальные реле	SP	0-137, 320-717	-
Глобальный ввод/вывод	GX	0-3777	
Константа	K	-	1-16

Флаги	**Описание**
SP53	«1», когда значение операнда больше, чем значение, с которым может работать аккумулятор
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD числом

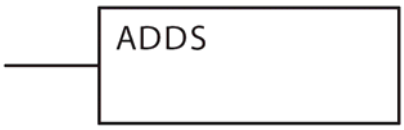
ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

В следующем примере, когда X6 включен, значение дискретных ячеек X0-X3 будет загружено в аккумулятор командой Load Formatted. Двоичное значение в аккумуляторе будет разделено на двоичное значение в ячейках C0–C3 командой Divide Formatted. Значение в аккумуляторе выводится на выходы Y10–Y13 командой Out Formatted.



Add Top of Stack (ADDS)

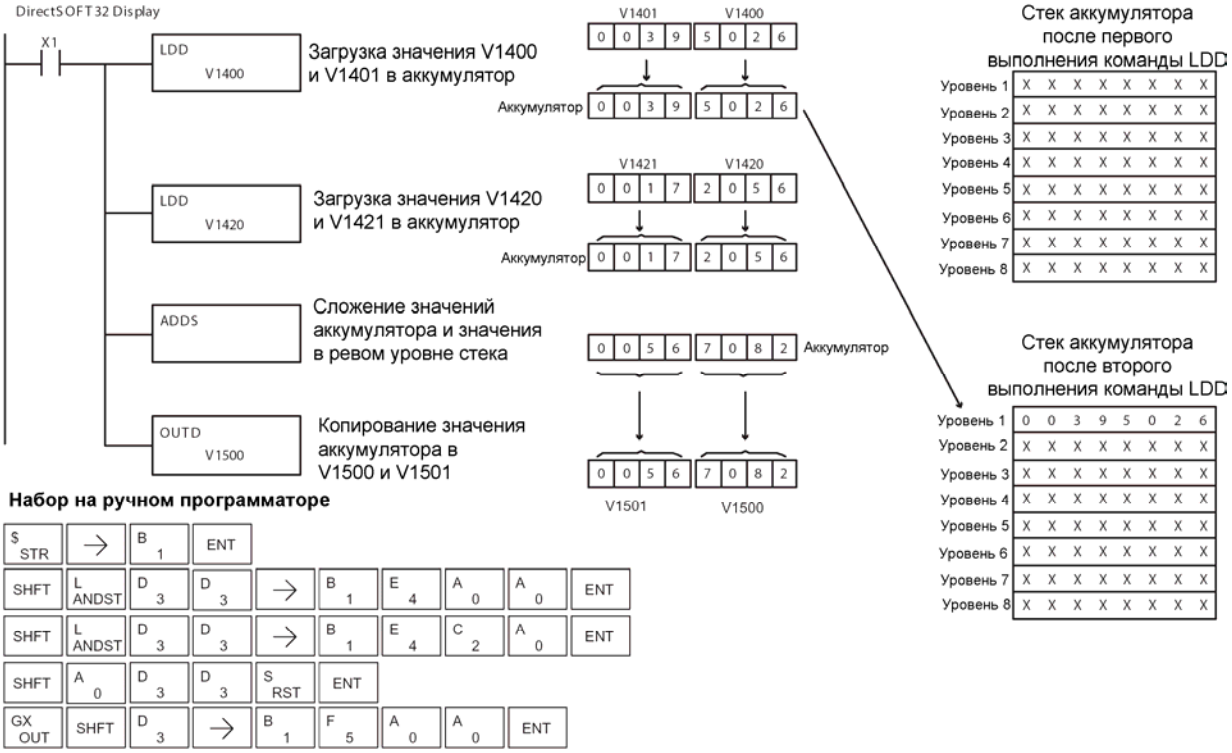
Add Top of Stack — 32-битная команда, которая выполняет сложение BCD значение аккумулятора с BCD значением в первом уровне стека аккумулятора. Результат находится в аккумуляторе. Значение в первом уровне стека удаляется и все значения в стеке поднимаются вверх на один уровень.



Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP66	«1», когда 16-битовая команда сложения приводит к переносу
SP67	«1», когда 32-битовая команда сложения приводит к переносу
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD 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 V1400 и V1401 будет загружено в аккумулятор командой Load Double. Затем загружается значение V1420 и V1421 командой Load Double в аккумулятор, а предыдущее значение аккумулятора перемещается на первый уровень стека. Значение с первого уровня стека и аккумулятора складываются командой Add Stack. Значение в аккумуляторе выводится в V1500 и V1501 командой Out Double.



Набор на ручном программаторе

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	C 2	A 0	ENT
SHFT	A 0	D 3	D 3	S RST	ENT			
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT

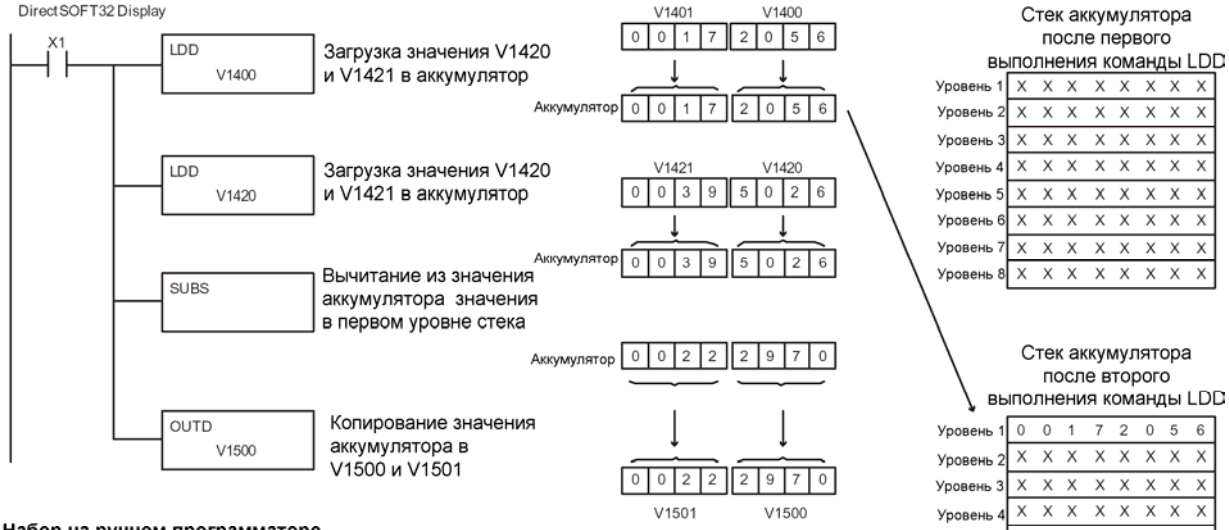
Subtract Top of Stack (SUBS)

Subtract Top of Stack — 32-битная команда, которая выполняет вычитание из BCD значения аккумулятора BCD значения в первом уровне стека аккумулятора. Результат находится в аккумуляторе. Значение в первом уровне стека удаляется и все значения в стеке поднимаются вверх на один уровень.

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP66	«1», когда 16-битовая команда сложения приводит к переносу
SP67	«1», когда 32-битовая команда сложения приводит к переносу
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD 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 V1400 и V1401 будет загружено в аккумулятор командой Load Double. Затем загружается значение V1420 и V1421 командой Load Double в аккумулятор, а предыдущее значение аккумулятора перемещается на первый уровень стека. Из значения аккумулятора вычитается значение первого уровня стека командой Subtract Top of Stack Значение в аккумуляторе выводится в V1500 и V1501 командой Out Double.

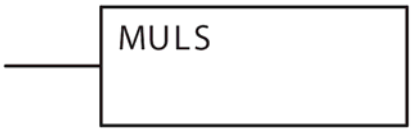


Набор на ручном программаторе

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	C 2	A 0	ENT
SHFT	S RST	SHFT	U ISG	B 1	S RST	ENT			
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT	

Multiply Top of Stack (MULS)

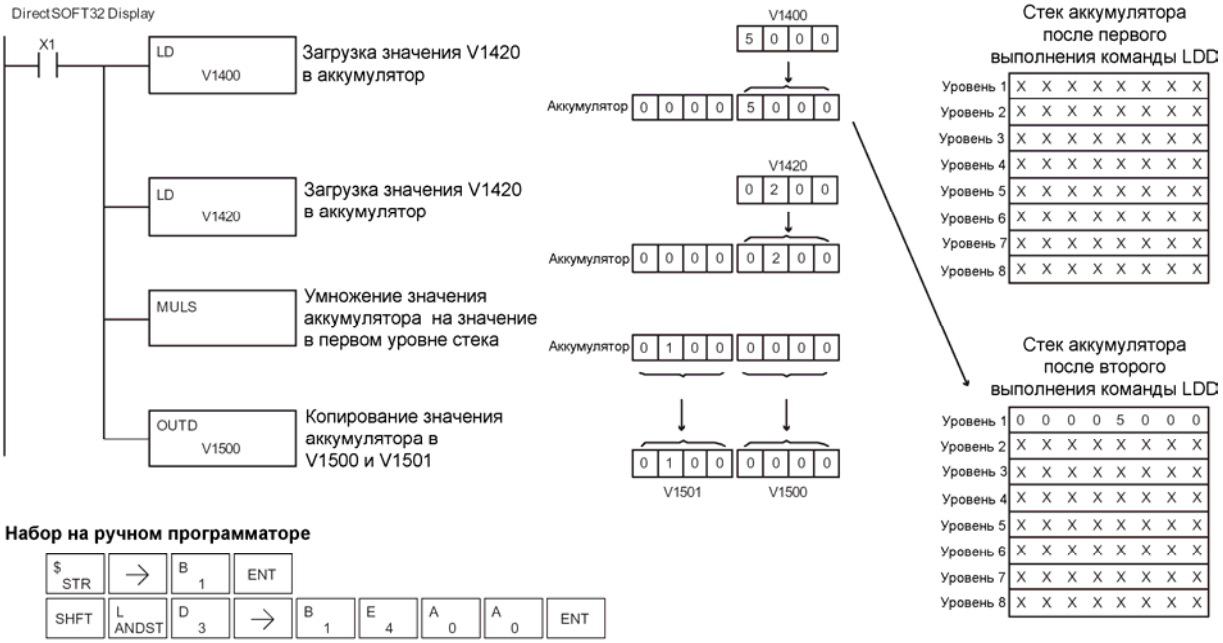
Multiply Top of Stack — 16-битная команда, которая выполняет умножение 4-х разрядного BCD значения аккумулятора и 4-х разрядного BCD значения в первом уровне стека аккумулятора. Результат находится в аккумуляторе. Значение в первом уровне стека удаляется и все значения в стеке поднимаются вверх на один уровень.



Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD 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 V1400 будет загружено в аккумулятор командой Load. Затем загружается значение V1420 командой Load в аккумулятор, а предыдущее значение аккумулятора перемещается на первый уровень стека. Значение аккумулятора умножается на значение первого уровня стека командой Multiply Top of Stack. Значение в аккумуляторе выводится в V1500 и V1501 командой Out Double.



Набор на ручном программаторе

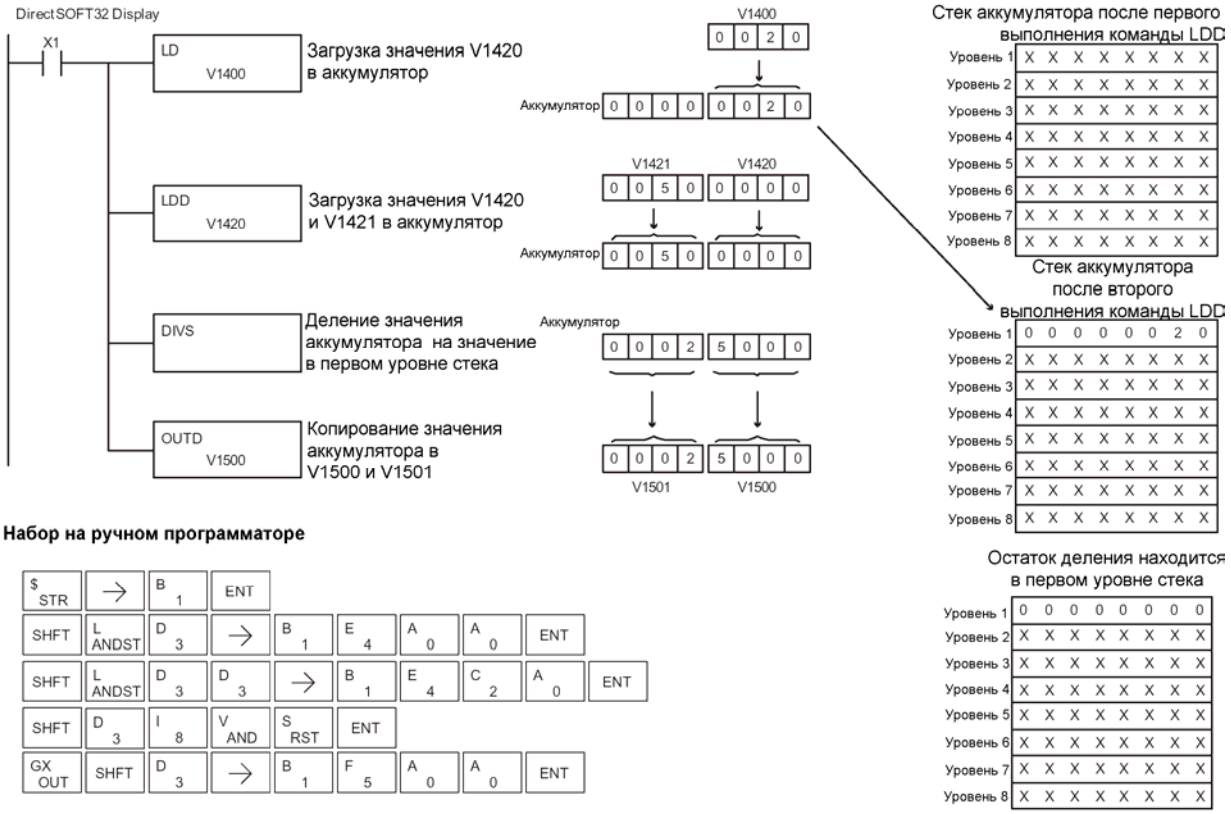
Divide by Top of Stack (DIVS)

Divide Top of Stack — 32-битная команда, которая выполняет деление 8-разрядного BCD значения в аккумуляторе на 4-разрядное BCD значение в первом уровне стека аккумулятора. Результат находится в аккумуляторе, а остаток в первом уровне стека.

Флаги	Описание
SP53	«1», когда значение операнда больше, чем значение, с которым может работать аккумулятор
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP66	«1», когда 16-битовая команда сложения приводит к переносу
SP67	«1», когда 32-битовая команда сложения приводит к переносу
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD числом

ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

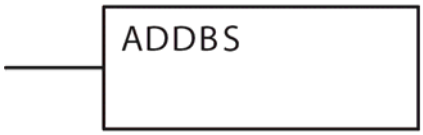
В следующем примере, когда X1 включен, значение V1400 будет загружено в аккумулятор командой Load. Затем загружается значение V1420 командой Load Double в аккумулятор, а предыдущее значение аккумулятора перемещается на первый уровень стека. Значение аккумулятора делится на значение первого уровня стека командой Divide Stack. Значение в аккумуляторе выводится в V1500 и V1501 командой Out Double.



Набор на ручном программаторе

Add Binary Top of Stack (ADDBS)

Add Binary Top of Stack — 32-битная команда, которая выполняет сложение двоичного значения аккумулятора с двоичным значением в первом уровне стека аккумулятора. Результат находится в аккумуляторе. Значение в первом уровне стека удаляется и все значения в стеке поднимаются вверх на один уровень.



Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP66	«1», когда 16-битовая команда сложения приводит к переносу
SP67	«1», когда 32-битовая команда сложения приводит к переносу
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP73	«1», когда сложение или вычитание со знаком приводит к некорректному биту знака.

ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

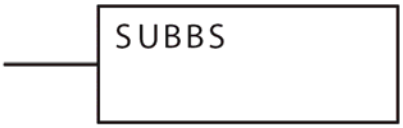
В следующем примере, когда X1 включен, значение V1400 и V1401 будет загружено в аккумулятор командой Load Double. Затем загружается значение V1420 и V1421 командой Load Double в аккумулятор, а предыдущее значение аккумулятора перемещается на первый уровень стека. Значение с первого уровня стека и аккумулятора складываются командой Add Stack. Значение в аккумуляторе выводится в V1500 и V1501 командой Out Double.

Набор на ручном программаторе

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	C 2	A 0	ENT
SHFT	A 0	D 3	D 3	B 1	S RST	ENT			
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT	

Subtract Binary Top of Stack (SUBBS)

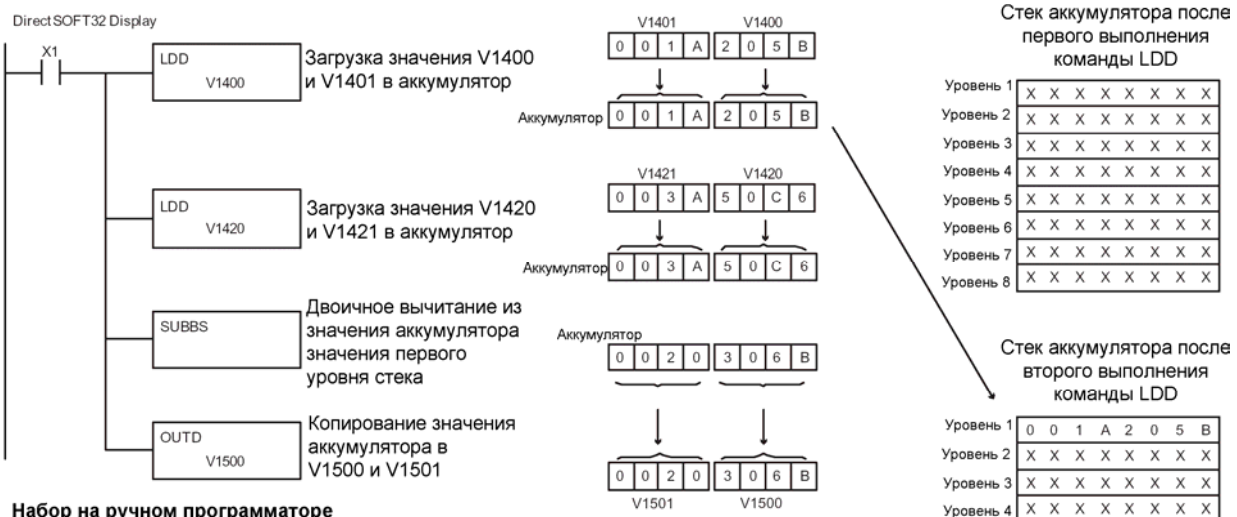
Subtract Binary Top of Stack — 32-битная команда, которая выполняет вычитание из двоичного значения аккумулятора двоичное значение в первом уровне стека аккумулятора. Результат находится в аккумуляторе. Значение в первом уровне стека удаляется и все значения в стеке поднимаются вверх на один уровень.



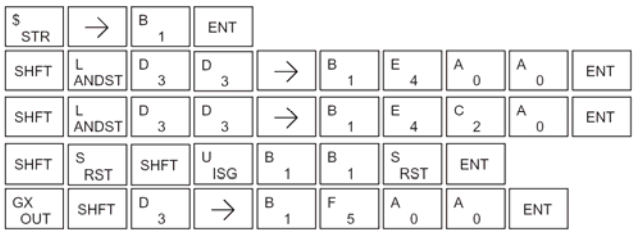
Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP64	«1», когда 16-битовая команда вычитания приводит к заимствованию
SP65	«1», когда 32-битовая команда вычитания приводит к заимствованию
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное

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 V1400 и V1401 будет загружено в аккумулятор командой Load Double. Затем загружается значение V1420 и V1421 командой Load Double в аккумулятор, а предыдущее значение аккумулятора перемещается на первый уровень стека. Из значения аккумулятора вычитается значение первого уровня стека командой Subtract Stack Значение в аккумуляторе выводится в V1500 и V1501 командой Out Double.

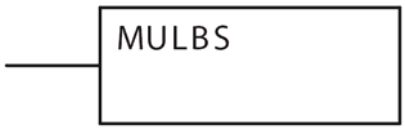


Набор на ручном программаторе



Multiply Binary Top of Stack (MULBS)

Multiply Binary Top of Stack — 16-битная команда, которая выполняет умножение 16-и битного двоичного значения аккумулятора и 16-и битного двоичного значения в первом уровне стека аккумулятора. Результат находится в аккумуляторе и может быть размером до 32 бит(8-мь цифр максимум). Значение в первом уровне стека удаляется и все значения в стеке поднимаются вверх на один уровень.



Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» 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 V1400 будет загружено в аккумулятор командой Load. Затем загружается значение V1420 командой Load в аккумулятор, а предыдущее значение аккумулятора перемещается на первый уровень стека. Значение аккумулятора умножается на значение первого уровня стека командой Multiply Binary Stack. Значение в аккумуляторе выводится в V1500 и V1501 командой Out Double.

Набор на ручном программаторе

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→	B 1	E 4	A 0	A 0	ENT
SHFT	L ANDST	D 3	→	B 1	E 4	C 2	A 0	ENT
SHFT	M ORST	U ISG	L ANDST	B 1	S RST	ENT		
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT

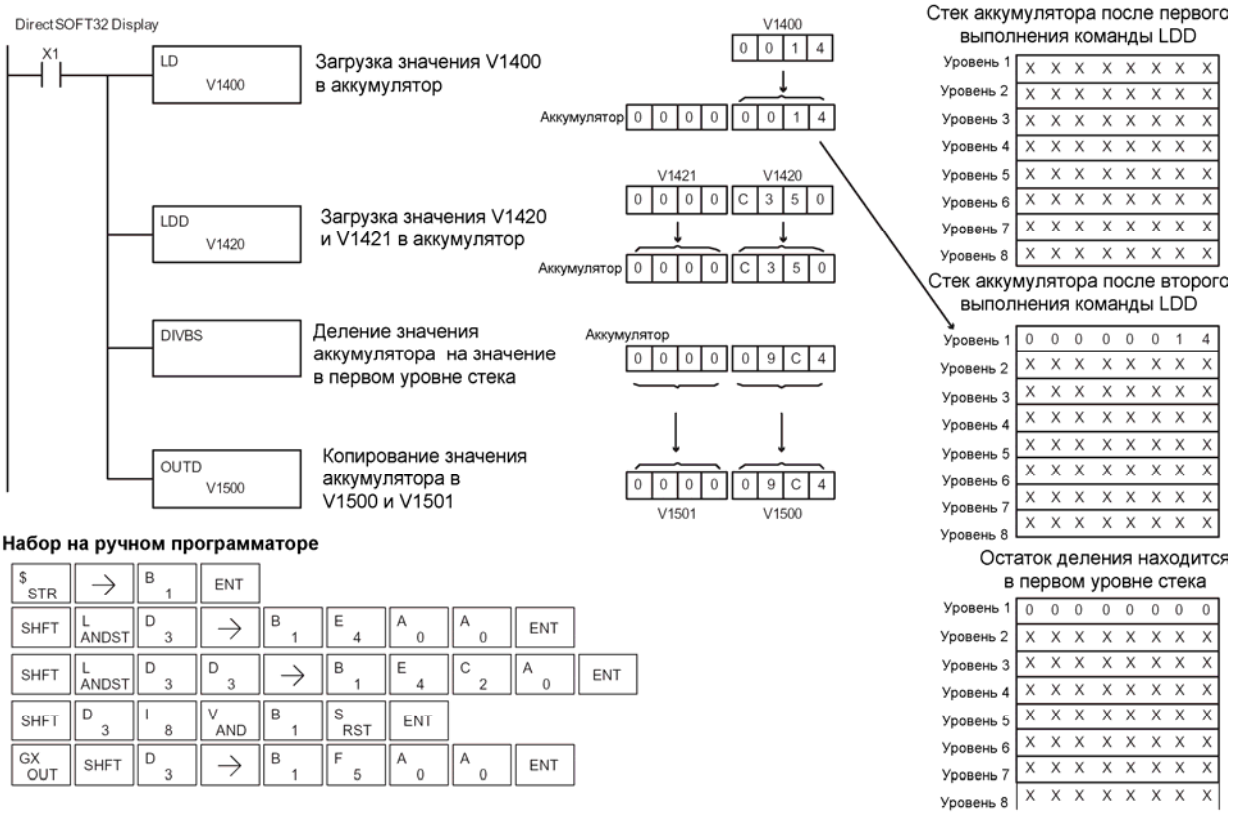
Divide Binary by Top OF Stack (DIVBS)

Divide Binary Top of Stack — 32-битная команда, которая выполняет деление 32-х битного двоичного значения в аккумуляторе на 16-ти битное двоичное значение в первом уровне стека аккумулятора. Результат находится в аккумуляторе, а остаток в первом уровне стека.

Флаги	Описание
SP53	«1», когда значение операнда больше, чем значение, с которым может работать аккумулятор
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» 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 V1400 будет загружено в аккумулятор командой Load. Затем загружается значение V1420 и V1421 командой Load Double в аккумулятор, а предыдущее значение аккумулятора перемещается на первый уровень стека. Значение аккумулятора делится на значение первого уровня стека командой Divide Binary Top of Stack. Значение в аккумуляторе выводится в V1500 и V1501 командой Out Double.



Набор на ручном программаторе

Трансцендентные функции

Процессор DL06 имеет специальные функции для работы с вещественными числами. Трансцендентные функции включают в себя тригонометрический синус, косинус и тангенс, а также их обратные функции (арксинус, арккосинус и арктангенс). Функция квадратного корня также сгруппирована с этими функциями.

Трансцендентальные математические команды оперируют вещественными числами в аккумуляторе (использовать BCD или двоичное число эти функции на могут). Результат, являющийся вещественным числом, сохраняется в аккумуляторе. Функция квадратного корня работает на полном диапазоне положительных вещественных чисел. Функции синуса, косинуса и тангенса требуют чисел, выраженных в радианах. Вы можете работать с углами, выраженными в градусах, предварительно преобразовав их к радианам командой Radian (RAD), и затем выполняя тригонометрические функции над ними. Все трансцендентные функции используют следующие биты флага.

Флаги	Описание
SP53	«1», когда значение операнда больше, чем значение, с которым может работать аккумулятор
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP72	«1» в любое время, когда значение в аккумуляторе - недопустимое число с плавающей запятой
SP73	«1», когда сложение или вычитание со знаком приводит к некорректному биту знака.
SP75	«1», когда выполняется вещественная операция над невещественным числом.

Sine Real (SINR)

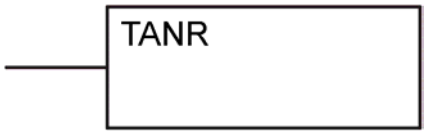
Команда Sine Real вычисляет синус вещественного числа находящегося в аккумуляторе. Результат остается в аккумуляторе. И первоначальное число, и результат имеют 32-битный формат представления вещественного числа IEEE.

Cosine Real (COSR)

Команда Cosine Real вычисляет косинус вещественного числа находящегося в аккумуляторе. Результат остается в аккумуляторе. И первоначальное число, и результат имеют 32-битный формат представления вещественного числа IEEE.

Tangent Real (TANR)

Команда Tangent Real вычисляет тангенс вещественного числа находящегося в аккумуляторе. Результат остается в аккумуляторе. И первоначальное число, и результат имеют 32-битный формат представления вещественного числа IEEE.

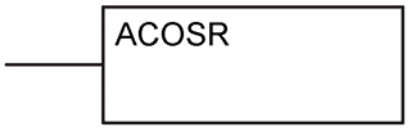


Arc Sine Real (ASINR)

Команда Arc Sine Real вычисляет арксинус вещественного числа находящегося в аккумуляторе. Результат остается в аккумуляторе. И первоначальное число, и результат имеют 32-битный формат представления вещественного числа IEEE.

Arc Cosine Real (ACOSR)

Команда Arc Cosine Real вычисляет арккосинус вещественного числа находящегося в аккумуляторе. Результат остается в аккумуляторе. И первоначальное число, и результат имеют 32-битный формат представления вещественного числа IEEE.



Arc Tangent Real (ATANR)

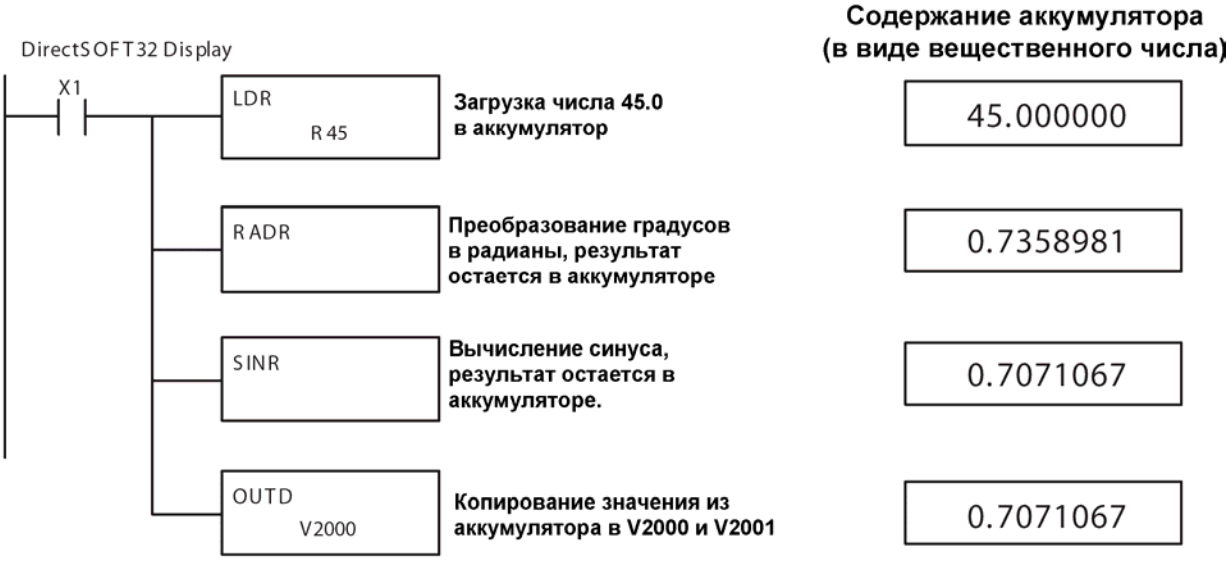
Команда Arc Tangent Real вычисляет арктангенс вещественного числа находящегося в аккумуляторе. Результат остается в аккумуляторе. И первоначальное число, и результат имеют 32-битный формат представления вещественного числа IEEE.

Square Root Real (SQRTR)

Команда Square Root Real вычисляет квадратный корень вещественного числа находящегося в аккумуляторе. Результат остается в аккумуляторе. И первоначальное число, и результат имеют 32-битный формат представления вещественного числа IEEE.

ПРИМЕЧАНИЕ. Функция квадратного корня может быть полезна в отдельных ситуациях. Однако, если Вы применяете функцию извлечения квадратного корня для измерения диафрагменного расходомера как PV(задание) в PID-регуляторе, обратите внимание, что PID-регулятор уже имеет встроенную функцию извлечения квадратного корня.

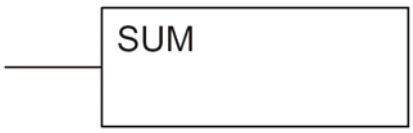
Следующий пример вычисляет синус 45 градусов. Так как эти трансцендентные функции функционируют только с вещественными числами, то мы выполняем команду LDR (загрузка вещественного числа) 45. Тригонометрические функции функционируют только в радианах, так что мы должны преобразовать градусы в радианы, используя команду RADR. После выполнения команды SINR, мы используем команду OUTD, чтобы переместить результат от аккумулятора в V-память. 32-битный результат требует применения команды Out Double.



Команды работы с битами

Sum (SUM)

Команда Sum считает число битов в аккумуляторе, которые установлены в «1». Результат в шестнадцатиричном виде находится в аккумуляторе.



Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем

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	→	B 1	ENT						
SHFT	L ANDST	D 3	F 5	→	B 1	A 0	→	I 8	ENT
SHFT	S RST	SHFT	U ISG	M ORST	→	ENT			
GX OUT	→	PREV	PREV	PREV	B 1	F 5	A 0	A 0	ENT

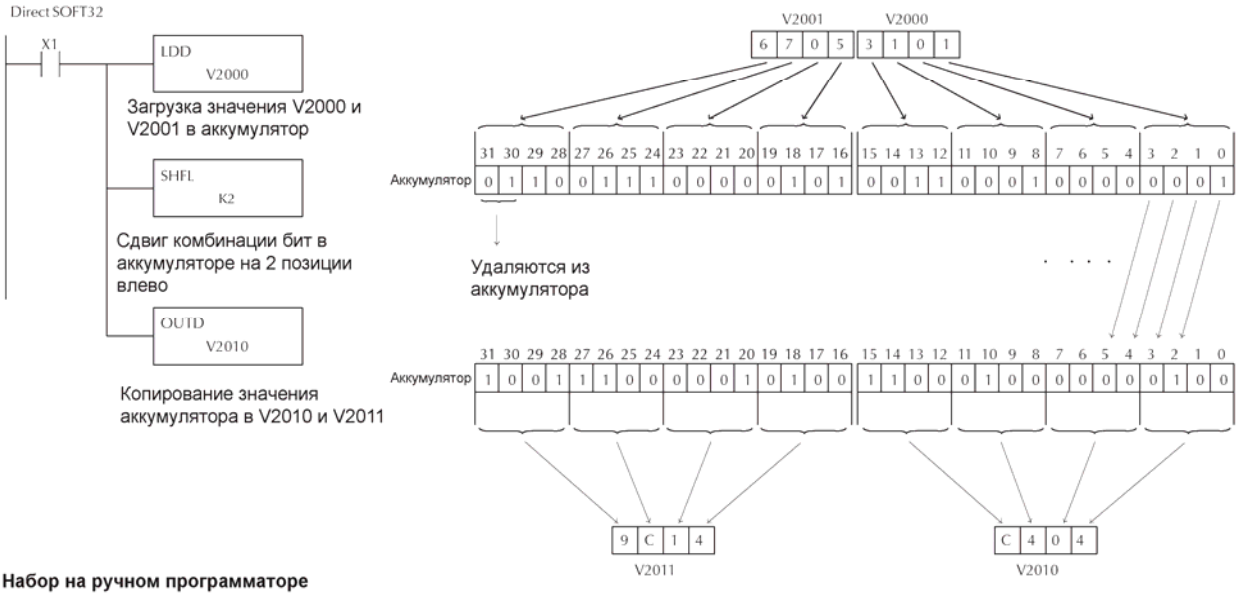
Shift Left (SHFL)

Shift Left — 32-битная команда, которая сдвигает биты в аккумуляторе на указанное число (Aaaa) мест влево. Незанятые позиции заполняются нулями, а биты, сдвинутые из аккумулятора, теряются.

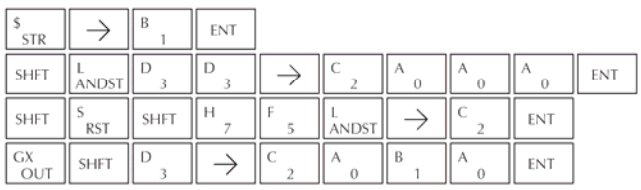
Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти
Константа	K	1-32

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное

В следующем примере, когда X1 включен, значение в V2000 и V2001 будет загружено в аккумулятор командой Load Double. Биты в аккумуляторе сдвигаются на 2 бита влево, командой Shift Left. Значение в аккумуляторе копируется в V2010 и V2011, командой Out Double.

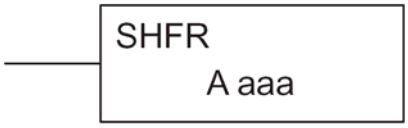


Набор на ручном программаторе



Shift Right (SHFR)

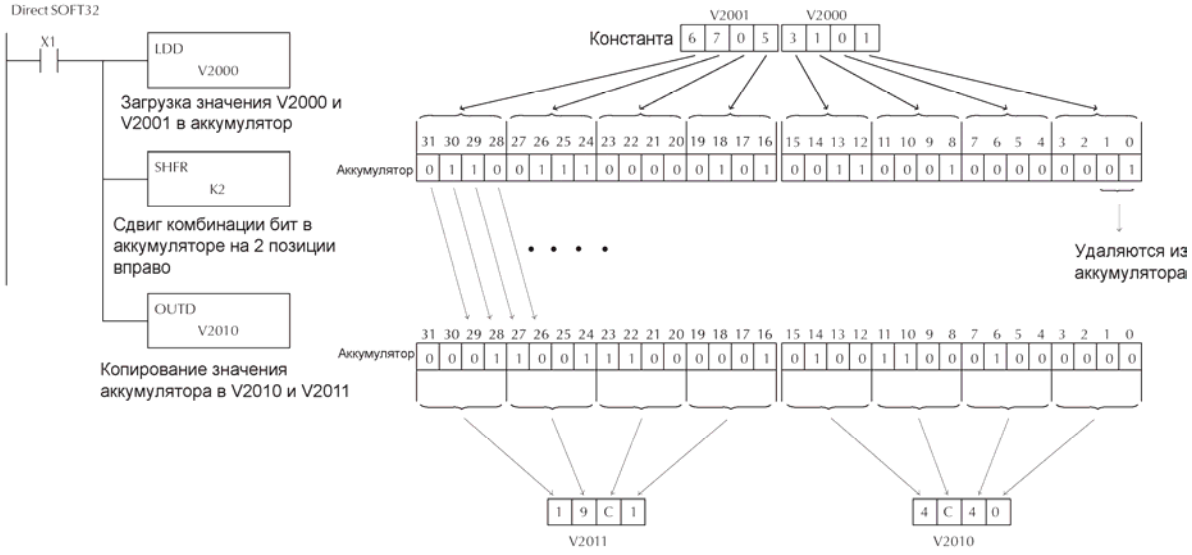
Shift Right — 32-битная команда, которая сдвигает биты в аккумуляторе на указанное число (Аааа) мест вправо. Незанятые позиции заполняются нулями, а биты, сдвинутые из аккумулятора, теряются.



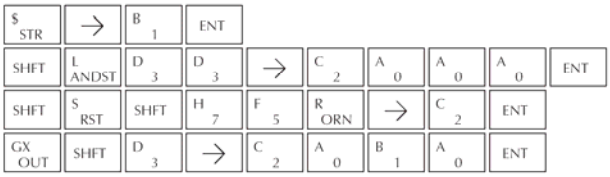
Тип данных операнда		DL06 Диапазон
	A	**ааа**
V-память	V	Смотри карту памяти
Константа	K	1-32

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное

В следующем примере, когда X1 включен, значение в V2000 и V2001 будет загружено в аккумулятор командой Load Double. Биты в аккумуляторе сдвигаются на 2 бита вправо, командой Shift Right. Значение в аккумуляторе копируется в V2010 и V2011, командой Out Double.



Набор на ручном программаторе

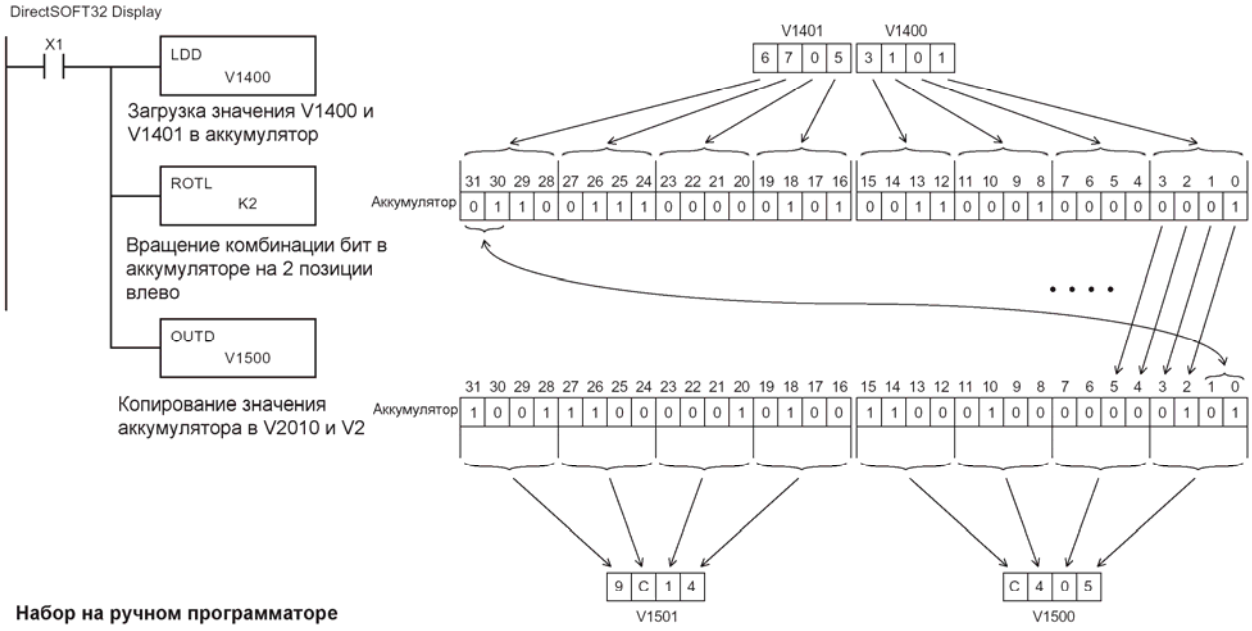


Rotate Left (ROTL)

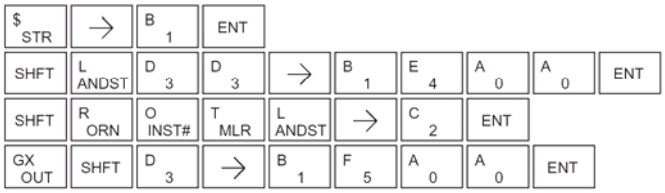
Rotate Left — 32-битная команда, которая вращает биты в аккумуляторе на указанное число (Aaaa) мест влево.

Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти
Константа	K	1-32

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Биты в аккумуляторе вращаются на 2 бита влево, командой Rotate Left. Значение в аккумуляторе копируется в V1500 и V1501, командой Out Double.



Набор на ручном программаторе

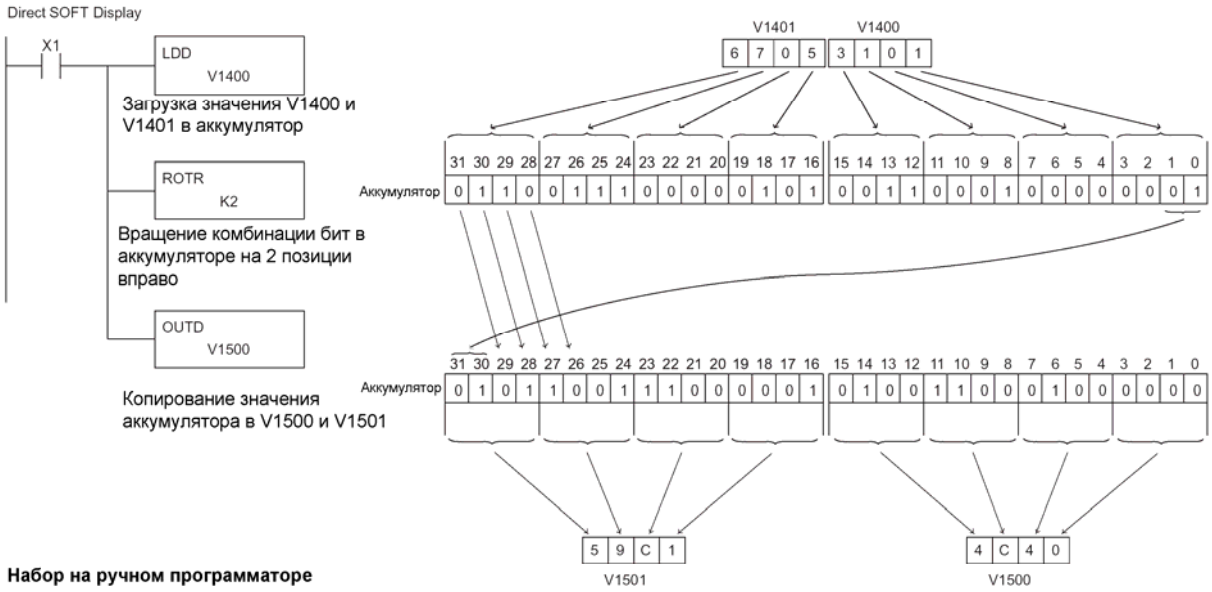


Rotate Right (ROTR)

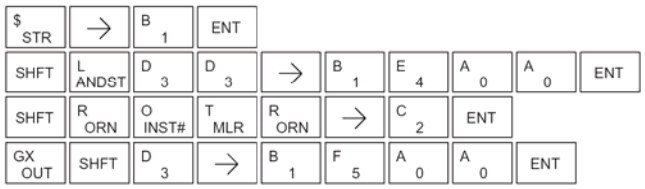
Rotate Right — 32-битная команда, которая вращает биты в аккумуляторе на указанное число (Аааа) мест вправо.

Тип данных операнда		DL06 Диапазон
	A	**ааа**
V-память	V	Смотри карту памяти
Константа	K	1-32

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор командой Load Double. Биты в аккумуляторе вращаются на 2 бита вправо, командой Rotate Right. Значение в аккумуляторе копируется в V1500 и V1501, командой Out Double.



Набор на ручном программаторе



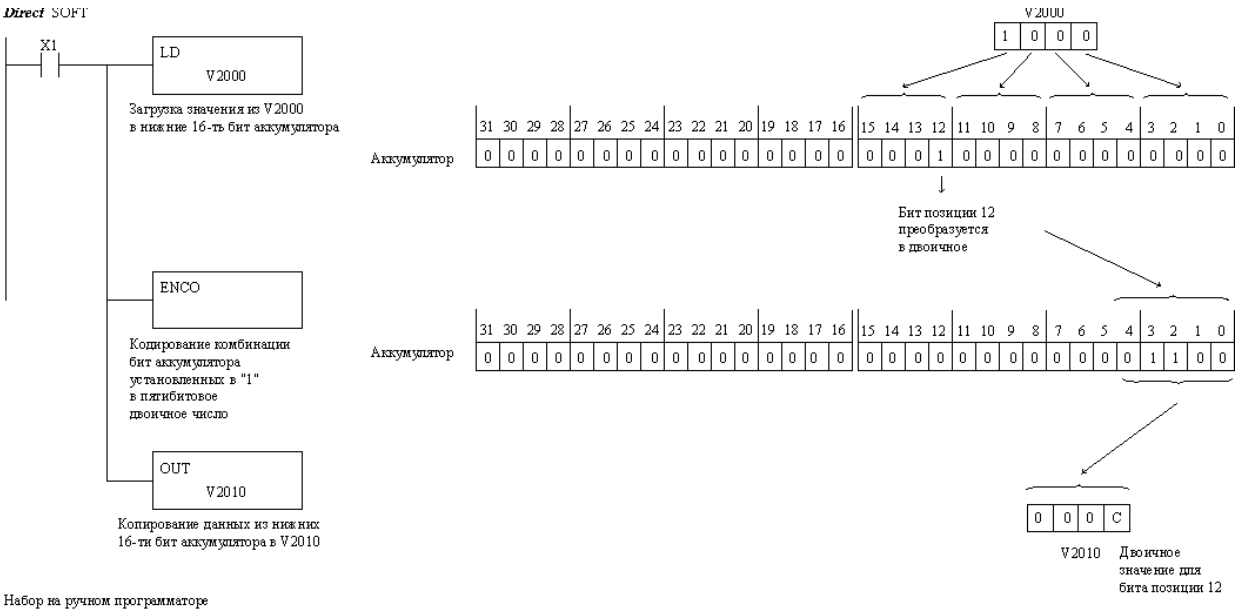
Encode (ENCO)

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 HEX (десятичное число 31) в аккумулятор. Если кодируемое значение 0000 или 0001, то команда поместит ноль в аккумулятор. Если кодируемое значение имеет более чем одну позицию бита, установленную в «1», то самая младшая «1» будет закодирована, и SP53 будет включен.

Флаги	Описание
SP53	«1», когда значение операнда больше, чем значение, с которым может работать аккумулятор
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» 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 V2000 загружается в аккумулятор командой Load. Позиция бита, установленная в «1» в аккумуляторе кодируется в соответствующее 5-битовое двоичное значение командой Encode. Значение в нижних 16 битах аккумулятора копируется в V2010, командой Out.



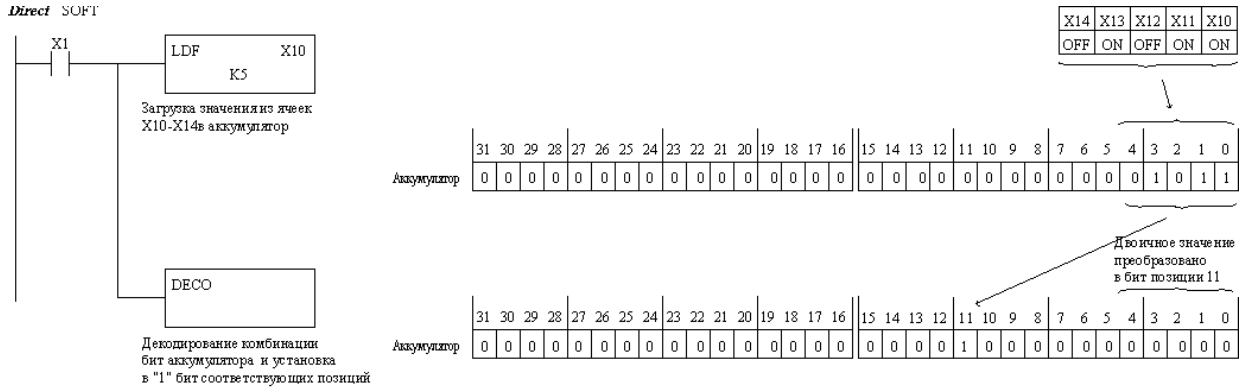
Набор на ручном программаторе

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	E 4	N TMR	C 2	O INST#	ENT			
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT

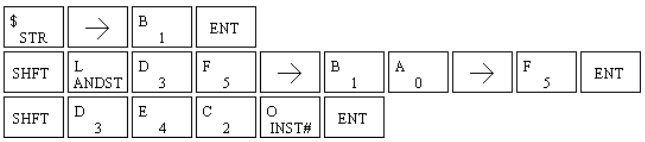
Decode (DECO)

Команда Decode декодирует 5-битовое двоичное значение 0-31 (0-1F HEX) в аккумуляторе, устанавливая соответствующую позицию бита в 1. Если аккумулятор содержит значение F (HEX), 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 значение декодируется.

В следующем примере, когда X1 включен, значение, составленное из дискретных ячеек X10-X14, загружается в аккумулятор командой Load Formatted. Пять бит в аккумуляторе декодируются, устанавливая соответствующую позицию бита в «1», командой Decode.



Набор на ручном программаторе



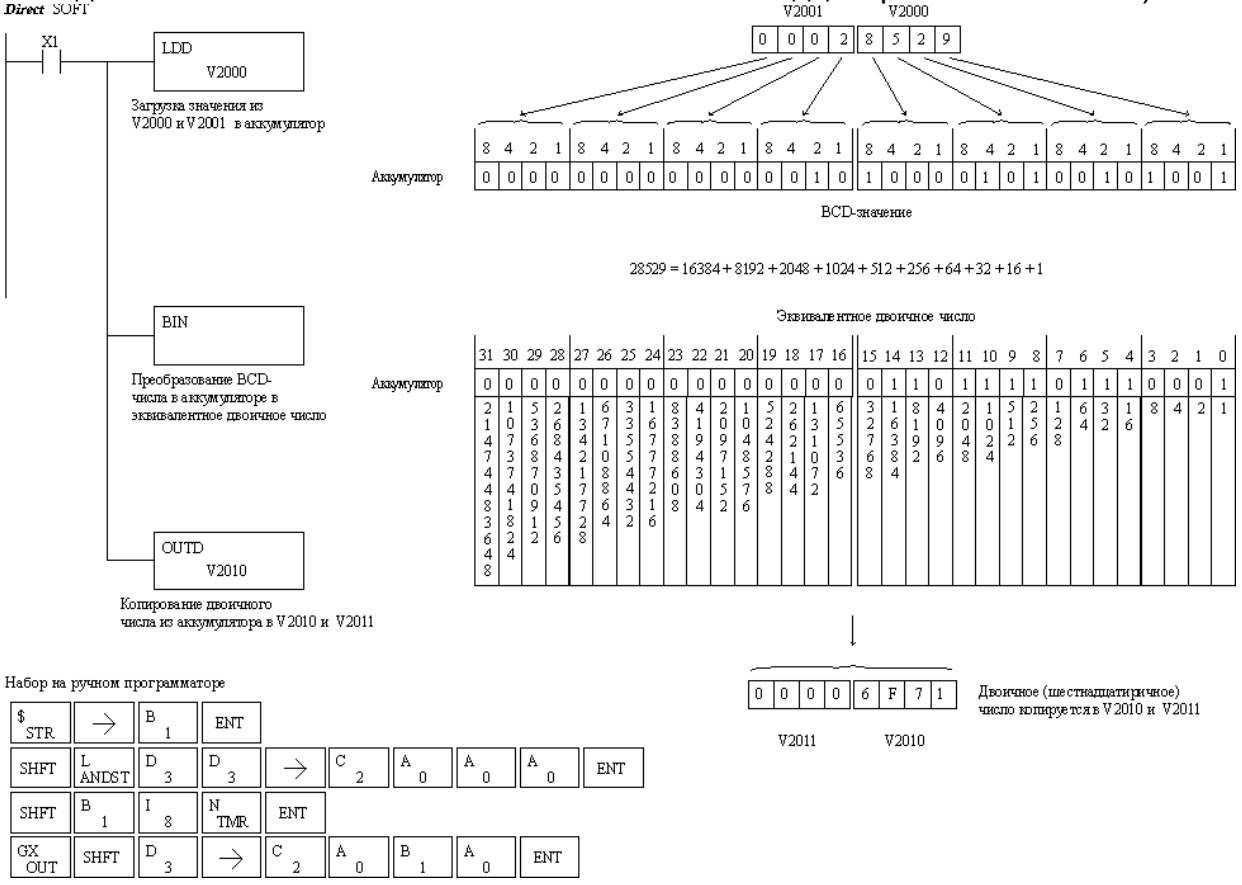
Команды преобразования чисел в аккумуляторе

Binary (BIN)

Команда Binary преобразует BCD значение в аккумуляторе в эквивалентное двоичное значение. Результат находится в аккумуляторе.

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP75	«1», когда выполняется BCD команда с не-BCD числом

В следующем примере, когда X1 включен, значение в V2000 и V2001 загружается в аккумулятор, командой Load Double. BCD значение в аккумуляторе преобразуется в двоичный (шестнадцатиричный) эквивалент, командой Binary. Двоичное значение в аккумуляторе копируется в V2010 и V2011 командой Out Double. (Ручной программатор будет показывать двоичное значение в V2010 и V2011 как шестнадцатиричное значение.)

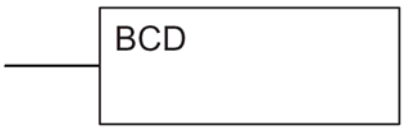


Набор на ручном программаторе

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	B 1	I 8	N TMR	ENT				
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT

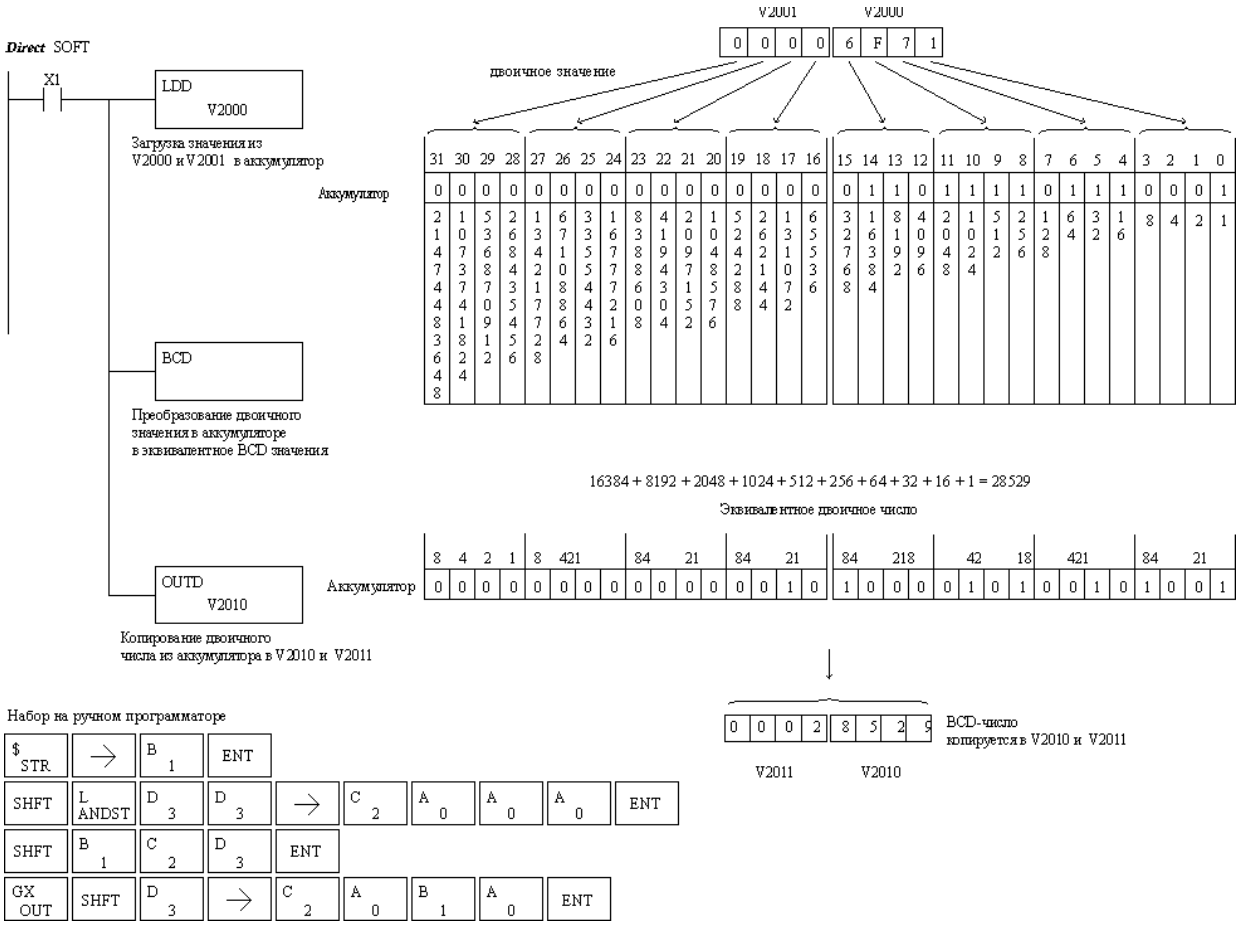
Binary Coded Decimal (BCD)

Команда Binary Coded Decimal преобразует двоичное значение в аккумуляторе в эквивалентное BCD значение. Результат находится в аккумуляторе.



Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное

В следующем примере, когда X1 включен, двоичное (HEX) значение в V2000 и V2001 загружается в аккумулятор, командой Load Double. Двоичное значение в аккумуляторе преобразуется в BCD эквивалентное значение, командой Binary Coded Decimal. BCD значение в аккумуляторе копируется в V2010 и V2011, командой Out Double.



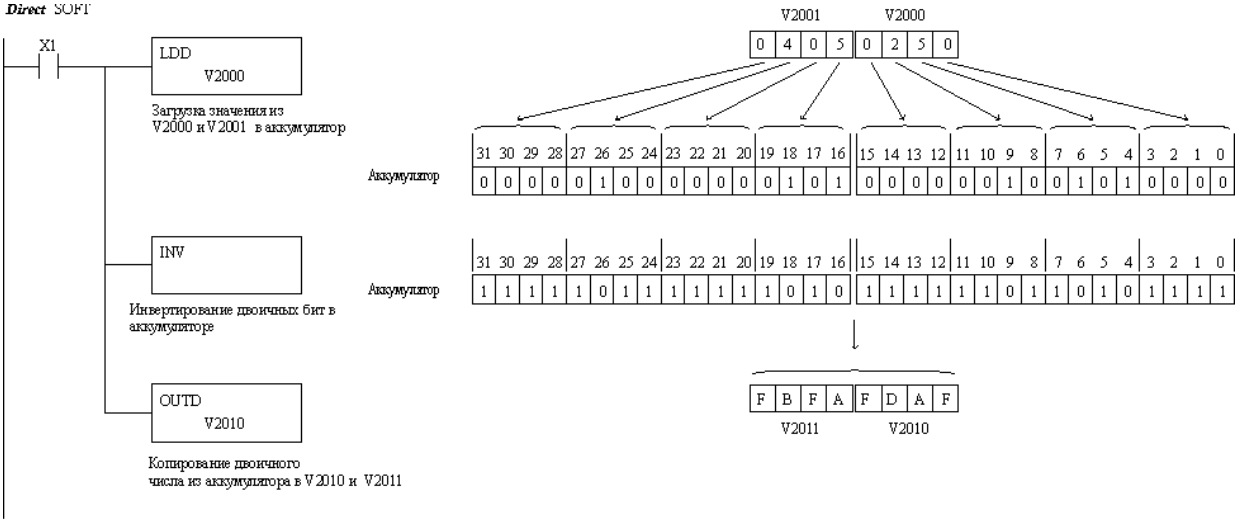
Набор на ручном программаторе

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	B 1	C 2	D 3	ENT				
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT

Invert (INV)

Команда Invert инвертирует 32-битное значение в обратном коде в аккумуляторе. Результат хранится в аккумуляторе.

В следующем примере, когда X1 включен, значение в V2000 и V2001 будет загружено в аккумулятор, командой Load Double. Значение в аккумуляторе инвертируется, командой Invert. Значение в аккумуляторе копируется в V2010 и V2011, командой Out Double.



Набор на ручном программаторе

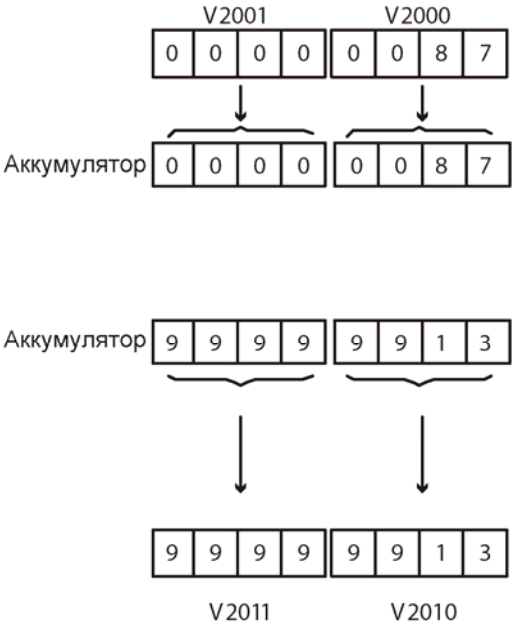
$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	I 8	N TMR	V AND	ENT					
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT	

Ten's Complement (BCDCPL)

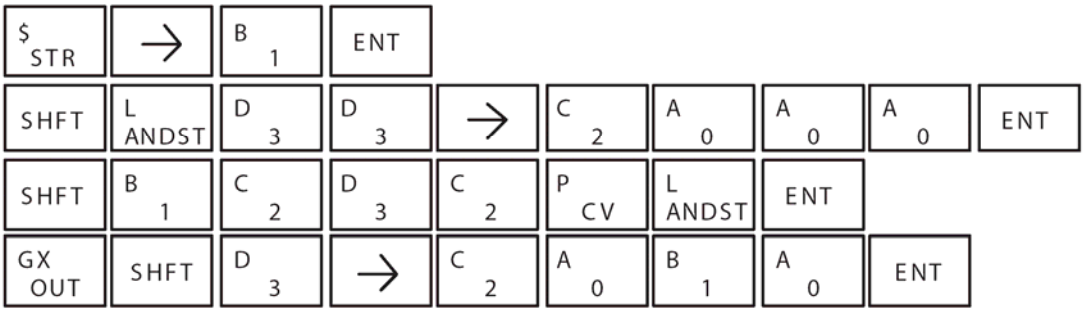
Команда Ten's Complement выполняет поразрядное дополнение до «10» 8-ми разрядного BCD значения в аккумуляторе. Результат находится в аккумуляторе. Вычисления производятся по следующей формуле:

100000000
– accumulator value
10's complement value

В следующем примере, когда X1 включен, значение в V2000 и V2001 загружается в аккумулятор, командой Load Double. 8-ми разрядное BCD значения в аккумуляторе доплняется до 10 командой Ten's Complement. Значение в аккумуляторе копируется в V2010 и V2011, командой Out Double.



Набор на ручном программаторе



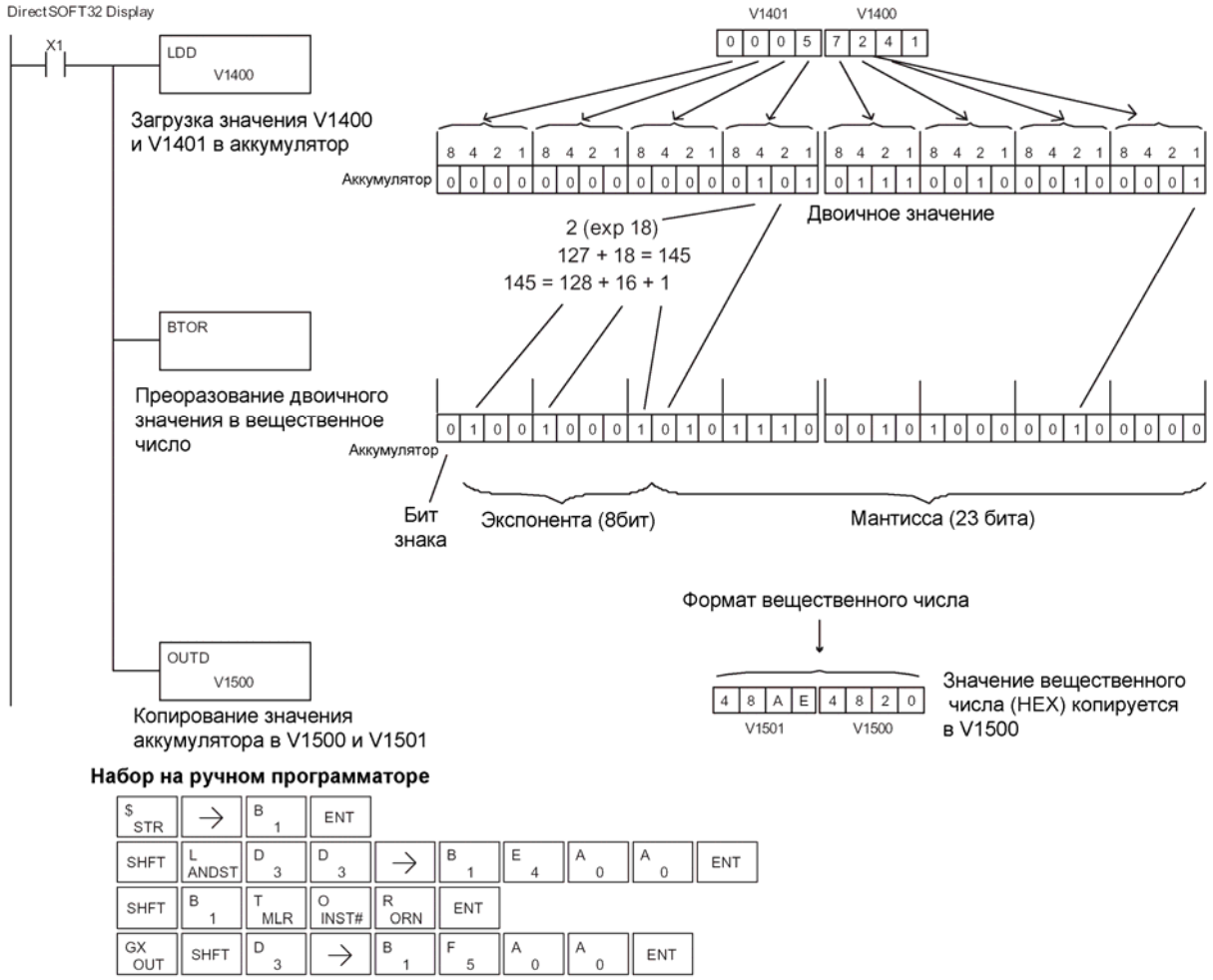
Binary to Real Conversion (BTOR)

Команда Binary-to-Real преобразует, двоичное значение в аккумуляторе в формат эквивалентного вещественного числа (с плавающей запятой). Результат хранится в аккумуляторе. И двоичное значение, и вещественное число может использовать все 32 бита аккумулятора.

ПРИМЕЧАНИЕ. Эта команда работает только с **двоичными** значениями. Она не будет работать с десятичными значениями со знаком.

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор, командой Load Double. Команда BTOR преобразует, двоичное значение в аккумуляторе в формат эквивалентного вещественного числа. Старший байт преобразуется в показатель (экспоненту) вещественного числа, добавлением 127 (десятичное число) к этому числу. Затем остающиеся биты копируются в мантиссу как показано на рисунке. Значение в аккумуляторе копируется в V1500 и V1501, командой Out Double.. Ручной программатор покажет значение ячеек V1500 и V1501 как шестнадцатиричное значение.



Набор на ручном программаторе

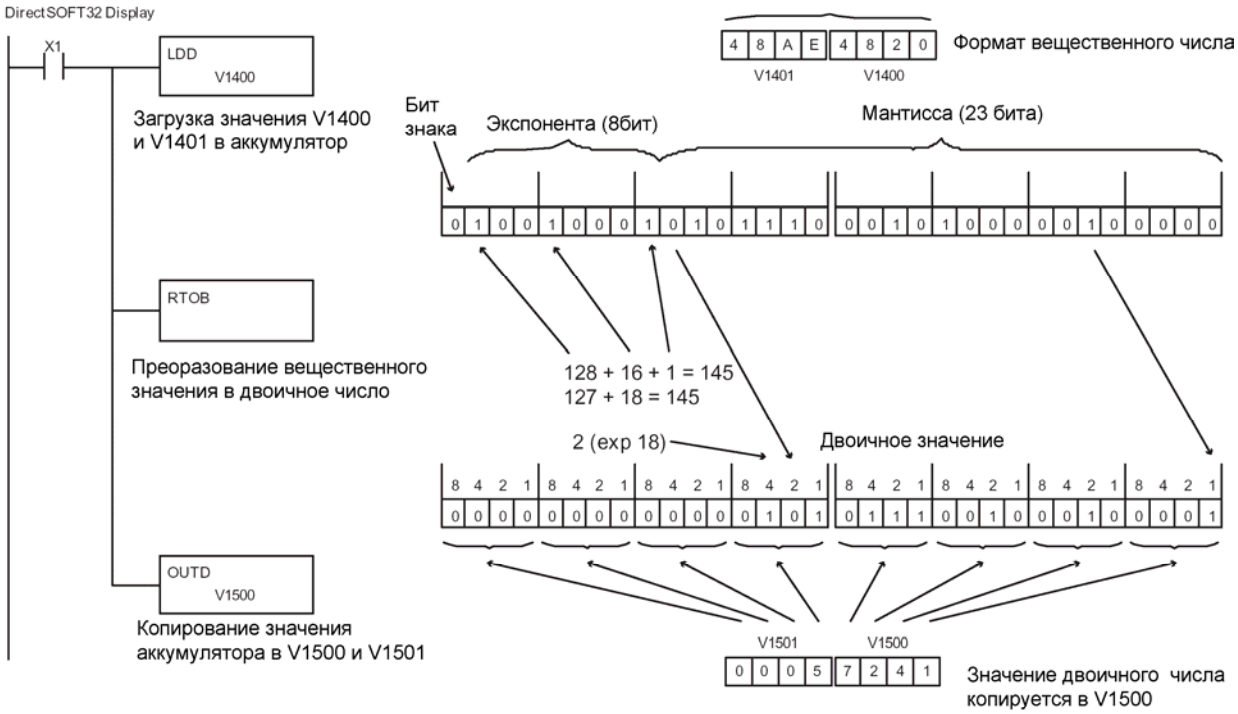
Real to Binary Conversion (RTOB)

Команда Real-to-Binary преобразует, вещественное число в аккумуляторе в двоичное значение. Результат хранится в аккумуляторе. И двоичное значение, и вещественное число может использовать все 32 бита аккумулятора.

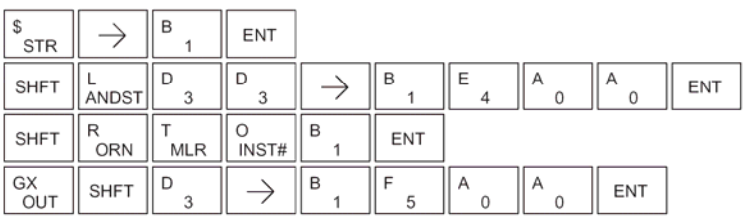
ПРИМЕЧАНИЕ 1: Дробная часть результата будет отсечена.
ПРИМЕЧАНИЕ 2: Если вещественное число отрицательное, оно становится десятичным числом со знаком.

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP72	«1» в любое время, когда значение в аккумуляторе - недопустимое число с плавающей запятой
SP73	«1», когда сложение или вычитание со знаком приводит к некорректному биту знака.
SP75	«1», когда число невозможно преобразовать в двоичный вид.

В следующем примере, когда X1 включен, значение в V1400 и V1401 будет загружено в аккумулятор, командой Load Double. Команда RTOB преобразует, вещественное число в аккумуляторе в двоичный формат. Значение в аккумуляторе копируется в V1500 и V1501, командой Out Double. Ручной программатор покажет значение ячеек V1500 и V1501 как шестнадцатиричное значение.



Набор на ручном программаторе



Radian Real Conversion (RADR)

Команда Radian Real Conversion преобразует вещественное число градусов в аккумуляторе в эквивалетное вещественное число радиан. Результат хранится в аккумуляторе.

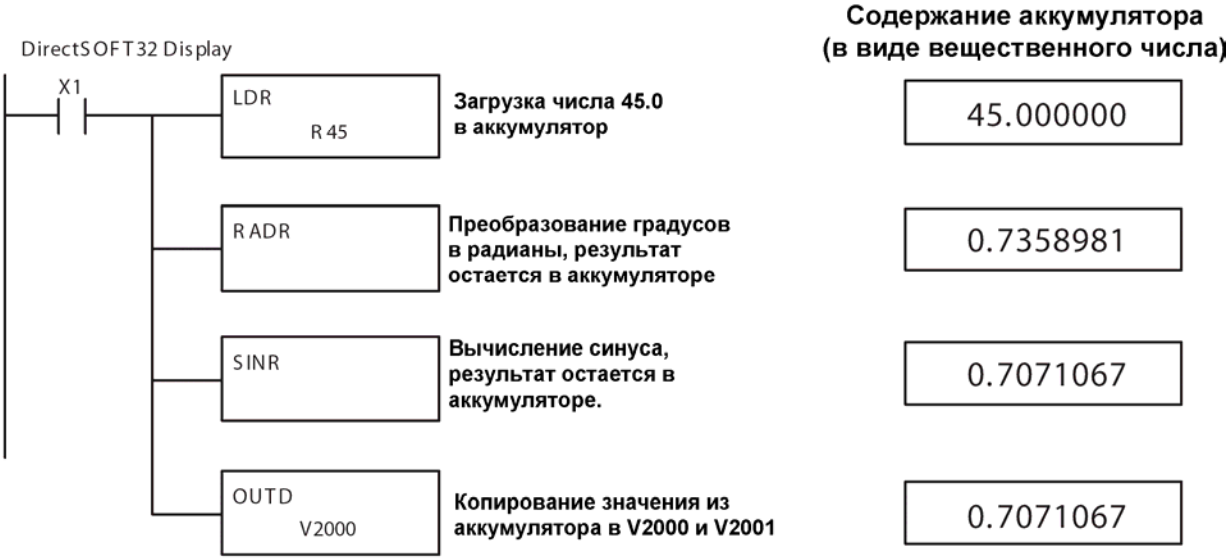
Degree Real Conversion (DEGR)

Команда Degree Real преобразует вещественное число радиан в аккумуляторе в эквивалетное вещественное число градусов. Результат хранится в аккумуляторе..

Две инструкции, описанные выше преобразовывают вещественные числа в аккумуляторе из градусов в радианы и обратно. Полный круг содержит 360 градусов и приблизительно 6.28 радиан. Оба варианта могут быть положительными и отрицательными вещественными числами, или значениями углов больше, чем полный круг. Эти функции очень полезны при использовании с трансцендентными тригонометрическими функциями.

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное
SP71	«1» в любое время, когда определено, что указатель ссылается на недопустимую ячейку V-памяти
SP72	«1» в любое время, когда значение в аккумуляторе - допустимое число с плавающей запятой
SP74	«1» в любое время, когда математическая операция с плавающей запятой приводит к ошибке переполнения.
SP75	«1», когда выполняется BCD команда с не-BCD числом

Следующий пример вычисляет синус 45 градусов. Так как эти трансцендентные функции функционируют только с вещественными числами, то мы выполняем команду LDR (загрузка вещественного числа) 45. Тригонометрические функции функционируют только в радианах, так что мы должны преобразовать градусы в радианы, используя команду RADR. После выполнения команды SINR, мы используем команду OUTD, чтобы переместить результат от аккумулятора в V-память. 32-битный результат требует применения команды Out Double.



ASCII-to-HEX (ATH)

Команда ASCII-to-HEX преобразует таблицу ASCII значений в таблицу шестнадцатиричных 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: загрузите число ячеек V-памяти для ASCII таблицы в первый уровень стека аккумулятора.

Шаг 2: загрузите начальную ячейку V-памяти для ASCII таблицы в аккумулятор. Этот параметр должен быть шестнадцатиричным значением.

Шаг 3: определите стартовую ячейку V-памяти (Vaaa) для шестнадцатиричной таблицы в команде ASCII-to-HEX.

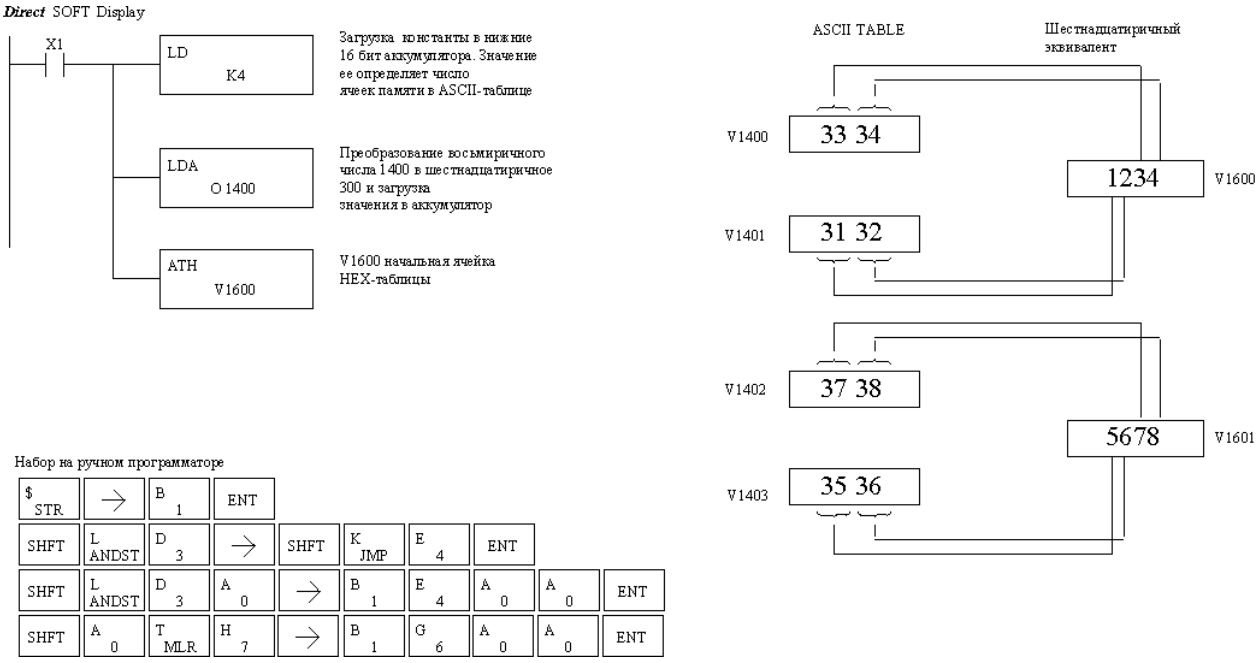
Полезная подсказка: для параметров, которые требуют шестнадцатиричных значений при ссылке на ячейки памяти, может быть использована команда LDA для преобразования восьмеричного адреса в шестнадцатиричный эквивалент и загрузки значения в аккумулятор.

Тип данных операнда	**DL06 Диапазон**
A	**aaa**
V-память V	Смотри карту памяти

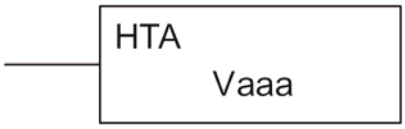
Флаги	**Описание**
SP53	«1», когда значение операнда больше, чем значение, с которым может работать аккумулятор

В примере на следующей странице, когда X1 включен, константа (K4) загружается в аккумулятор, командой Load Double и будет помещена в первый уровень стека аккумулятора, когда выполнится 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 HEX в ASCII			
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)



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: загрузите число ячеек V-памяти для шестнадцатиричной таблицы в первый уровень стека аккумулятора.

Шаг 2: загрузите начальную ячейку V-памяти для шестнадцатиричной таблицы в аккумулятор. Этот параметр должен быть шестнадцатиричным значением.

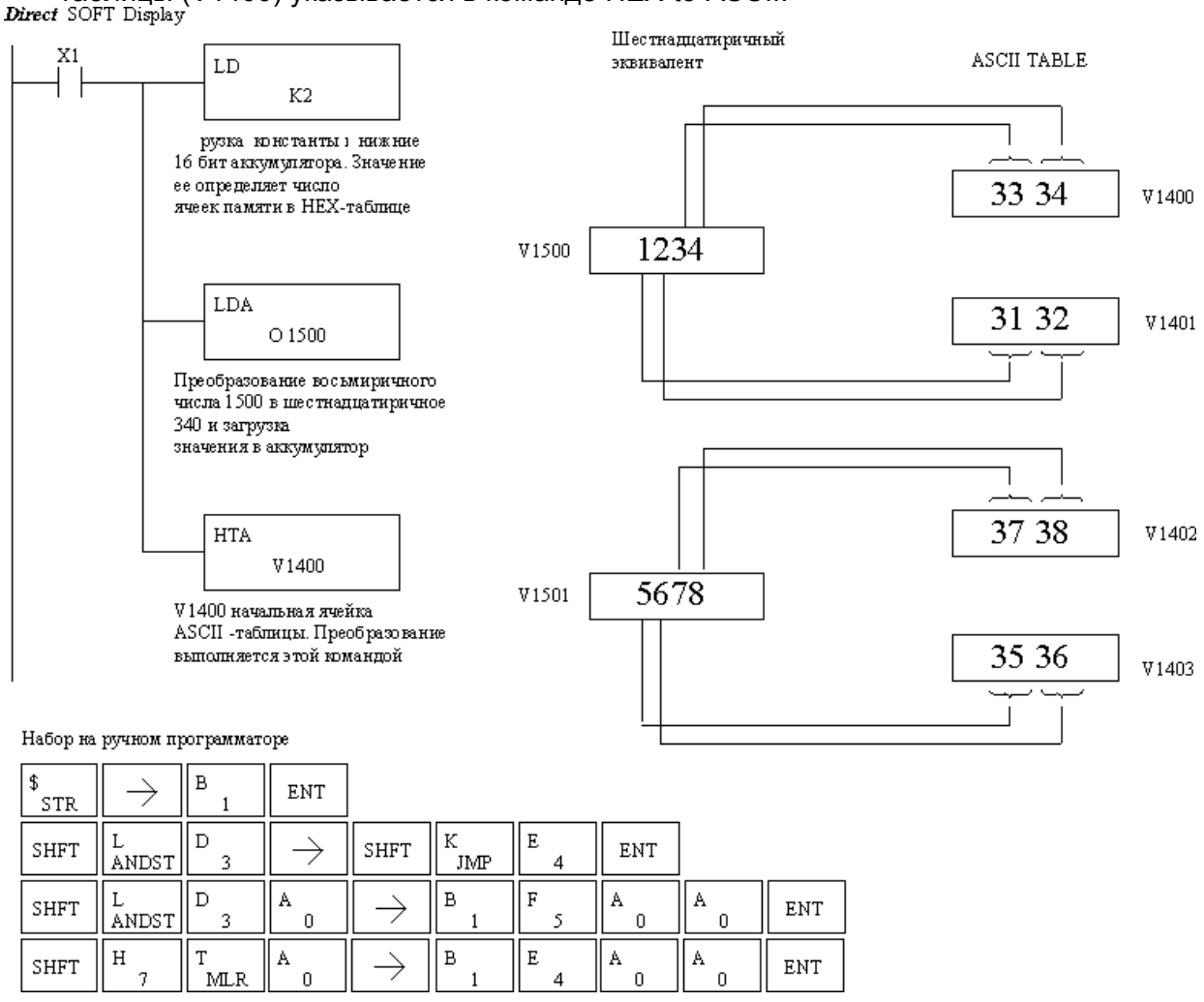
Шаг 3: определите стартовую ячейку V-памяти (Vaaa) для ASCII таблицы в команде HEX-to-ASCII.

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 загрузки значения в аккумулятор.

Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти

Флаги	Описание
SP53	«1», когда значение операнда больше, чем значение, с которым может работать аккумулятор

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

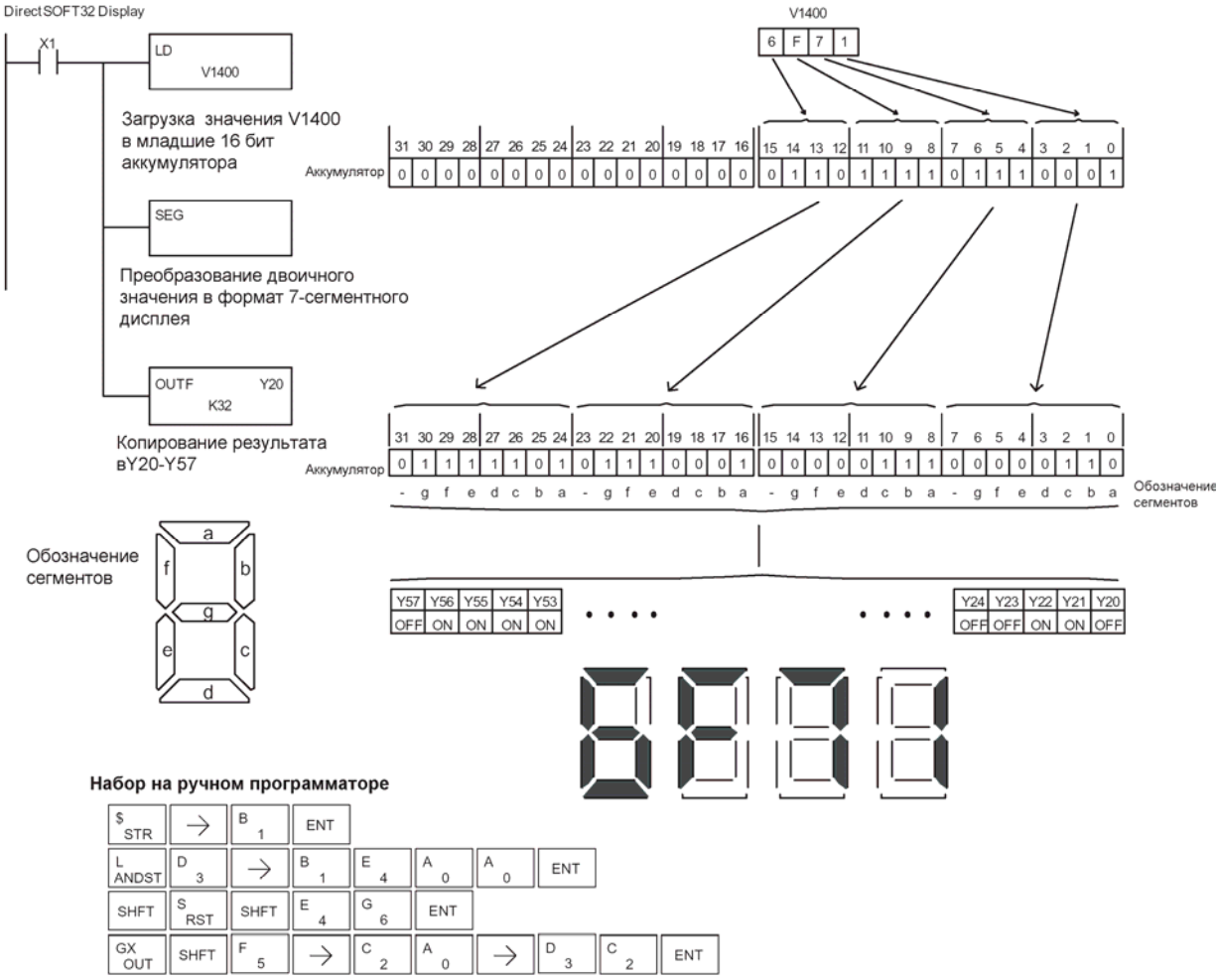
$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	SHFT	K JMP	E 4	ENT		
SHFT	L ANDST	D 3	A 0	→	B 1	F 5	A 0	A 0	ENT
SHFT	H 7	T MLR	A 0	→	B 1	E 4	A 0	A 0	ENT

Возможные значения ASCII для преобразования HEX в ASCII			
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)

Команда BCD/Segment преобразует 4-х разрядное шестнадцатиричное значение в аккумуляторе в формата 7-сегментного дисплея. Результат сохраняется в аккумуляторе.

В следующем примере, когда X1 включен, значение в V1400 загружается в младшие 16 бит аккумулятора, используя команду Load. Двоичное значение (HEX) в аккумуляторе преобразуется в семь формата 7-сегментного дисплея, используя команду Segment. Комбинация бит в аккумуляторе копируется в выходные ячейки Y20-Y57, используя команду Out Formatted.



Набор на ручном программаторе

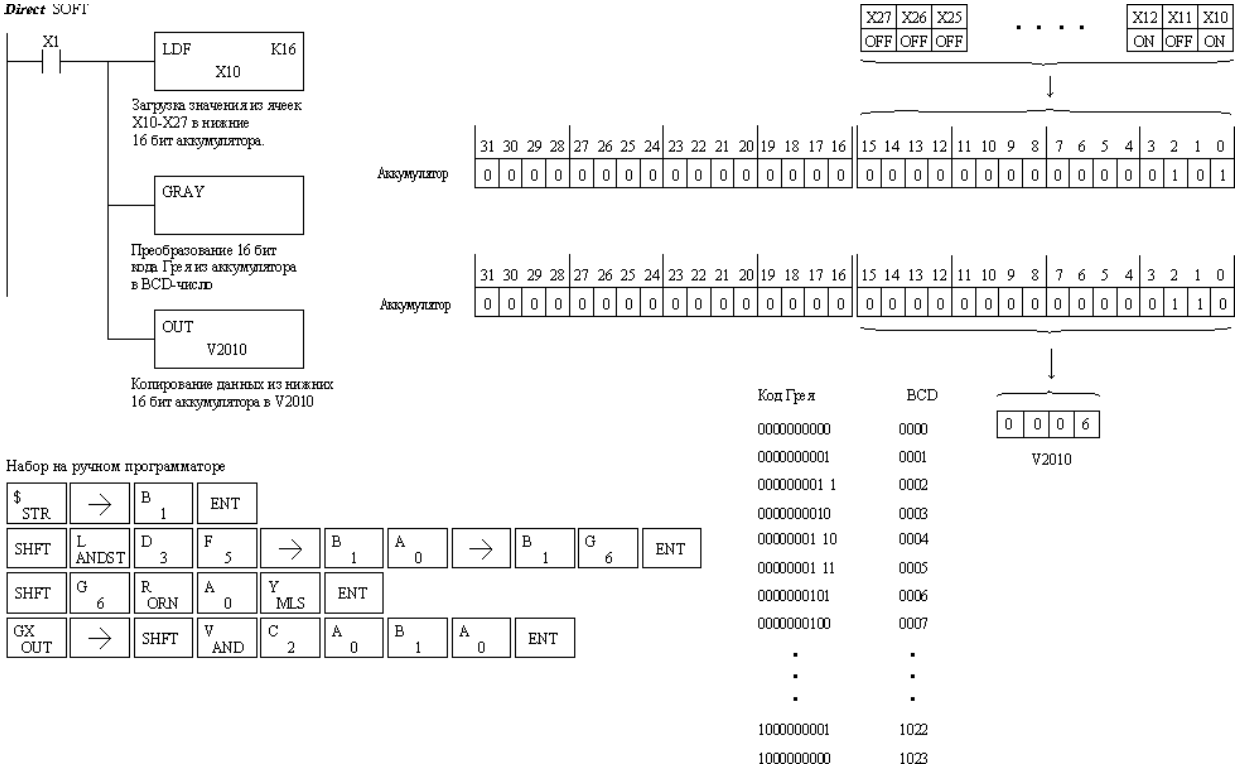
$ STR	→	B 1	ENT					
L ANDST	D 3	→	B 1	E 4	A 0	A 0	ENT	
SHFT	S RST	SHFT	E 4	G 6	ENT			
GX OUT	SHFT	F 5	→	C 2	A 0	→	D 3	C 2	ENT

Gray Code (GRAY)

Команда Gray Code преобразует 16-битное значение в коде Грэя (двоичный циклический код) в BCD значение. BCD преобразование требует 10 бит аккумулятора. Верхние 22 бита установлены в «0». Эта команда создана для использования с устройствами (обычно энкодерами), которые используют код Грэя для схем оцифровки. Команда Gray Code непосредственно преобразует число в коде Грэя в BCD число для устройств, имеющих разрешение 512 или 1024 точек на цикл. Чтобы использовать устройство, имеющее разрешение 360 точек на цикл, Вы должны вычесть BCD значение 76 из преобразованного значения. Для устройства, имеющего разрешение 720 точек на цикл, Вы должны вычесть BCD значение 152.

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное

В следующем примере, когда X1 включен, двоичное значение, представленное в X10-X27 загружается в аккумулятор, командой Load Formatted. Значение в коде Грэя в аккумуляторе преобразуется в BCD, командой Gray Code. Значение в нижних 16 битах аккумулятора копируется в V2010.



Shuffle Digits (SFLDGT)

Команда Shuffle Digits меняет местами максимум 8 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: загрузите меняемые значения (цифры) в первый уровень стека аккумулятора.

Шаг 2: загрузите в аккумулятор порядок, по которому будут переставлены цифры.

Шаг 3: вставьте команду Shuffle Digits.

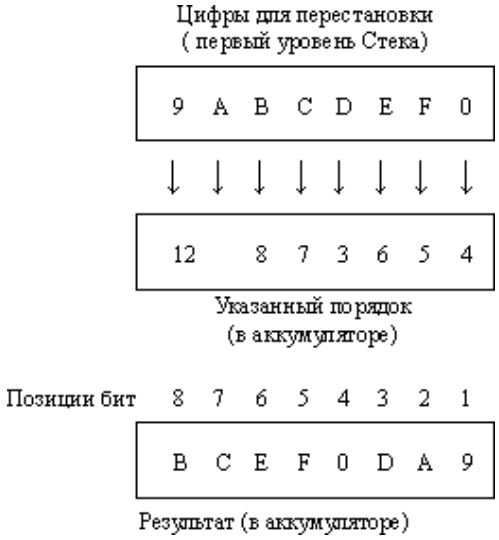
Примечание: если число, используемое для определения порядка, содержит 0 или 9-F, то соответствующая позиция будет установлена в 0.

Примечание: если число, используемое для определения порядка, содержит идентичные числа, то действительно старшее из них.

Флаги	Описание
SP63	«1», когда в результате выполнения команды значение в аккумуляторе является нулем
SP70	«1» в любое время, когда значение в аккумуляторе отрицательное

Блок-схема Shuffle Digits

Могут быть переставлены максимум 8 цифр. Разряды в первом уровне стека аккумулятора определяют цифры, которые будут переставлены. Они соответствуют разрядам в акумуляторе, которые определяют порядок, по которому будут переставлены цифры. Цифры меняются местами и результат остается в аккумуляторе.

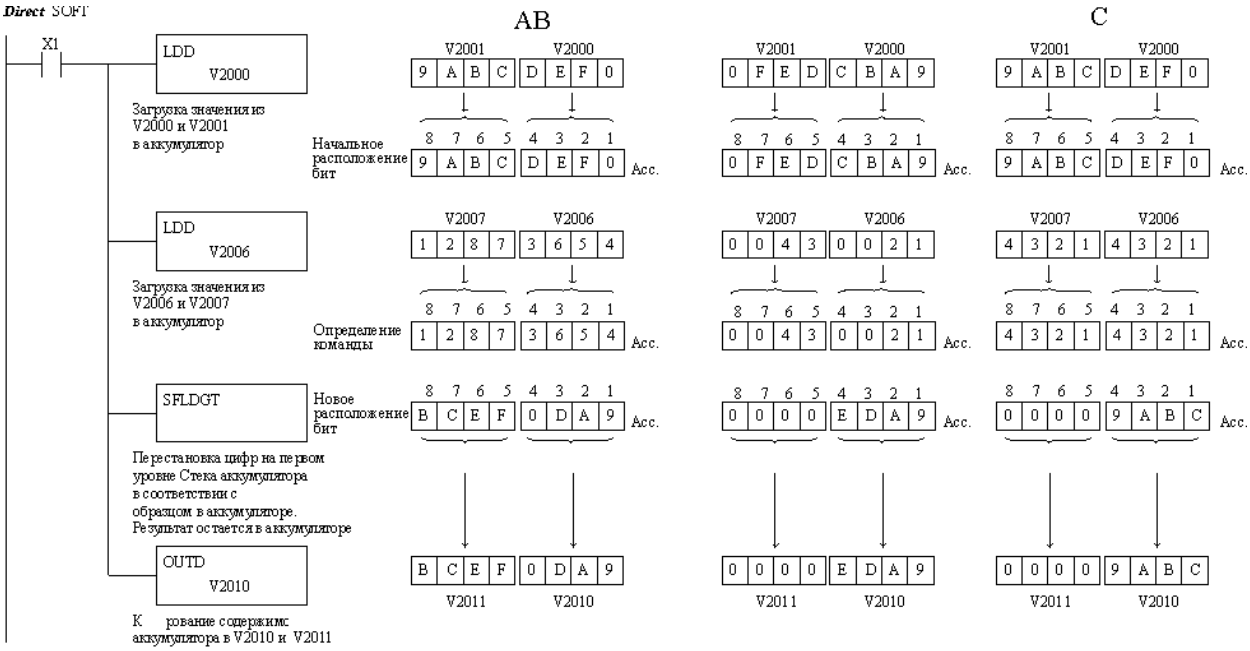


В следующем примере, когда X1 включен, значение в первом уровне стека аккумулятора будет реорганизовано в порядке, определенном значением в аккумуляторе.

Пример A показывает, как Shuffle Digits работает, когда 0 или 9-F не используются при определении порядка перестановки цифр, а также не имеется идентичных чисел при определении порядка.

Пример B показывает, как Shuffle Digits работает, когда 0 или 9-F используются при определении порядка перестановки цифр. Обратите внимание, что при выполнении команды Shuffle Digits, разрядные позиции в первой ячейке стека, которые имели соответственно 0 или 9-F в аккумуляторе (определенный порядок), устанавливаются в «0».

Пример C показывает, как Shuffle Digits работает, когда идентичные числа используются при определении порядка перестановки цифр. Обратите внимание, что при выполнении команды Shuffle Digits, в результате используется старшее из идентичных чисел.



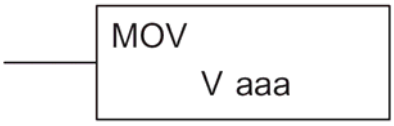
Набор на ручном программаторе

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	G 6	ENT
SHFT	S RST	SHFT	F 5	L ANDST	D 3	G 6	T MLR	ENT	
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT	

Табличные команды

Move (MOV)

Команда Move перемещает значения 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: загрузите число перемещаемых ячеек V-памяти в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным значением.

Шаг 2: загрузите начальную ячейку V-памяти для ячеек, перемещаемых в аккумулятор. Этот параметр должен быть шестнадцатеричным значением.

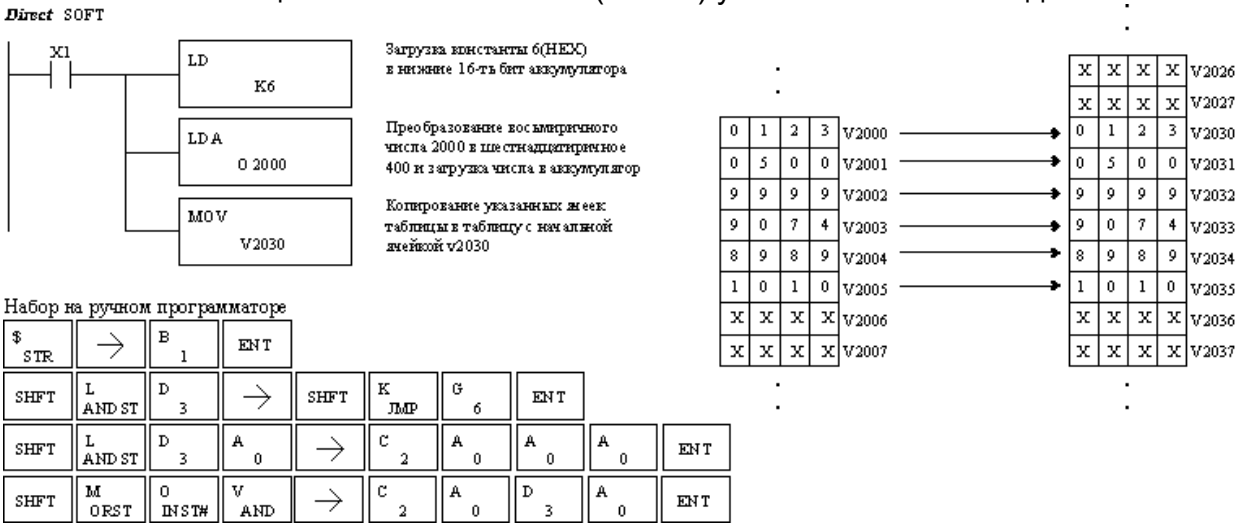
Шаг 3: вставьте команду Move, которая указывает стартовую ячейку V-памяти (Vaaa) для таблицы места назначения.

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 загрузки значения в аккумулятор.

Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти

Флаги	Описание
SP53	«1», когда значение операнда больше, чем значение, с которым может работать аккумулятор

В следующем примере, когда X1 включен, константа (K6) загружается в аккумулятор, используя команду Load. Это значение указывает длину таблицы и помещается в первой ячейке стека после выполнения команды Load Address. Восьмеричный адрес 2000 (V2000), который является начальной ячейкой для исходной таблицы, загружается в аккумулятор. Ячейка таблицы места назначения (V2030) указывается в команде Move.

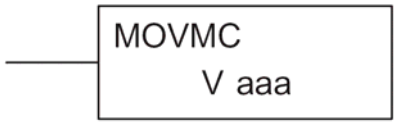


Набор на ручном программаторе

$ STR	→	B 1	ENT					
SHFT	L AND ST	D 3	→	SHFT	K JMP	G 6	ENT	
SHFT	L AND ST	D 3	A 0	→	C 2	A 0	A 0	A 0	ENT
SHFT	M ORST	O INST#	V AND	→	C 2	A 0	D 3	A 0	ENT

Move Memory Cartridge(MOVMC) /Load Label (LDLBL)

Команда Move Memory Cartridge используется для копирования данных между V-памятью и памятью программы. Команда Load Label используется только с командой MOVMC при копировании данных из памяти программы в V-память.



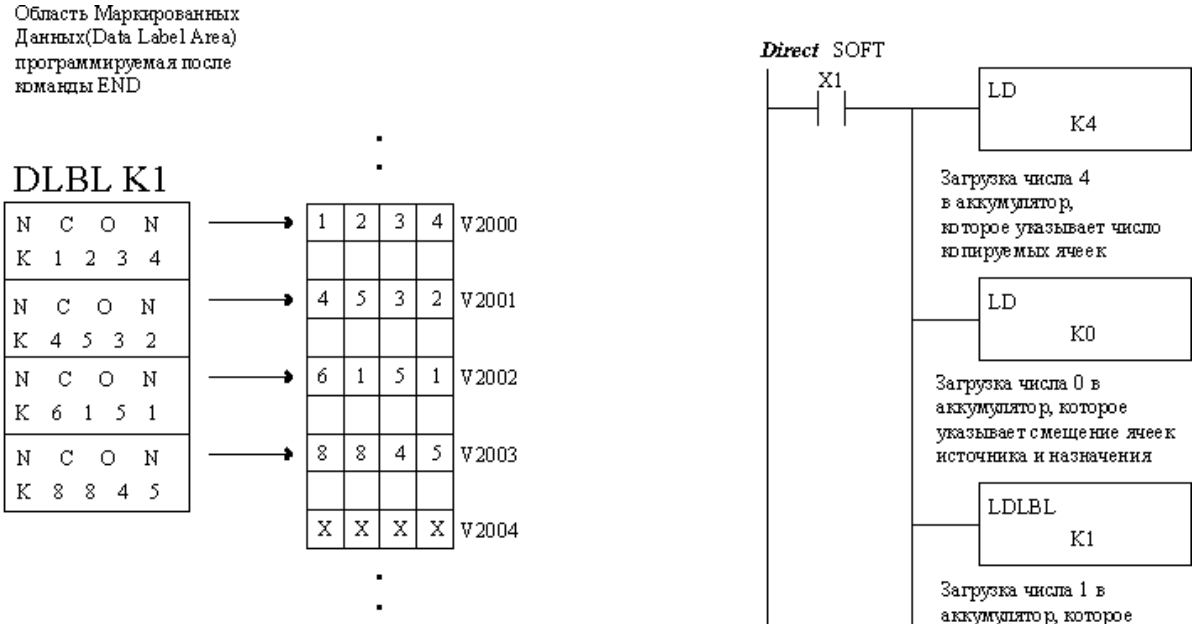
Чтобы копировать данные между V-памятью и памятью программы, функциональные параметры загружаются в первые два уровня стека аккумулятора и аккумулятор двумя дополнительными командами. Ниже показаны шаги, необходимые для программирования функций Move Memory Cartridge и Load Label.

Шаг 1: загрузите число копируемых слов во второй уровень стека аккумулятора.
Шаг 2: загрузите смещение для области метки данных в памяти программы и начальный блок V-памяти в первый уровень стека аккумулятора.
Шаг 3: загрузите исходную метку данных (LDLBL Kaaa) в аккумулятор при копировании данных из памяти программы в V-память. Загрузите исходный адрес в аккумулятор при копировании данных из V-памяти в память программы. Это адрес, откуда значение будет скопировано. Если исходный адрес является ячейкой V-памяти, то значение должно быть введено в шестнадцатиричном коде.
Шаг 4: вставьте команду MOVMC, которая указывает адрес назначения (Aaaa). Это адрес, куда значение будет скопировано.

Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти

Копирование данных из области меток данных в V-память

В следующем примере данные копируются из области меток данных (Data Label Area) в V-память. Когда X1 включен, константа (K4) загружается в аккумулятор, используя команду Load. Это значение указывает длину таблицы и помещается во второй ячейке стека после выполнения следующих команд Load и Load Label (LDLBL). Константа (K0) загружается в аккумулятор, используя команду 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, используя команду LDLBL. Команда MOVMC указывает стартовую ячейку адресата и выполняет копирование данных из области метки данных в V-память.



Набор на ручном программаторе

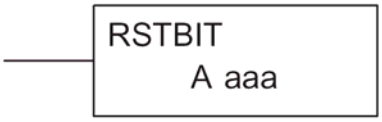
$ STR	→	B 1	ENT				
SHFT	L ANDST	D 3	→	SHFT	K JMP	E 4	ENT
SHFT	L ANDST	D 3	→	SHFT	K JMP	A 0	ENT
SHFT	L ANDST	D 3	L ANDST	B 1	L ANDST	→	B 1	ENT
SHFT	M ORST	O INST#	V AND	M ORST	C 2	→	C 2	A 0	A 0	A 0	ENT

SETBIT

Команда Set Bit устанавливает отдельный бит в единицу в указанном диапазоне V-памяти.

RSTBIT

Команда Reset Bit сбрасывает отдельный бит в ноль в указанном диапазоне V-памяти.



Следующее описание подходит к обоим табличным командам Set Bit и Reset Bit .

Шаг 1: загрузите длину таблицы(число ячеек V-памяти) в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным значением в пределах от 0 до FF.

Шаг 2: загрузите адрес начальной ячейки таблицы в V-памяти. Этот параметр должен быть шестнадцатиричным значением. Вы можете использовыать команду LDA для преобразования восьмиричного адреса в шестнадцатиричный.

Шаг 3: вставьте команду Set Bit или Reset Bit. Значение по данному адресу в команде определяет номер бита, который Вы хотите установить или сбросить. Номер бита должен быть восьмиричным числом и первый бит в таблице имеет номер «0».

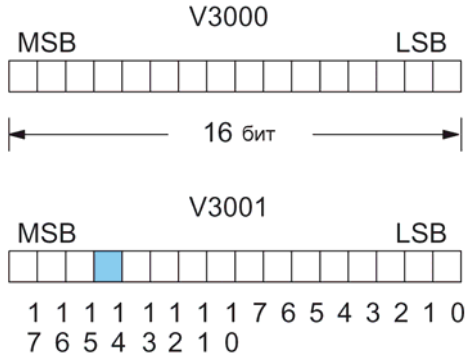
Полезный совет: — Помните, что каждая ячейка V-памяти содержит 16 бит. Таким образом, биты первого слова таблицы пронумерованы от 0 до 17 в восьмеричном виде. Например, если длина таблицы - шесть слов, то 6 слов = (6 x 16) бит, = 96(десятичное число) бит, или 140(восьмеричное). Допустимый диапазон числа бит для данной ссылки был бы от 0 до 137(восьмеричное число). SP 53 будет установлен, если указанный бит будет находится вне диапазона таблицы.

Тип данных операнда		DL06 Диапазон
	A	**aaa**
V-память	V	Смотри карту памяти

Флаги	Описание
SP53	«1», когда значение операнда больше, чем значение, с которым может работать аккумулятор

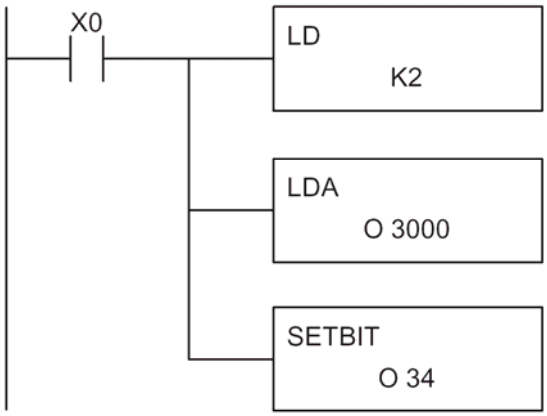
ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

Например, предположим, что мы имеем таблицу, начинающуюся с V3000, и длиной в два слова, как показано справа Каждое слово в таблице содержит 16 битов, или от 0 до 17 в восьмеричном виде. Чтобы устанавливать бит 12 в втором слове, мы используем восьмеричную ссылку (бит 14). Затем мы вычисляем восьмеричный адрес бита от начала таблицы, как 17+14=34 (восьмеричное). На следующей странице показана программа, устанавливающая этот бит в “1”.



В этом примере релейной схемы, мы используем вход X0, чтобы начать выполнение операцию по установки бита. Сначала, мы загрузим длину таблицы (2 слова) в стек аккумулятора. Затем, мы загружаем начальный адрес в аккумулятор. Затем, используя команду LDA, преобразуем V3000, восьмиричное число, в шестнадцатеричный вид. И в заключение, мы используем команду Set Bit (или Reset Bit) и указываем восьмеричный адрес бита (бит 34), как ссылку на таблицу.

Direct SOFT Display32



Загрузка константы 2 (Hex) в младшие 16 бит аккумулятора

Преобразование восьмиричного числа 3000 в шестнадцатиричное и загрузка его в аккумулятор. Это число является начальным адресом таблицы.

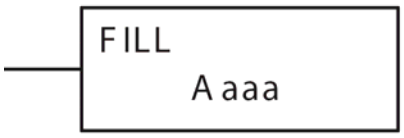
Установка бита 34(восьмиричное число) в таблице в значение “1”

Набор на ручном программаторе

$ STR	→	A 0	ENT					
SHFT	L ANDST	D 3	→	PREV	C 2	ENT		
SHFT	L ANDST	D 3	A 0	→	D 3	A 0	A 0	A 0	ENT
X SET	SHFT	B 1	I 8	T MLR	NEXT	D 3	E 4	ENT

Fill (FILL)

Команда Fill заполняет таблицу до 255 ячеек V-памяти значением (Aaaa), который является ячейкой V-памяти или 4-х разрядной константой. Функциональные параметры загружены в первый уровень стека и аккумулятор двумя дополнительными командами.



Ниже даны шаги необходимые при использовании функции Fill.

Шаг 1:— Загрузите число ячеек V-памяти который необходимо заполнить в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным числом в пределах от 0 до FF.

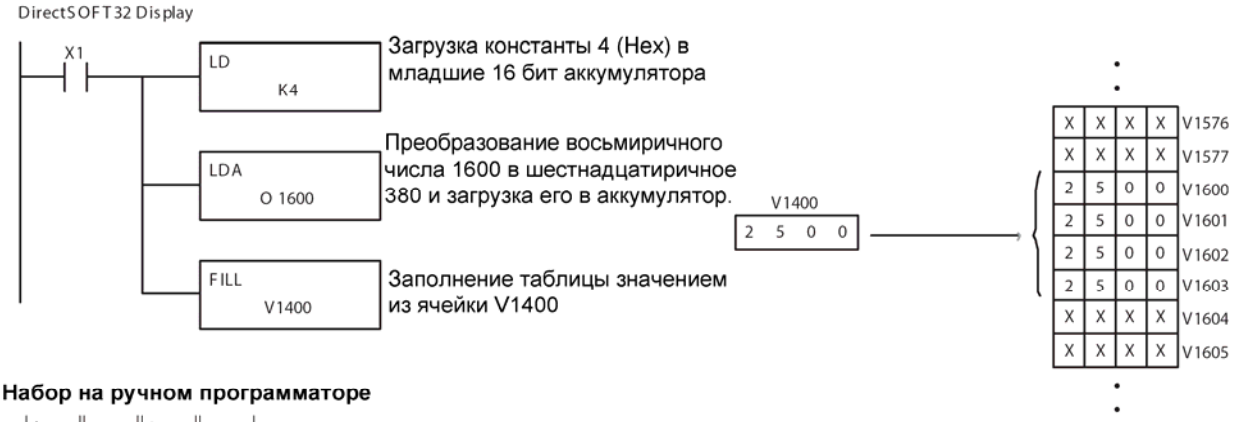
Шаг 2:— Загрузите начальный адрес ячейки V-памяти для таблицы в аккумулятор. Этот параметр должен быть шестнадцатиричным числом.

Шаг 3:— Вставьте команду Fill, которая определяет значение для заполнения таблицы.

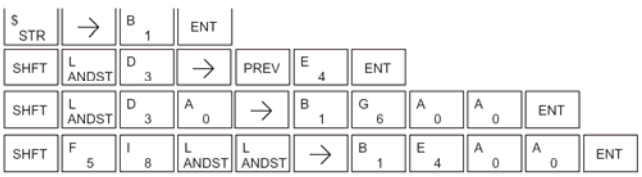
Полезный совет: — Вы можете использовать команду LDA для преобразования восьмиричного адреса в шестнадцатиричный.

Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-FFFF

В следующем примере, когда X1 включен, константа загружается в аккумулятор, используя команду Load. Это значение определяет длину таблицы и помещается в первый уровень стека аккумулятора, когда команда Load Address будет выполнена. Восьмеричный адрес 1600 (V1600) - начальная ячейка таблицы, загружается в аккумулятор, используя инструкцию Load Address. Значение для заполнения таблицы (V1400) определяется в команде Fill.

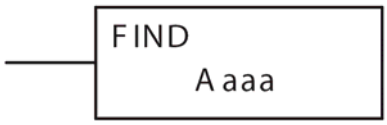


Набор на ручном программаторе



Find (FIND)

Команда Find используется для поиска заданной величины в таблице до 255 ячеек V-памяти. Функциональные параметры загружены в первый и второй уровни стека аккумулятора и сам аккумулятор тремя дополнительными командами. Ниже перечислены шаги необходимые при использовании функции Find.



Шаг 1:— Загрузите число ячеек V-памяти во второй уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным числом в пределах от 0 до FF.

Шаг 2:— Загрузите начальный адрес ячейки V-памяти для таблицы в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным числом.

Шаг 3:— Загрузите в аккумулятор смещение ячейки, с которой начинается поиск. Этот параметр должен быть шестнадцатиричным числом.

Шаг 4:— Вставьте команду Find определяющую значение, которое должно быть найдено в таблице.

Результат:—

Смещение от начального адреса до первой ячейки V-памяти, которая содержит значение поиска (шестнадцатиричное значение), возвращается в аккумулятор. SP53 будет установлен на том случае, если адрес в смещении определен вне таблицы, или значение не найдено. Если значение не найдено в аккумулятор будет возвращено значение «0».

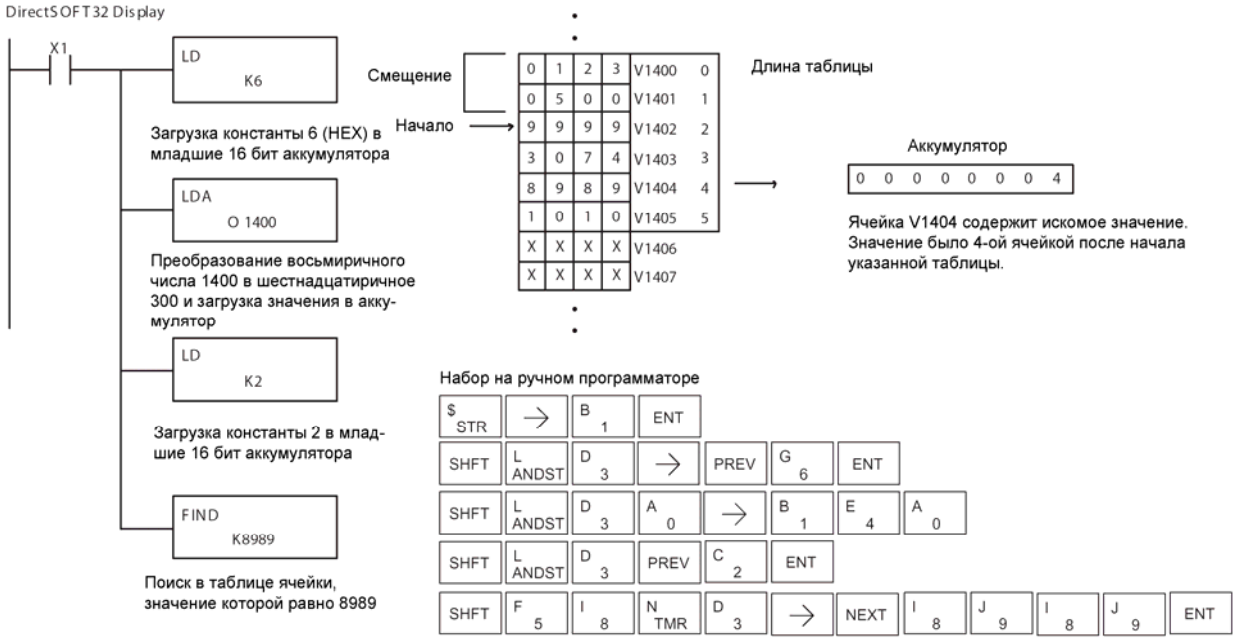
Полезный совет: — Вы можете использовать команду LDA для преобразования восьмиричного адреса в шестнадцатиричный.

Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Константа	K	0-FFFF

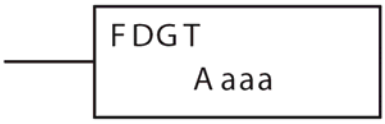
Флаги	Описание
SP53	«1», когда не найдено искомое значение в указанной таблице

ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги. Указатель для этой команды может начинаться с 0 и сохраняется в аккумуляторе

В следующем примере, когда X1 включен, константа загружается в аккумулятор, используя команду Load. Это значение определяет длину таблицы и помещается во второй уровень стека, при выполнении следующих команд Load Address и Load. Восьмеричный адрес 1400 (V1400) - начальная ячейка таблицы загружается в аккумулятор. Это значение помещается в первый уровень стека аккумулятора, при выполнении следующей команды Load. Смещение (K2) загружается в младшие 16 бит аккумулятора, используя команду Load. Значение, которое будет найдено в таблице определено в команде Find. Если найдено значение равное значению поиска, смещение (от начальной ячейки таблицы), где значение размещено, сохраняется в аккумуляторе.



Find Greater Than (FDGT)



Команда Find Greater Than используется для поиска, первого местонахождение значения в таблице V-памяти, которое является больше заданной величины (Aaaa), которая может быть или ячекой V-памяти или 4-разрядная константа. Функциональные параметры загружены в первый уровень стека и аккумулятор двумя дополнительными инструкциями.

Ниже перечислены шаги, необходимые для работы с командой Find Greater Than.

Шаг 1:— Загрузите длину таблицы в V-памяти в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным числом в пределах от 0 до FF.

Шаг 2:— Загрузите начальный адрес ячейки V-памяти для таблицы в аккумулятора. Этот параметр должен быть шестнадцатиричным числом.

Шаг 3:— Вставьте команду FDGT определяющую значение больше указанного.

Результат:— Смещение от начального адреса до первой ячейки V-памяти, которая содержит значение больше, чем указанное (шестнадцатиричное значение), возвращается в аккумулятор. SP53 будет установлен в том случае, если адрес в смещении определен вне таблицы, или значение не найдено. Если значение не найдено в аккумулятор будет возвращено значение «0».

Полезный совет: — Вы можете использовать команду LDA для преобразования восьмиричного адреса в шестнадцатиричный.

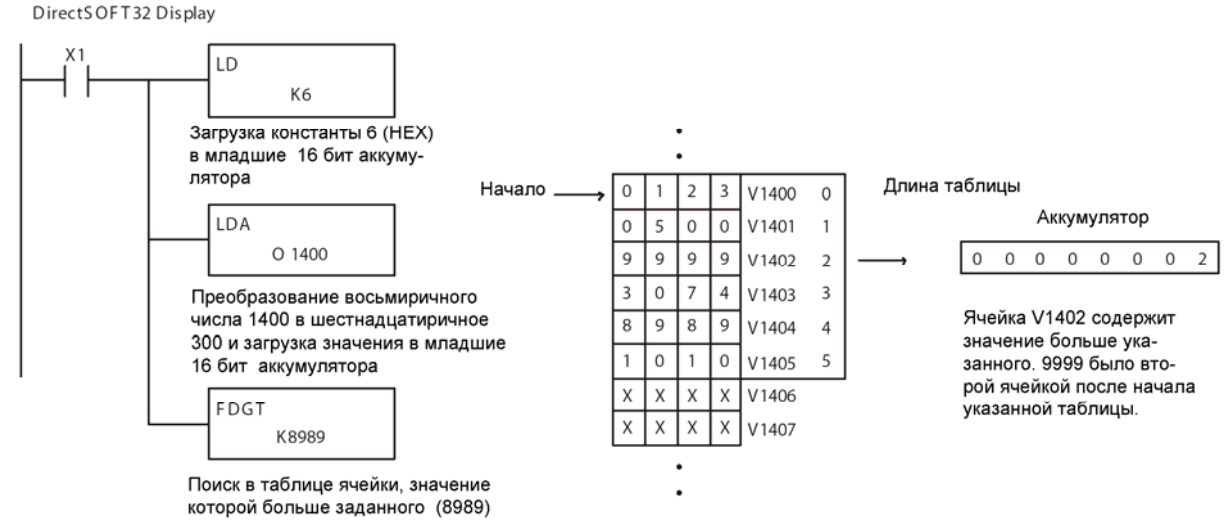
ПРИМЕЧАНИЕ. Эта команда не имеет смещения, типа, требуемого для команды Find.

Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти
Константа	K	0-FFFF

Флаги	Описание
SP53	«1», когда не найдено значение больше искомого в указанной таблице

ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги. Указатель для этой команды может начинаться с 0 и сохраняется в аккумуляторе

В следующем примере, когда X1 включен, константа K6 загружается в аккумулятор, используя команду Load. Это значение определяет длину таблицы и помещается в первый уровень стека, при выполнении следующей команд Load Address. Восьмеричный адрес 1400 (V1400) - начальная ячейка таблицы загружается в аккумулятор. Ячейка, значение которой будет больше указанного определяется в команде Find Greater Than. Если найдено значение больше искомого, смещение (от начальной ячейки таблицы), где значение размещено, сохраняется в аккумуляторе. Если значение больше искомого не найдено, в аккумулятор помещается 0 и устанавливается SP53 .



Набор на ручном программаторе

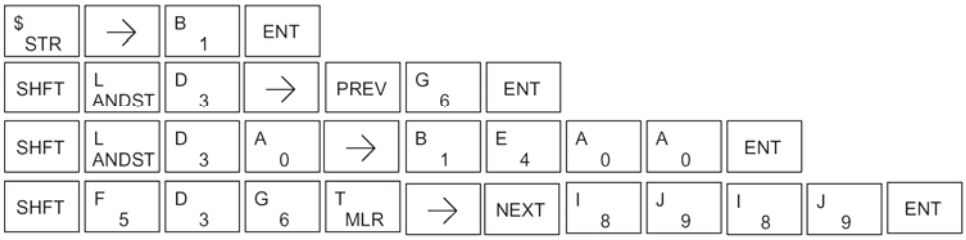
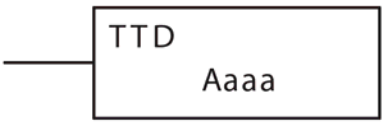


Table to Destination (TTD)



Команда Table To Destination перемещает значение из таблицы V-памяти в ячеки V-памяти и увеличивает указатель таблицы на 1. Первая ячейка V-памяти в таблице содержит указатель таблицы, который указывает следующую ячейку в таблице, которая будет перемещена. Инструкция будет выполнена, только если в данном цикле сканирование вход включен. Указатель таблицы сбросится в 1, когда значение равно последней ячейке в таблице. Функциональные параметры загружаются в первый уровень стека и аккумулятор двумя дополнительными инструкциями. Ниже перечислены шаги, необходимые для использования команды Table To Destination

Шаг 1:— Загрузите длину таблицы (число ячеек V-памяти) в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным числом в пределах от 0 до FF.

Шаг 2:— Загрузите начальный адрес ячейки V-памяти для таблицы в аккумулятор(помните, что начальная ячейка используется как указатель таблицы). Этот параметр должен быть шестнадцатиричным числом.

Шаг 3:— Вставьте команду TTD, которая определяет адрес ячейки V-памяти (Vaaa).

Полезный совет: — Вы можете использовать команду LDA для преобразования восьмиричного адреса в шестнадцатиричный и загрузки его в аккумулятор.

Полезный совет: — Инструкция будет выполнятся каждый цикл сканирования, если входная логика включена. Если Вы не хотите, чтобы инструкция выполнилась для более чем одно сканирование, используйте команду одиночного импульса PD во входной логике.

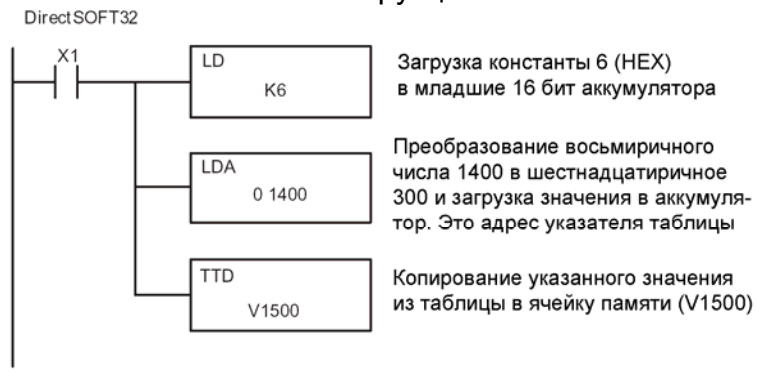
Полезный совет: — Ячейка указателя должна быть установлена в значение начальной ячейки для работы с таблицей. Необходимо использовать специальное реле SP0 или команду одиночного импульса PD, в этом случае значение будет установлено только в одном цикле сканирования и не будет влиять на работу команды.

Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти

Флаги	Описание
SP56	«1», когда значение указателя таблицы равно длине таблицы.

ПРИМЕЧАНИЕ. Флаги состояния (SP) не только допустимы до исполнения другой команды, которая использует, те же самые флаги, или конца цикла сканирования, но, и указатель для этих команд сначала равен 0 и сбрасывается, когда достигнута длина таблицы. На первый взгляд может показаться, что указатель должен сброситься в 0. Однако, он сбрасывается в 1.

В следующем примере, когда X1 включен, константа (K6) загружается в аккумулятор, используя команду Load. Это значение определяет длину таблицы и помещается в первый уровень стека после того, как команда Load Address будет выполнена. Восьмеричный адрес 1400 (V1400) - начальная ячейка для исходной таблицы – загружается в аккумулятор. Помните, V1400 используется как ячейка указателя, и, фактически, не часть таблицы как источника данных. Адрес ячейки (V1500) определен в команде Table to Destination. Указатель таблицы (в этом случае V1400) будет увеличен на “1” после каждого выполнения TTD инструкции.



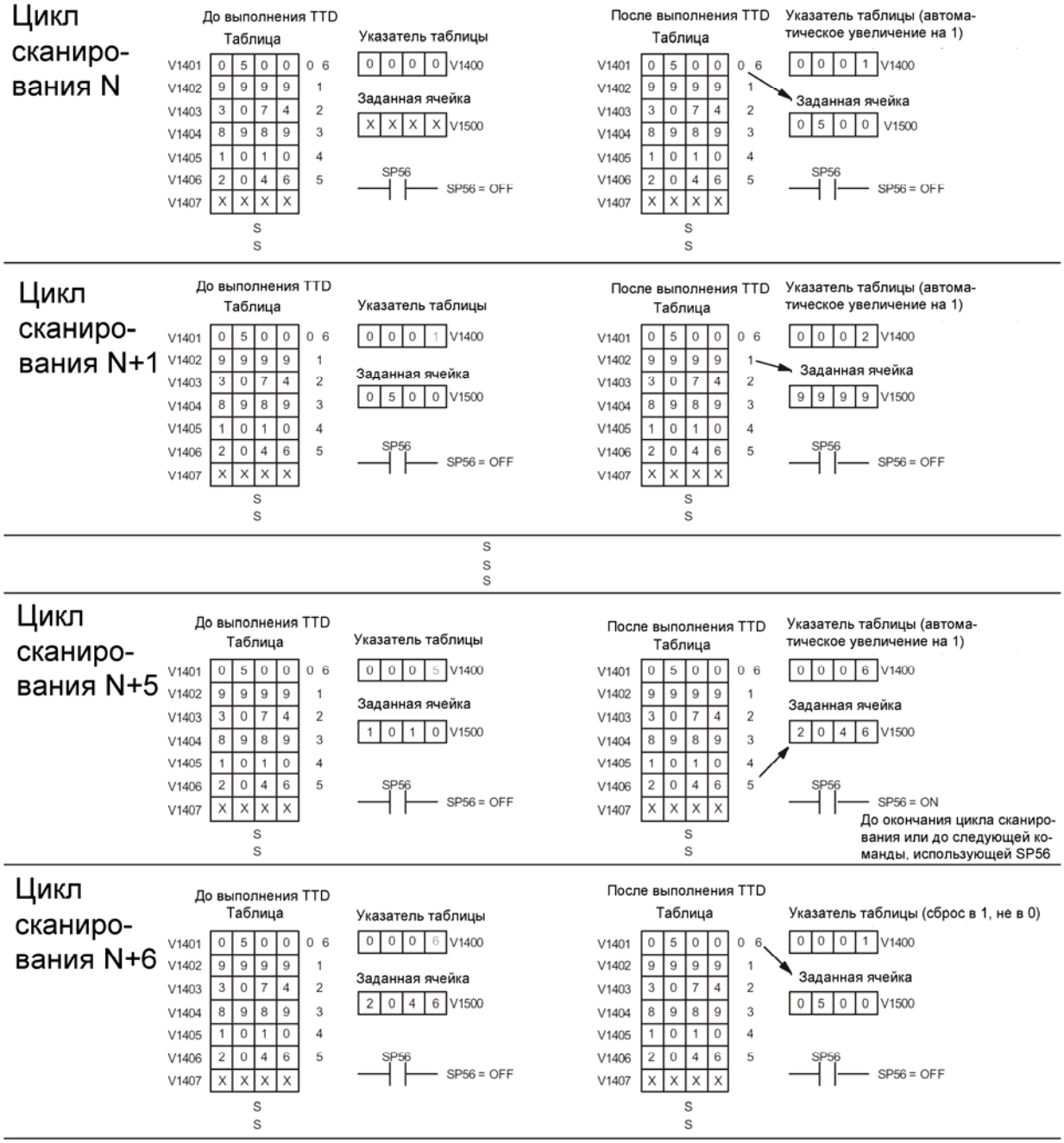
Набор на ручном программаторе

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	PREV	G 6	ENT			
SHFT	L ANDST	D 3	A 0	→	B 1	E 4	A 0	A 0	ENT
SHFT	T MLR	T MLR	D 3	→	B 1	F 5	A 0	A 0	ENT

Важно понять, как пронумерованы ячейки таблицы. Если Вы рассмотрите таблицу в примере, вы увидите, 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, и затем сбрасывается в единицу.

Наш пример, также использует нормально-открытый входной контакт (X1), чтобы управлять выполнением команды. Так как цикл сканирования процессора совершается за минимальное время, то и автоматическое увеличение указателя таблицы в цикле произойдет очень быстро. Если такое положение вещей является для Вас проблемой, то воспользуйтесь специальным реле SP56 или командой одиночного импульса (PD) с защелкой (например C1), чтобы таблица циклически прошла все ячейки и затем останавилась. Логика, показанная здесь не обязательна, и дана как пример.

На следующем рисунке показаны результаты выполнения нашего примера программы в каждом цикле. Обратите внимание, как указатель автоматически изменяется в цикле от 0 до 6, и затем, сбрасывается в 1 вместо 0. Также, обратите внимание, что SP56 включается только при окончании цикла.



Remove from Bottom (RFB)

Команда Remove From Bottom перемещает значение из таблицы V-памяти в ячеки V-памяти и уменьшает указатель таблицы на 1. Первая ячейка V-памяти в таблице содержит указатель таблицы, который указывает следующую ячейку в таблице, которая будет перемещена. Инструкция будет выполнена, только если в данном цикле сканирование вход включен. Команда окончит свое выполнение в тот момент, когда указатель таблицы станет равным 0. Функциональные параметры загружаются в первый уровень стека и аккумулятор двумя дополнительными командами. Ниже перечислены шаги, необходимые для использования команды Remove From Bottom

Шаг 1:— Загрузите длину таблицы (число ячеек V-памяти) в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным числом в пределах от 0 до FF.

Шаг 2:— Загрузите начальный адрес ячейки V-памяти для таблицы в аккумулятор (помните, что начальная ячейка используется как указатель таблицы). Этот параметр должен быть шестнадцатиричным числом.

Шаг 3:— Вставьте команду RFB, которая определяет адрес ячейки V-памяти (Vaaa).

Полезный совет: — Вы можете использовать команду LDA для преобразования восьмиричного адреса в шестнадцатиричный и загрузки его в аккумулятор.

Полезный совет: — Инструкция будет выполнятся каждый цикл сканирования, если входная логика включена. Если Вы не хотите, чтобы команда выполнялась более чем один цикл сканирования, используйте команду одиночного импульса PD во входной логике.

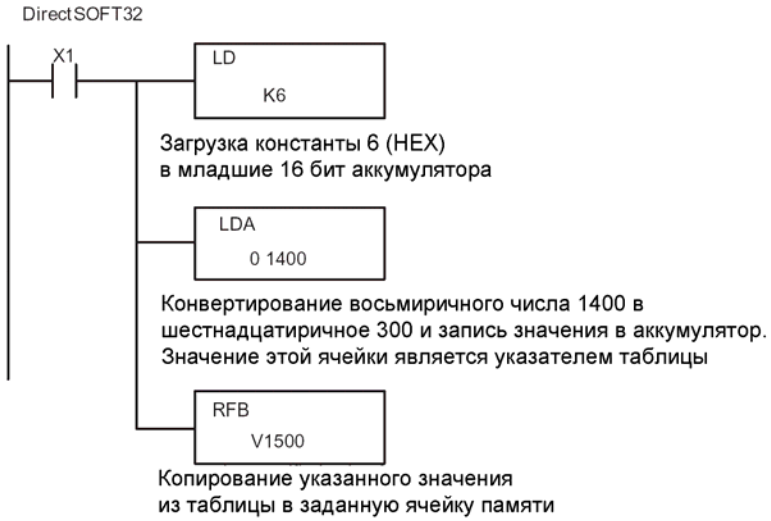
Полезный совет: — Ячейка указателя должна быть установлена в значение начальной ячейки для работы с таблицей. Необходимо использовать специальное реле SP0 или команду одиночного импульса PD, в этом случае значение будет установлено только в одном цикле сканирования и не будет влиять на работу команды.

Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти

Флаги	Описание
SP56	«1», когда указатель таблицы равен 0.

ПРИМЕЧАНИЕ. Флаги состояния (SP) не только допустимы до исполнения другой команды, которая использует, те же самые флаги, или конца цикла сканирования, но, и указатель для данной команды должен быть установлен до начала выполнения. Указатель 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, используя команду Load. Это значение определяет длину таблицы и помещается в первый уровень стека после того, как команда Load Address будет выполнена. Восьмеричный адрес 1400 (V1400) - начальная ячейка для исходной таблицы – загружается в аккумулятор. Помните, V1400 используется как ячейка указателя, и, фактически, не часть таблицы как источника данных. Адрес ячейки (V1500) определен в команде Remove From Bottom. Указатель таблицы (в этом случае V1400) будет уменьшен на “1” после каждого выполнения RFB инструкции.



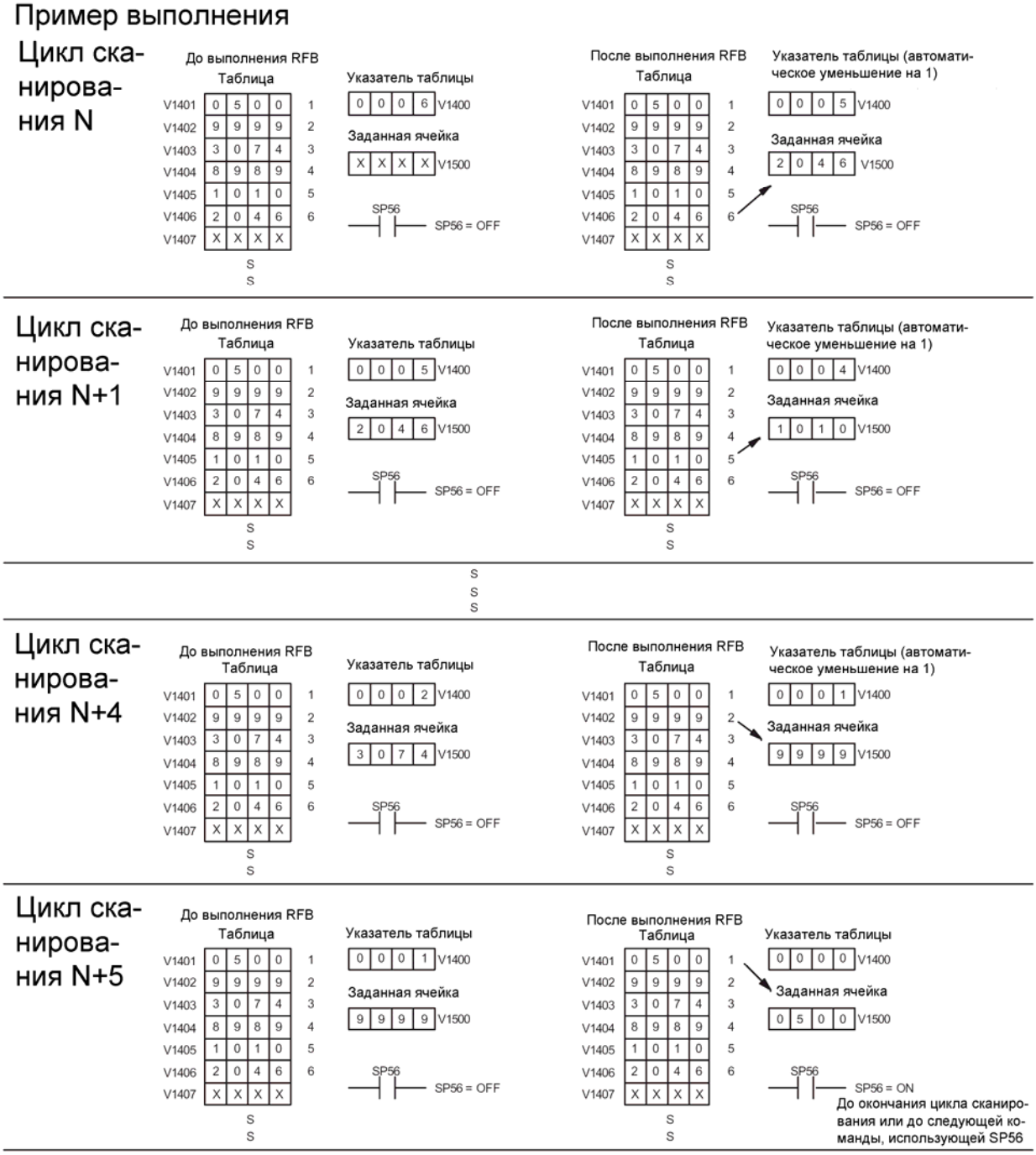
Набор на ручном программаторе

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	PREV	G 6	ENT			
SHFT	L ANDST	D 3	A 0	→	B 1	E 4	A 0	A 0	ENT
SHFT	R ORN	F 5	B 1	→	B 1	F 5	A 0	A 0	ENT

Важно понять, как пронумерованы ячейки таблицы. Если Вы рассмотрите таблицу в примере, вы увидите, что первая ячейка данных, V1401, будет использоваться, когда указатель равен единице. Вторая ячейка данных, V1402, будет использоваться, когда указатель равен двойке и т.д.

Наш пример, также использует нормально-открытый входной контакт (X1), чтобы управлять выполнением команды. Так как цикл сканирования процессора совершается за минимальное время, то и автоматическое увеличение указателя таблицы в цикле произойдет очень быстро. Если такое положение вещей является для Вас проблемой, то воспользуйтесь командой одиночного импульса (PD), чтобы удалить одно значение, каждый раз, когда входной в контакт переходит из состояния выключено в состояние включено.

На следующем рисунке показаны результаты выполнения нашего примера программы в каждом цикле. Обратите внимание, как указатель автоматически изменяется в цикле от 6 до 0. Также, обратите внимание, что SP56 включается только при окончании цикла.



Source to Table (STT)

Команда Source To Table перемещает значения из ячеек V-памяти в таблицу V-памяти и увеличивает указатель таблицы на 1. Когда указатель таблицы достигает конца таблицы, то он сбрасывается в 1. Первая ячейка V-памяти в таблице содержит указатель таблицы, указывающий на следующую ячейку в таблице, в которой будет сохранено значение. Инструкция будет выполнена, только если в данном цикле сканирование вход включен. Функциональные параметры загружаются в первый уровень стека и аккумулятор двумя дополнительными командами. Ниже перечислены шаги, необходимые для использования команды Source To Table

Шаг 1:— Загрузите длину таблицы (число ячеек V-памяти) в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным числом в пределах от 0 до FF.

Шаг 2:— Загрузите начальный адрес ячейки V-памяти для таблицы в аккумулятор(помните, что начальная ячейка используется как указатель таблицы). Этот параметр должен быть шестнадцатиричным числом.

Шаг 3:— Вставьте команду STT, которая определяет начальный адрес ячейки V-памяти (Vaaa). Из которой начинается перемещение данных в таблицу.

Полезный совет: — Вы можете использовать команду LDA для преобразования восьмиричного адреса в шестнадцатиричный и загрузки его в аккумулятор.

Полезный совет: — Инструкция будет выполнятся каждый цикл сканирования, если входная логика включена. Если Вы не хотите, чтобы инструкция выполнилась более чем один цикл сканирования всей таблицы, то используйте команду одиночного импульса PD во входной логике.

Полезный совет: — Значение счетчика таблицы должно быть установлено в значение начальной ячейки для работы с таблицей. При этом, он должен быть установлено в значение, которое находится в пределах длины таблицы. Например, если длина таблицы - 6 слов, то допустимый диапазон значений, которые могли бы быть в указателе между 0 и 6. Если значение находится за границами этого диапазона, данные не будут перемещаться. Также, необходимо использовать команду одиночного импульса PD, в этом случае значение будет установлено только в первом сканировании и будет не влиять на работу команды.

Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти

Флаги	Описание
SP56	«1», когда указатель таблицы равен длине таблицы.

ПРИМЕЧАНИЕ. Флаги состояния (SP) допустимо использовать до исполнения другой команды, которая использует, те же самые флаги, или конца цикла сканирования. Указатель для этих команд сначала равен 0 и автоматически сбрасывается в 1, когда достигнута длина таблицы.

В следующем примере, когда X1 включен, константа (K6) загружается в аккумулятор, используя команду Load. Это значение определяет длину таблицы и помещается в первый уровень стека после того, как команда Load Address будет выполнена. Восьмеричный адрес 1400 (V1400) - начальная ячейка для исходной таблицы – загружается в аккумулятор. Помните, V1400 используется как ячейка указателя, и, фактически, не часть таблицы как источника данных. Адрес ячейки (V1500) определен в Source To Table. Указатель таблицы будет увеличен на “1” после каждого выполнения команды.

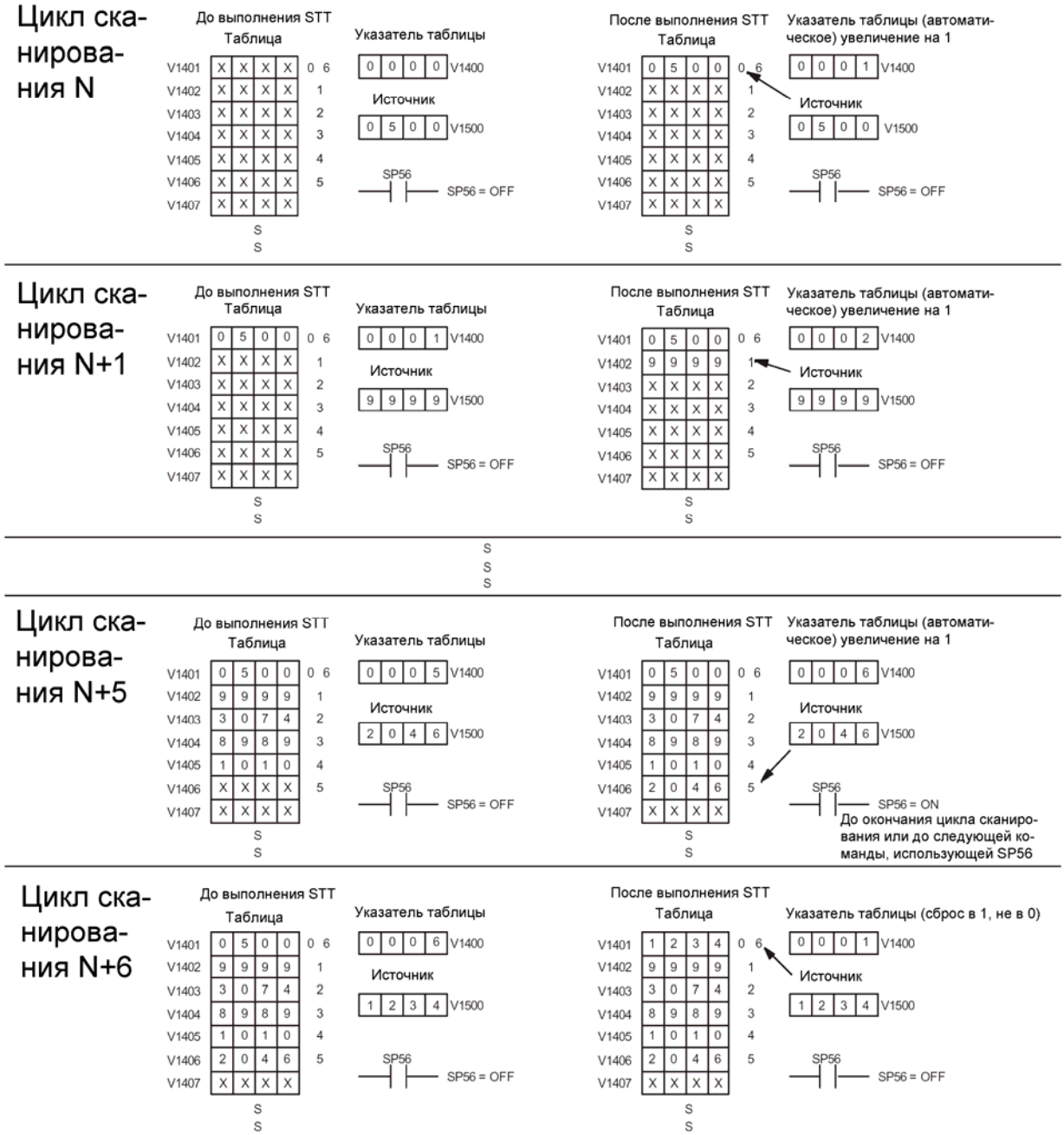
Набор на ручном программаторе

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	PREV	G 6	ENT			
SHFT	L ANDST	D 3	A 0	→	B 1	E 4	A 0	A 0	ENT
SHFT	S RST	SHFT	T MLR	T MLR	→	B 1	F 5	A 0	A 0	ENT

Важно понять, как пронумерованы ячейки таблицы. Если Вы рассмотрите таблицу в примере, вы увидите, 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, и затем сбрасывается в единицу.

Наш пример, также использует нормально-открытый входной контакт (X1), чтобы управлять выполнением команды. Так как цикл сканирования процессора совершается за минимальное время, то и автоматическое увеличение указателя таблицы в цикле произойдет очень быстро. Если такое положение вещей является для Вас проблемой, то воспользуйтесь командой одиночного импульса (PD), чтобы переместить только одно значение, каждый раз, когда входной в контакт переходит из состояния выключено в состояние включено.

На следующем рисунке показаны результаты выполнения нашего примера программы в каждом цикле. Обратите внимание, как указатель автоматически изменяется в цикле от 0 до 6, и затем, сбрасывается в 1 вместо 0. Также, обратите внимание, что SP56 включается только при окончании цикла. Хотя наш пример не показывает этого, но мы предполагаем, что имеется другая часть программы, которая изменяет значение в V1500 (источник данных) до выполнения STT инструкции. Это выполнять не обязательно, но при этом легче показать процедуру копирования источника данных в таблицу.



Remove from Table (RFT)

Команда Remove From Table удаляет значение из таблицы V-памяти и сохраняет его в ячейке V-памяти. Когда значение удалено из таблицы, то все остальные значения перемещаются вверх на 1 ячейку. Первая ячейка V-памяти в таблице содержит счетчик длины таблицы. Каждый раз при выполнении команды этот счетчик уменьшается на 1. Если счетчик длины равен нулю или больше максимальной длины таблицы (определенной в первом уровне стека) команда не выполняется, и SP56 будет включен.

Команда выполняется, только, если в данном цикле сканирования вход включен. Функциональные параметры загружаются в первый уровень стека и аккумулятор двумя дополнительными командами. Ниже перечислены шаги, необходимые для использования команды Remove From Table.

Шаг 1:— Загрузите длину таблицы (число ячеек V-памяти) в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным числом в пределах от 0 до FF.

Шаг 2:— Загрузите начальный адрес ячейки V-памяти для таблицы в аккумулятор (помните, что начальная ячейка используется как указатель таблицы). Этот параметр должен быть шестнадцатиричным числом.

Шаг 3:— Вставьте команду RFT, которая определяет адрес ячейки V-памяти (Vaaa). Эта ячейка памяти куда будет удалено значение.

Полезный совет: — Вы можете использовать команду LDA для преобразования восьмиричного адреса в шестнадцатиричный и загрузки его в аккумулятор.

Полезный совет: — Инструкция будет выполнятся каждый цикл сканирования, если входная логика включена. Если Вы не хотите, чтобы команда выполнялась более чем один цикл сканирования, используйте команду одиночного импульса PD во входной логике.

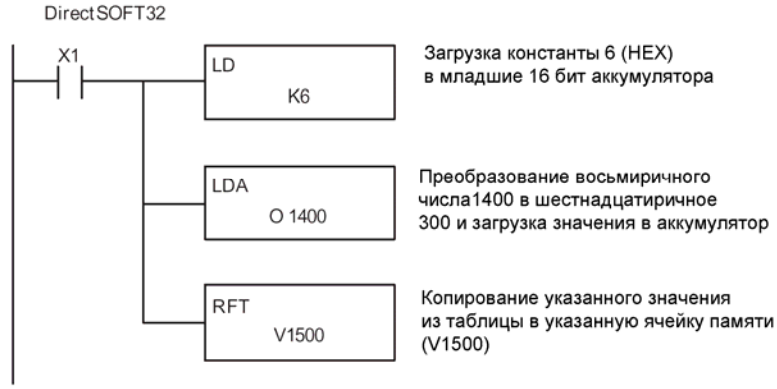
Полезный совет: — Значение счетчика таблицы должно быть установлено в значение начальной ячейки для работы с таблицей. При этом, он должен быть установлено в значение, которое находится в пределах длины таблицы. Например, если длина таблицы - 6 слов, то допустимый диапазон значений, которые могли бы быть в указателе между 1 и 6. Если значение находится за границами этого диапазона, данные не будут перемещаться. Также, необходимо использовать команду одиночного импульса PD, в этом случае значение будет установлено только в первом сканировании и будет не влиять на работу команды.

Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти

Флаги	Описание
SP56	«1», когда счетчик таблицы равен 0.

ПРИМЕЧАНИЕ. Флаги состояния (SP) не только допустимы до исполнения другой команды, которая использует, те же самые флаги, или конца цикла сканирования, но, и указатель для данной команды должен быть установлен до начала выполнения. Указатель 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, используя команду Load. Это значение определяет длину таблицы и помещается в первый уровень стека после того, как команда Load Address будет выполнена. Восьмеричный адрес 1400 (V1400) - начальная ячейка для исходной таблицы – загружается в аккумулятор. Помните, V1400 используется как ячейка указателя, и, фактически, не часть таблицы как источника данных. Адрес ячейки (V1500) определен в команде Remove From Table. Указатель таблицы (в этом случае V1400) будет уменьшен на “1” после каждого выполнения команды.



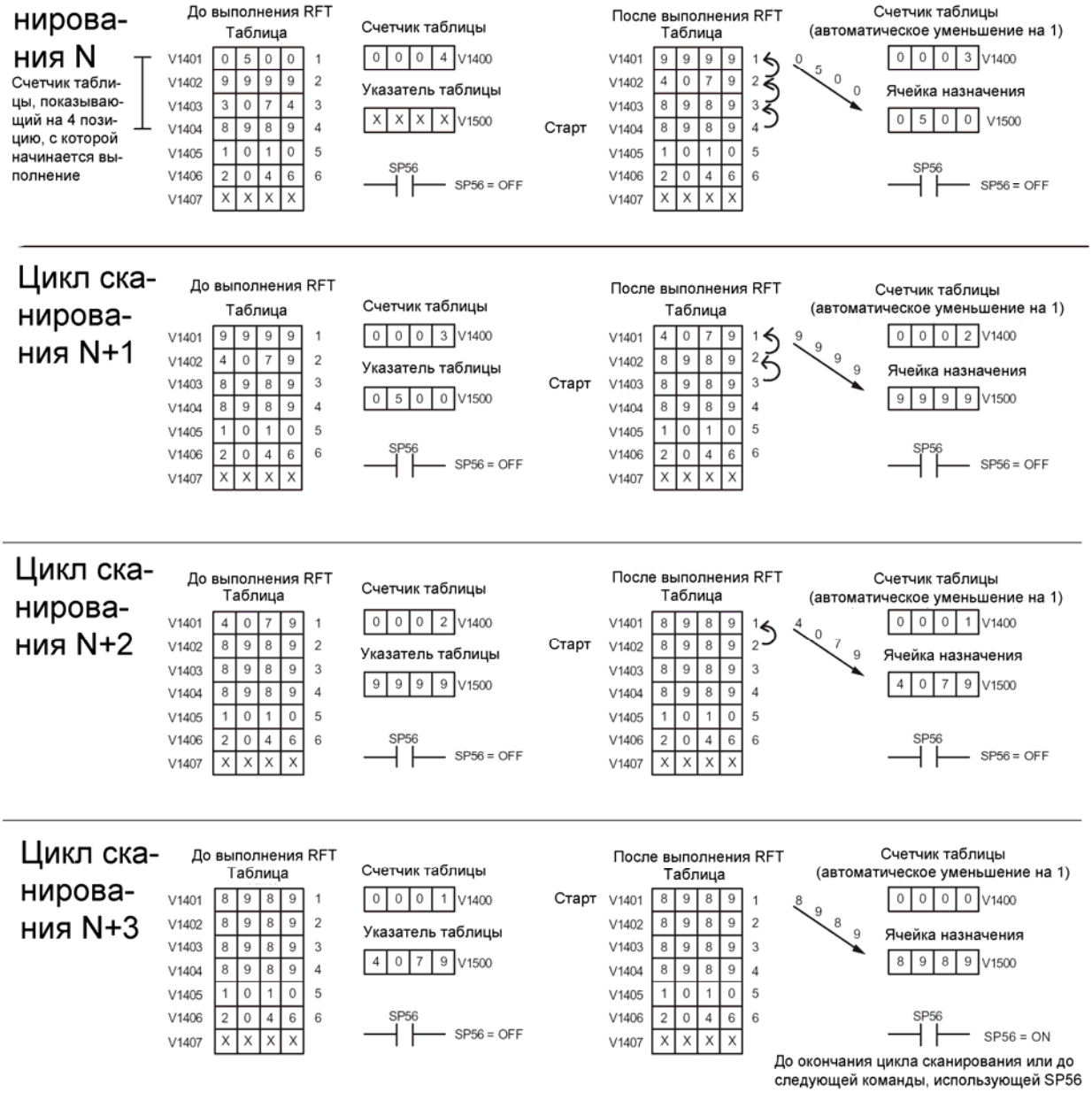
Набор на ручном программаторе

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→	PREV	G 6	ENT		
SHFT	L ANDST	D 3	A 0	→	B 1	E 4	A 0	A 0	ENT
SHFT	R ORN	F 5	T MLR	→	B 1	F 5	A 0	A 0	ENT

Так как счетчик таблиц определяет диапазон данных, которые будут удалены из таблицы, важно понять, как пронумерованы ячейки таблицы. Если Вы рассмотрите таблицу в примере, вы увидите, что ячейки пронумерованы сверху вниз. Например, если счетчик таблицы, равен 6, то все шесть из ячеек будут удалены при выполнения команды.

Наш пример, также использует нормально-открытый входной контакт (X1), чтобы управлять выполнением команды. Так как цикл сканирования процессора совершается за минимальное время, то и автоматическое уменьшение указателя таблицы в цикле произойдет очень быстро. Если такое положение вещей является для Вас проблемой, то воспользуйтесь командой одиночного импульса (PD), чтобы удалить одно значение, каждый раз, когда входной в контакт переходит из состояния выключено в состояние включено.

На следующем рисунке показаны результаты выполнения нашего примера программы в каждом цикле. В нашем примере мы показываем счетчик таблицы первоначально установленный в 4. (Не забудьте, Вы можете устанавливать счетчик таблиц в любое значение в пределах длины таблицы.) Счетчик таблицы автоматически уменьшается от 4 до 0, при выполнении команды. Обратите внимание, как последние две ячейки таблицы, 5 и 6, не удаляются из таблицы. Также, обратите внимание, что SP56 включается только при счетчике таблице равном нулю, в конце сканирования таблицы.



Add to Top (ATT)

Команда Add To Top сохраняет значение из ячейки V-памяти в таблицу V-памяти. Когда значение добавляется в таблицу, то все остальные значения перемещаются вниз на 1 ячейку.

Команда выполняется, только, если в данном цикле сканирования вход включен. Функциональные параметры загружаются в первый уровень стека и аккумулятор двумя дополнительными командами. Ниже перечислены шаги, необходимые для использования команды Add To Top.

Шаг 1:— Загрузите длину таблицы (число ячеек V-памяти) в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным числом в пределах от 0 до FF.

Шаг 2:— Загрузите начальный адрес ячейки V-памяти для таблицы в аккумулятор (помните, что начальная ячейка используется как указатель таблицы). Этот параметр должен быть шестнадцатиричным числом.

Шаг 3:— Вставьте команду ATT, которая определяет адрес ячейки V-памяти (Vaaa). Эта ячейка памяти откуда будет вставлено значение.

Полезный совет: — Инструкция будет выполнятся каждый цикл сканирования, если входная логика включена. Если Вы не хотите, чтобы команда выполнялась более чем один цикл сканирования, используйте команду одиночного импульса PD во входной логике.

Полезный совет: — Вы можете использовать команду LDA для преобразования восьмиричного адреса в шестнадцатиричный и загрузки его в аккумулятор.

Полезный совет: — Значение счетчика таблицы должно быть установлено в значение начальной ячейки для работы с таблицей. При этом, он должен быть установлен в значение, которое находится в пределах длины таблицы. Например, если длина таблицы - 6 слов, то допустимый диапазон значений, которые могли бы быть в указателе между 1 и 6. Если значение находится за границами этого диапазона или равно 0, то данные не будут перемещаться в таблицу. Также, необходимо использовать команду одиночного импульса PD, в этом случае значение будет установлено только в первом сканировании и не будет влиять на работу команды.

Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти

Флаги	Описание
SP56	«1», когда счетчик таблицы равен длине таблице.

ПРИМЕЧАНИЕ. Флаги состояния (SP) не только допустимы до исполнения другой команды, которая использует, те же самые флаги, или конца цикла сканирования, но, и указатель для данной команды должен быть установлен до начала выполнения. Указатель 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, используя команду Load. Это значение определяет длину таблицы и помещается в первый уровень стека после того, как команда Load Address будет выполнена. Восьмеричный адрес 1400 (V1400) - начальная ячейка для исходной таблицы и счетчик таблицы – загружается в аккумулятор. Адрес ячейки (V1500) определен в команде Add To Top. Счетчик таблицы будет увеличен на “1” после каждого выполнения команды.

Набор на ручном программаторе

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	PREV	G 6	ENT			
SHFT	L ANDST	D 3	A 0	→	B 1	E 4	A 0	A 0	ENT
SHFT	A 0	T MLR	T MLR	→	B 1	F 5	A 0	A 0	ENT

Для команды ATT, счетчик таблиц определяет число добавлений, которые могут быть сделаны прежде, чем команда остановит свое выполнение. Поэтому, полезно понять, как система использует этот счетчик, чтобы управлять выполнением команды.

Например, если счетчик таблиц был установлен в 2, и длина таблицы была бы 6 слов, то возможно произвести только 4 добавления данных прежде, чем выполнение будет остановлено. Это можно вычислить по формуле:

Длина таблицы - Счетчик таблицы = Число выполнений

Наш пример, также использует нормально-открытый входной контакт (X1), чтобы управлять выполнением команды. Так как цикл сканирования процессора совершается за минимальное время, то и автоматическое увеличение счетчика таблицы в цикле произойдет очень быстро. Если такое положение вещей является для Вас проблемой, то воспользуйтесь командой одиночного импульса (PD), чтобы удалить одно значение, каждый раз, когда входной в контакт переходит из состояния выключено в состояние включено.

На следующем рисунке показаны результаты выполнения нашего примера программы в каждом цикле. Счетчик таблицы первоначально установленный в 2, автоматически увеличивается от 2 до 6 при выполнении команды. Обратите внимание, что SP56 включается, когда счетчик таблицы равен 6, т.е. равен длине таблице. Хотя наш пример не показывает этого, но мы предполагаем, что имеется другая часть программы, которая изменяет значение в V1500 (источник данных) до выполнения команды ATT.

Пример выполнения

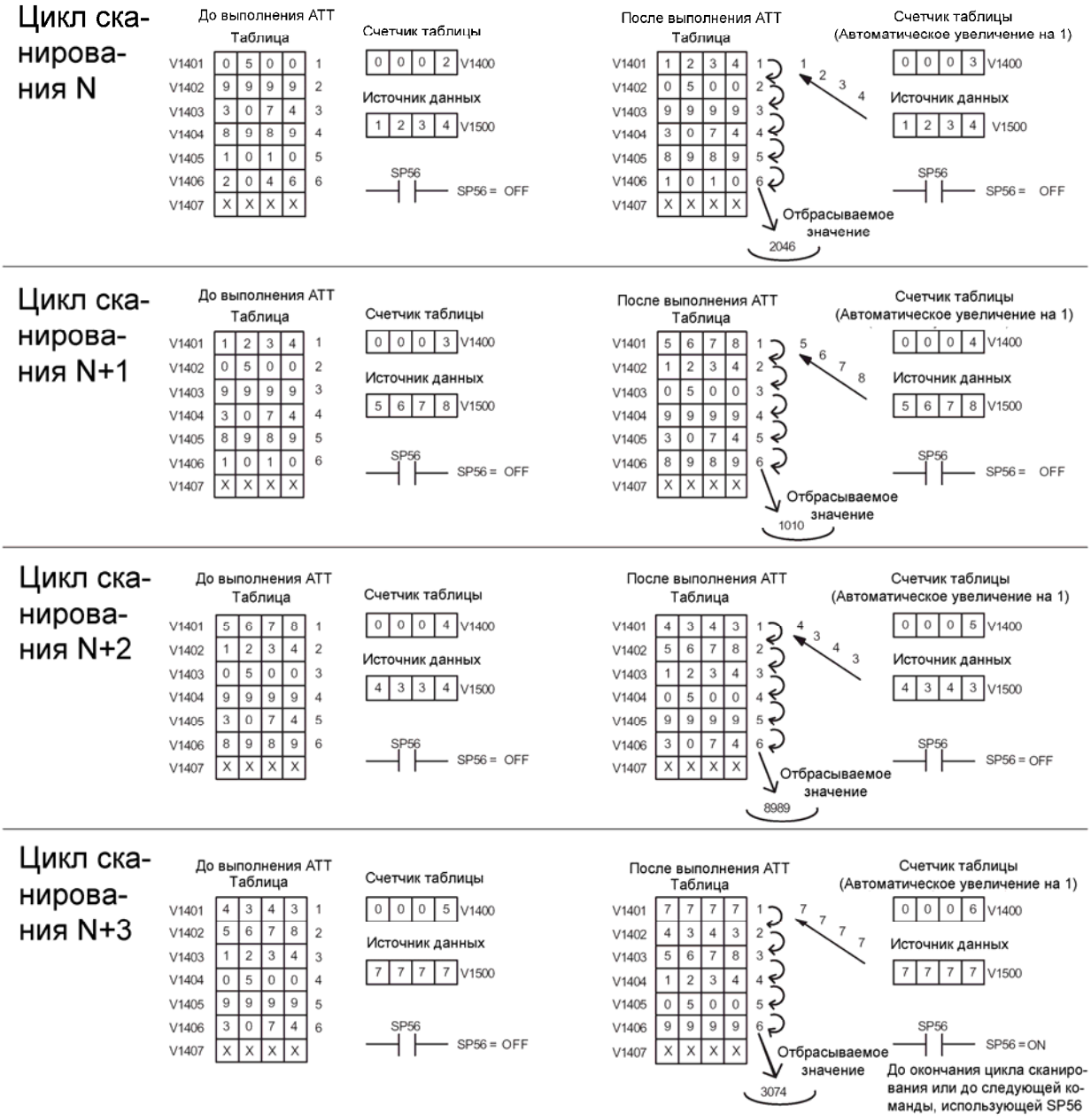
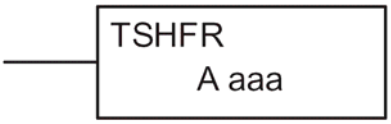


Table Shift Left (TSHFL)

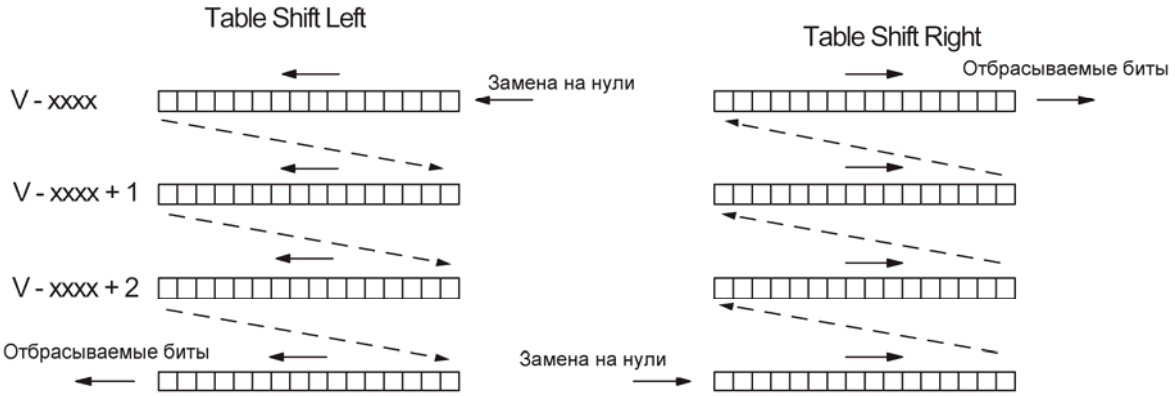
Команда Table Shift Left сдвигает все биты в таблице V-памяти влево, на определенное число разрядов.

Table Shift Right (TSHFR)

Команда Table Shift Right сдвигает все биты в таблице V-памяти вправо, на определенное число разрядов.



Следующее описание применяется к командам, и Table Shift Left, и Table Shift Right. Таблица – это только диапазон ячеек V-памяти. Команды Table Shift Left и Table Shift Right сдвигают биты последовательно по всей таблице. Биты перемещаются из конец одного слова и в противоположный конец следующего слова. В конце таблицы, биты или отбрасываются, или устанавливаются в нули. Пример таблицы приведенной ниже имеет длину четыре слова.



Шаг 1:— Загрузите длину таблицы (число ячеек V-памяти) в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным числом в пределах от 0 до FF.

Шаг 2:— Загрузите начальный адрес ячейки V-памяти для таблицы в аккумулятор. Этот параметр должен быть шестнадцатиричным числом. Вы можете использовать команду LDA, чтобы преобразовать восьмеричный адрес в шестнадцатеричный.

Шаг 3:— Вставьте команду Table Shift Left или Table shift Right, которая определяет число бит на которое производится сдвиг. Число бит должно быть в восьмеричным числом.

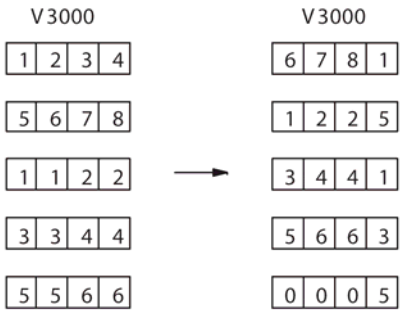
Полезный совет: — Не забудьте, что каждая ячейка V-памяти содержит 16 бит. Так, биты первого слова таблицы пронумерованы от 0 до 17 в восьмеричном виде. Если Вы хотите сдвинуть всю таблицу на 20 бит, то используйте восьмеричное число 24. SP53 будет установлен, если число бит, которые будут перемещены большее чем общее число бит, содержащееся в таблице. Флаг 67 будет установлен, если последний перемещенный бит (перед отброшенным) равен “1”.

Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти

Флаги	Описание
SP53	«1», когда число сдвигаемых бит больше общего числа бит таблицы.
SP67	«1», когда последний перемещенный бит (перед отброшенным) равен “1”.

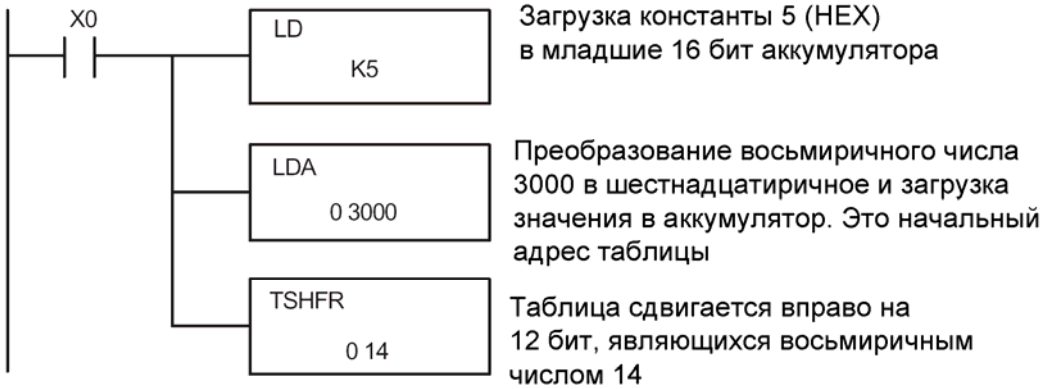
ПРИМЕЧАНИЕ. Флаги состояния действительны только до того момента, когда будет выполнена другая команда, использующая те же самые флаги.

Пример таблицы справа содержит BCD данные. Предположим, что мы хотим выполнить сдвиг таблицы вправо на 3 BCD-цифры (12 бит). При преобразовании к восьмеричному 12 бит равно восьмеричному числу 14. При использовании команды Table Shift Right с восьмеричным параметром 14, мы имеем возникающую в результате таблицу, показанную правее. Обратите внимание, что последовательность цифр 2-3-4 была отброшена, и последовательность цифр 0-0-0 была записана в нижней часте таблице.

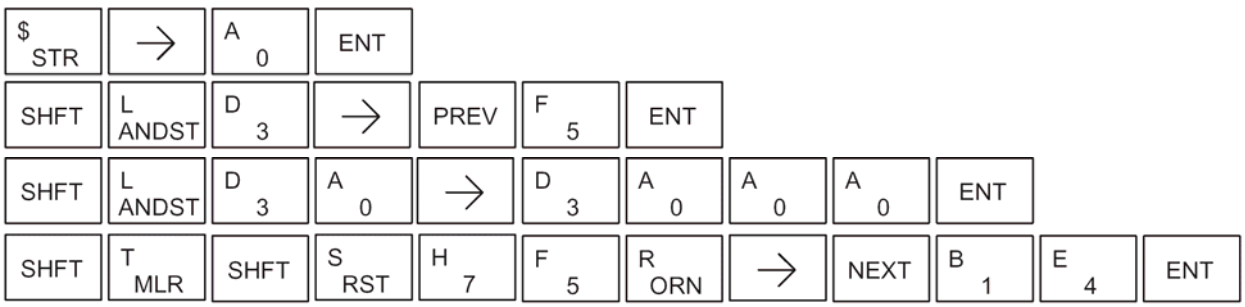


Следующий пример релейной программы предполагает, что данные в ячейках от V3000 до V3004 уже существуют как показано выше. Мы используем вход X0, чтобы начать работу команды Table Shift Right. Сначала, мы загрузим длину таблицы (5 слов) в стек аккумулятора. Затем, мы загружаем в аккумулятор начальный адрес. Поскольку V3000 - восьмиричное число, то мы должны преобразовать его в шестнадцатеричное, используя команду LDA. В заключение, мы используем команду Table Shift Right и определяем число бит, на которое будут смещены данные (десятичное число 12), т.е. восьмеричное число 14.

DirectSOFT 32



Набор на ручном программаторе

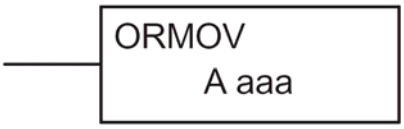


AND Move (ANDMOV)

Команда AND Move копирует данные из таблицы в определенную ячейку памяти, осуществляя операцию И каждого слова с данными, записанными в аккумуляторе.

OR Move (ORMOV)

Команда Or Move копирует данные из таблицы в определенную ячейку памяти, осуществляя операцию ИЛИ каждого слова с данными, записанными в аккумуляторе.



Exclusive OR Move (XORMOV)

Команда Or Move копирует данные из таблицы в определенную ячейку памяти, осуществляя операцию исключающего ИЛИ каждого слова с данными, записанными в аккумуляторе.

Следующее описание применяется к командам AND Move, OR Move, и Exclusive OR Move. Таблица – это только диапазон ячеек V-памяти. Эти команды копируют данные таблицы в другую указанную ячейку, предварительно выполняя логическую операцию над каждым словом с содержимым аккумулятора, записывая новую таблицу.

Шаг 1:— Загрузите длину таблицы (число ячеек V-памяти) в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным числом в пределах от 0 до FF.

Шаг 2:— Загрузите начальный адрес ячейки V-памяти для таблицы в аккумулятор. Этот параметр должен быть шестнадцатиричным числом. Вы можете использовать команду LDA, чтобы преобразовать восьмеричный адрес в шестнадцатеричный.

Шаг 3:— Загрузите BCD/hex комбинацию бит в аккумулятор, которая будет логически комбинироваться с записываемым содержанием таблицы.

Шаг 4:— Вставьте команду AND Move, OR Move, или XOR Move, которая определяет начальную ячейку в которую копируется создаваемая таблица. Эта новая таблица будет автоматически иметь такую же длину, как и первоначальная таблица.

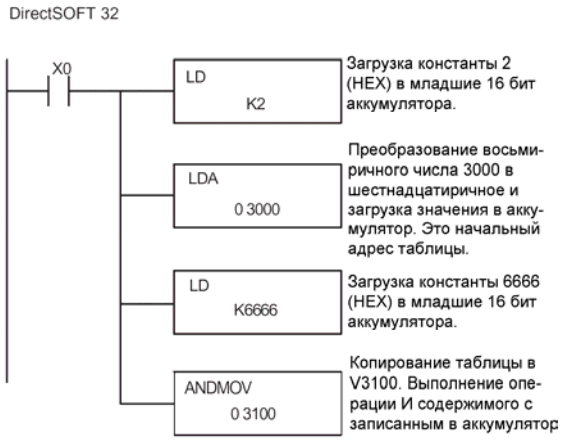
Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти

Пример таблицы справа содержит BCD данные. Предположим, что мы хотим переместить таблицу из двух слов начиная с V3000 и выполнить операцию И с константой K6666. Копия таблицы начиная с V3100 показывает результат операции И для каждого слова.

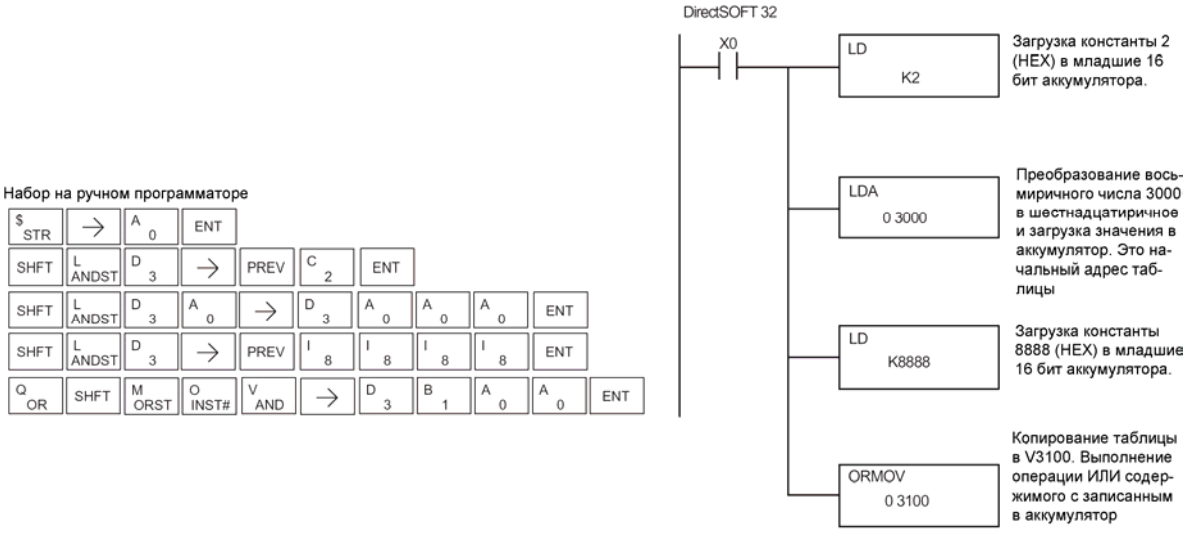
Программа на следующей странице выполняет команду ANDMOV в примере приведенном выше. Предположим, что данные в таблице V3000 - V3001 уже существуют. Сначала мы загружаем длину таблицы (два слова) в аккумулятор. Затем мы загружаем начальный адрес исходной таблицы, используя команду LDA. Затем мы загружаем данные, с которыми должна выполнятся операция И, в аккумулятор. В команде ANDMOV, мы определяем адрес таблицы результата V3100.

Набор на ручном программаторе

$ STR	→	A 0	ENT						
SHFT	L ANDST	D 3	→	PREV	C 2	ENT			
SHFT	L ANDST	D 3	A 0	→	D 3	A 0	A 0	A 0	ENT
SHFT	L ANDST	D 3	→	PREV	G 6	G 6	G 6	G 6	ENT
V AND	SHFT	M ORST	O INST#	V AND	→	D 3	B 1	A 0	A 0	ENT



Пример справа показывает таблицу из двух слов начиная с V3000 и и выполнение операции ИЛИ с константой K8888. Копия таблицы начиная с V3100 показывает результат операции ИЛИ для каждого слова



Набор на ручном программаторе

$ STR	→	A 0	ENT						
SHFT	L ANDST	D 3	→	PREV	C 2	ENT			
SHFT	L ANDST	D 3	A 0	→	D 3	A 0	A 0	A 0	ENT
SHFT	L ANDST	D 3	→	PREV	I 8	I 8	I 8	I 8	ENT
Q OR	SHFT	M ORST	O INST#	V AND	→	D 3	B 1	A 0	A 0	ENT

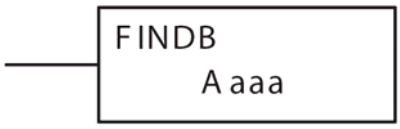
Программа сверху выполняет команду ORMOV в примере приведенном выше. Предположим, что данные в таблице V3000 - V3001 уже существуют. Сначала мы загружаем длину таблицы (два слова) в аккумулятор. Затем мы загружаем начальный адрес исходной таблицы, используя команду LDA. Затем мы загружаем данные, с которыми должна выполнятся операция ИЛИ, в аккумулятор. В команде ORMOV, мы определяем адрес таблицы результата V3100.

Пример справа показывает таблицу из двух слов начиная с V3000 и и выполнение операции исключающего ИЛИ с константой K3333. Копия таблицы начиная с V3100 показывает результат операции исключающего ИЛИ для каждого слова

Пример релейной программы для команды XORMOV подобен данному выше для ORMOV. Только используйте команду XORMOV. На ручном программаторе. Вы должны использовать клавишу SHFT и записать “XORMOV” по буквам.

Find Block (FINDB)

Команда Find Block выполняет поиск местонахождения определенного блока значений в таблице V-памяти. Функциональные параметры загружены в первый и второй уровни стека и аккумулятор тремя дополнительными командами. Если блок найден, начальный адрес будет сохранен в аккумуляторе. Если блок не найден, флаг SP53 будет установлен.



Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти

Флаги	Описание
SP53	«1», когда команда Find Block не нашла блок в указанной таблице V-памяти.

Ниже показаны необходимые шаги для использования функции Find Block.

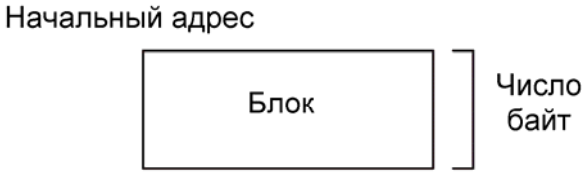
Шаг 1:— Загрузите число байт располагаемых в блоке. Этот параметр должен быть шестнадцатиричным числом в пределах от 0 до FF

Шаг 2:— Загрузите длину таблицы (число ячеек V-памяти) в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным числом в пределах от 0 до FF.

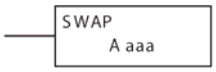
Шаг 3:— Загрузите конечный адрес для всех таблиц в аккумулятор. Этот параметр должен быть шестнадцатиричным числом. Вы можете использовать команду LDA, чтобы преобразовать восьмеричный адрес в шестнадцатеричный.

Шаг 4:— Загрузите начальный адрес для всех таблиц в аккумулятор. Этот параметр должен быть шестнадцатиричным числом. Вы можете использовать команду LDA, чтобы преобразовать восьмеричный адрес в шестнадцатеричный.

Шаг 5:— Вставьте команду Find Block, которая определяет начальную ячейку блока данных, для попытки поиска.



Swap (SWAP)



Команда Swap обменивает данные в двух таблицах равной длины.

Шаг 1:— Загрузите длину таблицы (число ячеек V-памяти) в первый уровень стека аккумулятора. Этот параметр должен быть шестнадцатиричным числом в пределах от 0 до FF. Помните, что таблицы должны быть одинаковой длины.

Шаг 2:— Загрузите начальный адрес ячейки V-памяти для таблицы в аккумулятор. Этот параметр должен быть шестнадцатиричным числом. Вы можете использовать команду LDA для конвертации восьмиричного адреса в шестнадцатиричное число.

Шаг 3:— Вставьте команду Swap, которая определяет начальный адрес второй таблицы V-памяти. Этот параметр должен быть шестнадцатиричным числом. Вы можете использовать команду LDA для конвертации восьмиричного адреса в шестнадцатиричное число.

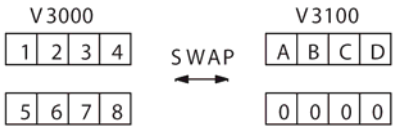
Полезный совет: — Инструкция будет выполнятся каждый цикл сканирования, если входная логика включена. Если Вы не хотите, чтобы команда выполнялась более чем один цикл сканирования, используйте команду одиночного импульса PD во входной логике.

Полезный совет: — Вы можете использовать команду LDA для преобразования восьмиричного адреса в шестнадцатиричный и загрузки его в аккумулятор.

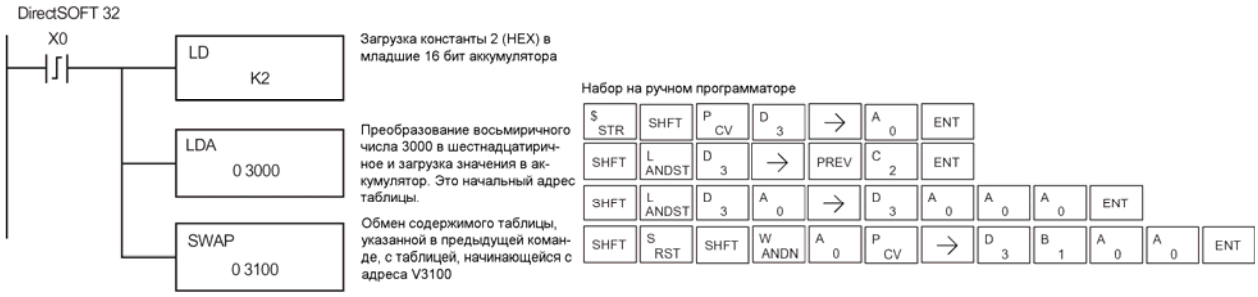
Полезный совет: — Перестановка данных выполняется в пределах одного цикла сканирования. Если инструкция выполняется в течении нескольких последовательных циклов сканирования, то будет трудно знать фактическое содержание любой таблицы в заданный момент времени. Таким образом, не забудьте, что применять эту команду следует только в одном сканировании.

Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти

Пример справа показывает таблицу из двух слов при V3000. Мы будем менять содержание с другой таблицей из двух слов начиная с 3100, используя команду Swap. Требуемая релейная программа дана ниже.



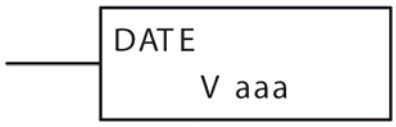
Пример программы, приведенный ниже использует контакт PD (одиночный импульс для перехода входа от выключено в включено). Сначала, мы загружаем длину таблиц (два слова) в аккумулятор. Затем мы загружаем адрес первой таблицы (V3000) в аккумулятор, используя команду LDA, преобразовывающую восьмеричный адрес в шестнадцатеричное число. Обратите внимание, что это не имеет значения, какую таблицу мы объявляем первой, потому что результаты перестановки будут одинаковы.



Команды Часы / Календарь

Date (DATE)

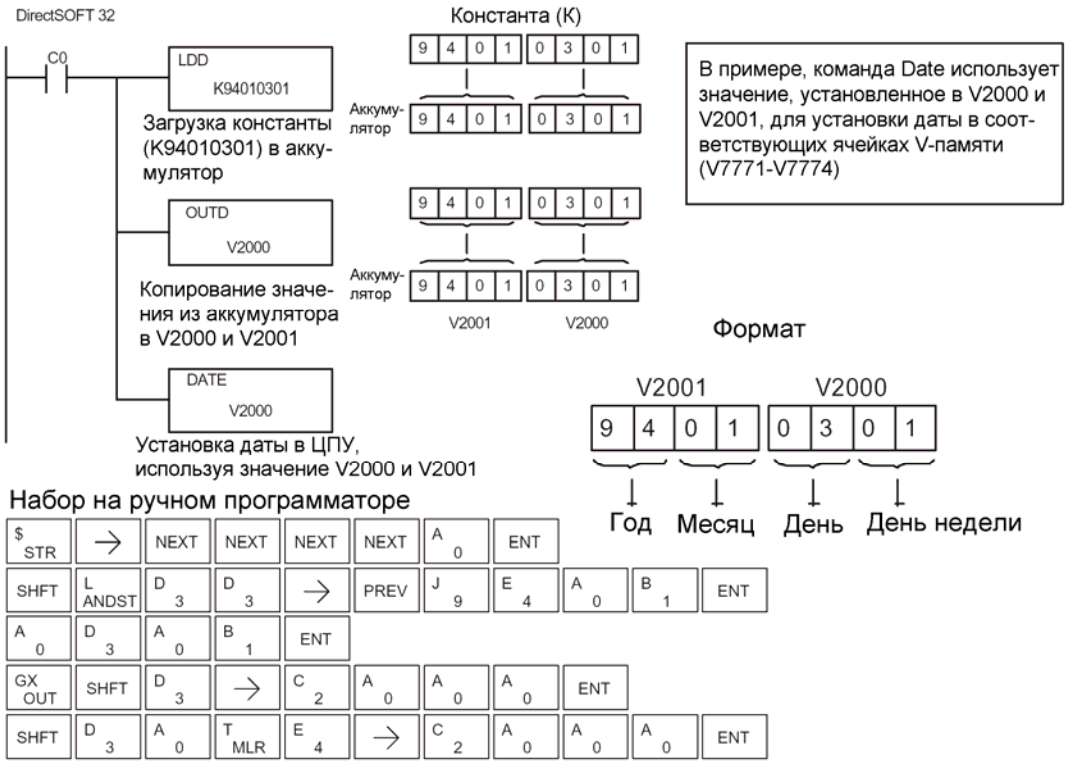
Команда Date может использоваться, чтобы установить дату в процессоре микроконтроллера. Инструкция требует двух последовательных ячеек V-памяти (Vaaa), чтобы установить дату. Если значения в определенных ячейках не допустимы, то дата не будет установлена. Текущая дата может читаться из 4 последовательных ячеек V-памяти (V7771-V7774).



В следующем примере, когда C0 включен, константа (K94010301) загружается в аккумулятор, используя команду Load Double (C0 должен быть контактом связанным с командой одиночного (PD)). Значение в аккумуляторе выводится в V2000, используя команду Out Double. Команда Date использует значение в V2000, чтобы установить дату в процессоре микроконтроллера.

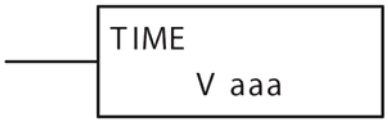
Дата	Диапазон	Ячейка V-памяти (BCD) (Только для чтения)
Год	0-99	V7774
Месяц	1-12	V7773
День	1-31	V7772
День Недели	0-06	V7771
Значения, введенные для дня недели: 0 = воскресенье, 1 = понедельник, 2 = вторник, 3 = среда, 4 = четверг, 5 = пятница, 6 = суббота		

Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти



Time (TIME)

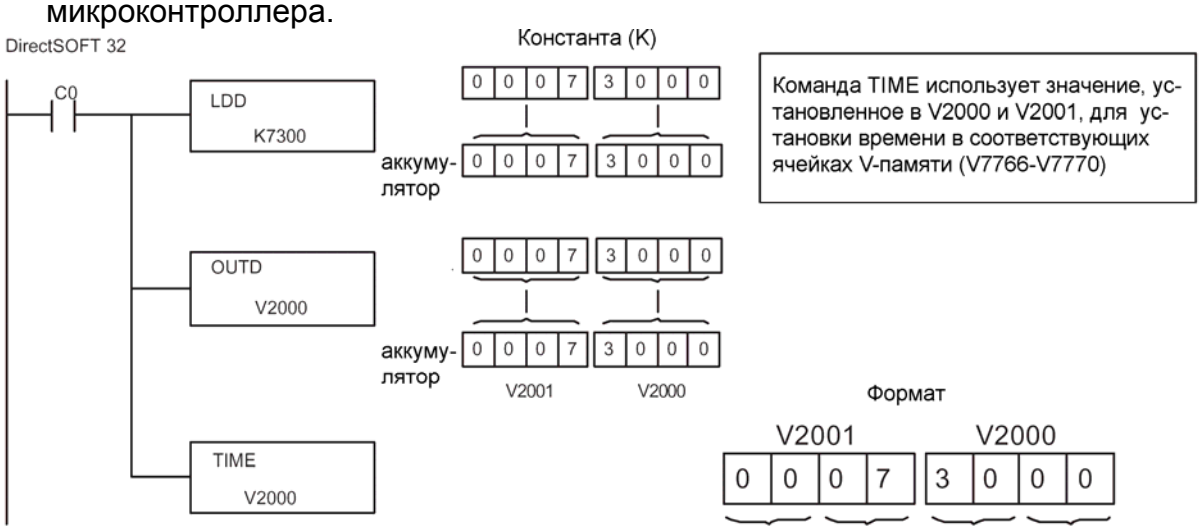
Команда Time может использоваться, для того чтобы установить время (24 часа) в процессоре микроконтроллера. Инструкция требует двух последовательных ячеек V-памяти (Vaaa), чтобы установить время. Если значения в определенных ячейках не допустимы, то время не будет установлено. Текущее время может читаться из 4 последовательных ячеек V-памяти V7747 и V7766-V7770.



Дата	Диапазон	Ячейка V-памяти (BCD) (Только для чтения)
1/100 секунды (10мс)	0-99	V7747
Секунды	0-59	V7766
Минуты	0-59	V7767
Час	0-23	V7770

Тип данных операнда		**Диапазон DL06**
	A	**aaa**
V-память	V	Смотри карту памяти

В следующем примере, когда C0 включен, константа (K73000) загружается в аккумулятор, используя команду Load Double (C0 должен быть контактом связанным с командой одиночного (PD)). Значение в аккумуляторе выводится в V2000, используя команду Out Double. Команда TIME использует значение в V2000, чтобы установить время в процессоре микроконтроллера.



Набор на ручном программаторе

$ STR	→	NEXT	NEXT	NEXT	NEXT	A 0	ENT
SHFT	L ANDST	D 3	D 3	→	PREV	H 7	D 3	A 0	A 0	A 0	ENT
A 0	D 3	A 0	B 1	ENT			
GX OUT	SHFT	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	T MLR	SHFT	I 8	M ORST	E 4	→	C 2	A 0	A 0	A 0	ENT

Команды управления процессором

No Operation (NOP)

No Operation – пустая (непрограммируемая) ячейка памяти

Direct SOFT32

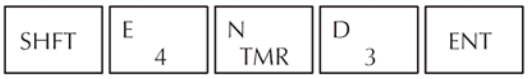
Набор на ручном программаторе

End (END)

Команда End отмечает точку окончания обычного сканирования программы. Команда End требуется в конце основного тела программы. Если команда End пропущена, то возникнет ошибка и ЦПУ не перейдет в режим Run. Метки данных, подпрограммы и программы прерывания находятся после команды End. Команда End — безусловная, поэтому для нее не требуется входной контакт.

Direct SOFT32

Набор на ручном программаторе

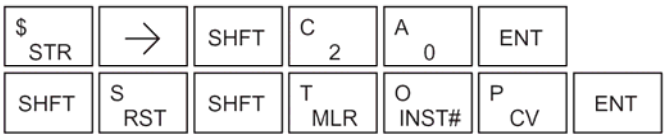


Stop (STOP)

Команда Stop меняет режим работы ЦПУ с режима Run на режим Program (Stop). Эта команда обычно используется, чтобы остановить работу ПЛК при сбоях. В следующем примере, когда C0 включено, ЦПУ останавливает работу и переключается на режим программирования

DirectSOFT32

Набор на ручном программаторе



Флаги	Описание
SP16	«1», когда DL06 находится в режиме TERM_PRG
SP53	«1», когда DL06 находится в режиме PRG

Reset Watch Dog Timer (RSTWT)

Команда Reset Watch Dog Timer сбрасывает таймер сканирования ЦПУ. Настройка по умолчанию для сторожевого таймера — 200 мс. Время сканирования очень редко превышает 200 мс, но это возможно. Циклы For/Next, подпрограммы, программа обработки прерываний 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 ЦПУ перейдет в режим програмированияProgram. Размещение команды RSTWT в программе очень важно. Команда должна быть выполнена до того момента, когда время сканирования превысит установку сторожевого таймера.

Если время сканирования гарантированно больше установки сторожевого таймера, значение времени ожидания может быть увеличено выше значения по умолчанию 200 мс, используя AUX55 на ручном программаторе или соответствующую дополнительную функцию в программном пакете. Эта установка устраняет необходимость в команде RSTWT.

В следующем примере таймер сканирования ЦПУ будет установлен в 0, когда команда RSTWT выполнена. Смотри команду For/Next для подробного примера

Direct SOFT 32

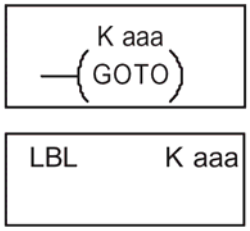
Набор на ручном программаторе

SHFT	R ORN	S RST	T MLR	W ANDN	T MLR	ENT

Команды управления программой

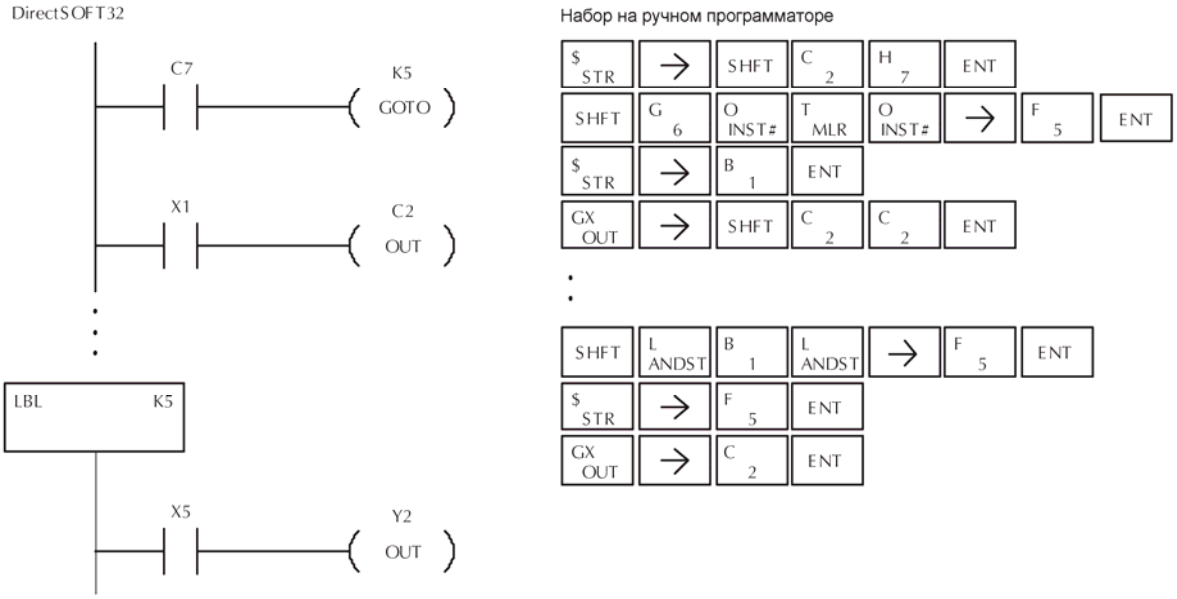
Goto Label (GOTO) (LBL)

Команда Goto / Label пропускает все инструкции между Goto и соответствующей командой LBL. Значение операнда для Goto и соответствующей команды LBL - одинаково. Когда команда Goto разрешается, логическая схема между Goto и командой LBL не выполняется. До 256 команд Goto и 256 команд LBL можно применять в программе.



Тип данных операнда		Диапазон DL06
		aaa
Константа	K	1-FFFF

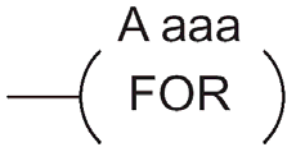
В следующем примере, когда C7 включен, вся логическая часть программы между GOTO и соответствующей командой LBL (обозначенная константой Kaaa) будет пропущена. Пропущенные команды не будут выполнены ЦПУ



For(FOR) / Next (NEXT)

Команды For и Next используются для выполнения участка программы на релейной логики между командами For и Next указанное число раз. Когда разрешена команда For, программа будет циклически повторяться указанное число раз.

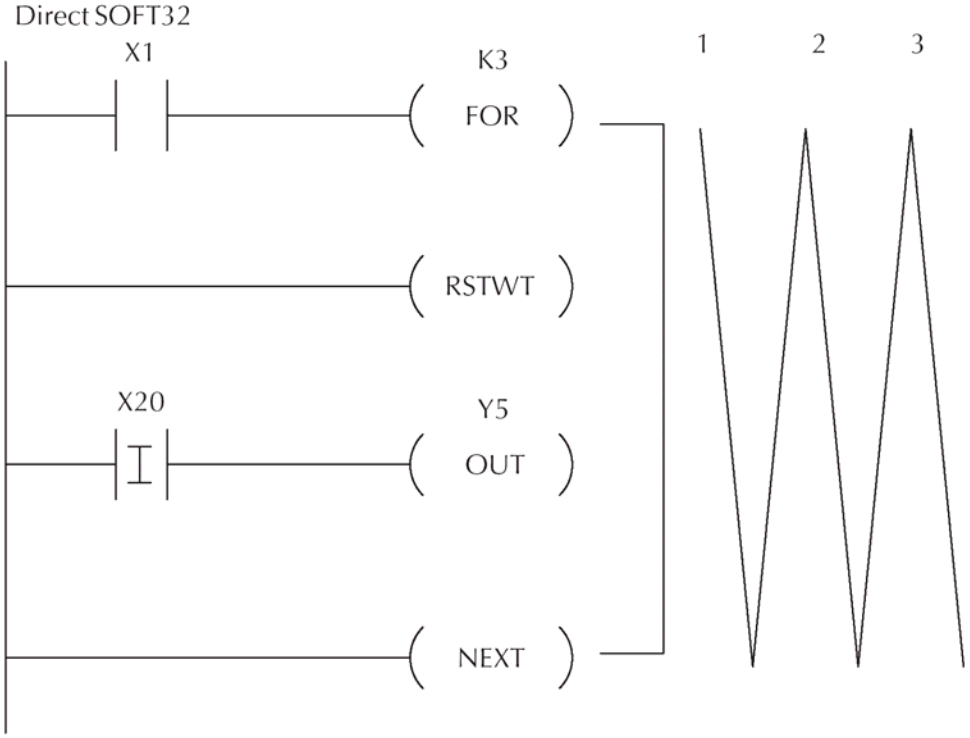
Если команда For не включена, то раздел релейной логики между командами For и Next не выполняется.



Команды For и Next не могут быть вложены. Во время выполнения цикла For / Next обычное обновление входов/выходов и служебные действия ЦПУ откладываются. Время сканирования программы 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 DL06
	A	**aaa**
V-память	V	Смотри карту памяти
Константа	K	1-9999

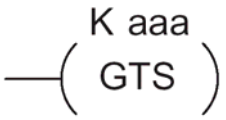
В следующем примере, когда X1 включен, прикладная программа внутри цикла For / Next будет выполняется три раза. Если X1 выключен, программа внутри цикла не будет выполнятся. Непосредственные команды могут быть нужны или не нужны в зависимости от Вашей прикладной программы. Также команда RSTWT не требуется, если цикл For / Next не увеличивает время сканирования больше установленного сторожевого таймера. Для более подробной информации относительно сторожевого таймера обратитесь к команде RSTWT



Набор на ручном программаторе

$ STR	→	B 1	ENT			
SHFT	F 5	O INST#	R ORN	→	D 3	ENT
SHFT	R ORN	S RST	T MLR	W ANDN	T MLR	ENT
$ STR	SHFT	I 8	→	C 2	A 0	ENT
GX OUT	→	F 5	ENT			
SHFT	N TMR	E 4	X SET	T MLR	ENT	

Goto Subroutine (GTS) (SBR)



Команда Goto Subroutine позволяет части программы релейной логики быть помещенной вне основного тела программы, чтобы выполнять ее только при необходимости. В программе может использоваться максимум 256 команд GTS и 256 команды SBR. Команды GTS могут быть вложены до 8 уровней. Если этот максимальный предел будет превышен, то произойдёт ошибка E412. Обычно эти команды используются в прикладных программах, когда блок логики программы может выполняться медленно, и его не требуется выполнять при каждом сканировании. Метка подпрограммы и вся связанная логика помещается после команды End в программе. Когда подпрограмма вызывается из главной программы, ЦПУ будет выполнять подпрограмму (SBR) с той же самой константой (K), что и команда GTS, которая вызвает подпрограмму.

Помещенный в подпрограмму код сканируется и выполняется только при необходимости, так как он находится после команды End. Код, который не сканируется, не влияет на общее время сканирования программы.

Subroutine Return (RT)

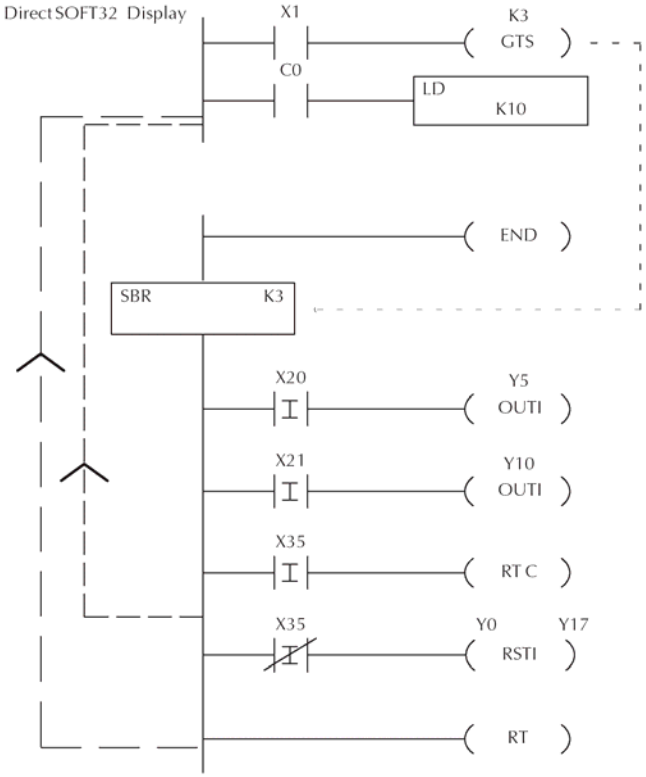
Тип данных операнда		Диапазон DL06
		aaa
Константа	K	1-FFFF

Когда в подпрограмме выполняется команда Subroutine Return, ЦПУ 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)

Команда Subroutine Return Conditional — дополнительная команда, используемая с входным контактом для выполнения условного возврата из подпрограммы. Команда Subroutine Return (RT) также требуется для завершения подпрограммы.

В следующем примере, когда X1 включен, будет вызвана подпрограмма K3. ЦПУ перейдет к метке подпрограммы K3, и будет выполняться релейная логика в подпрограмме. Если X35 включен, то ЦПУ вернется в главную программу по команде RTC. Если X35 не включен, то Y0-Y17 будут сброшены, и затем ЦПУ вернется в главное тело программы



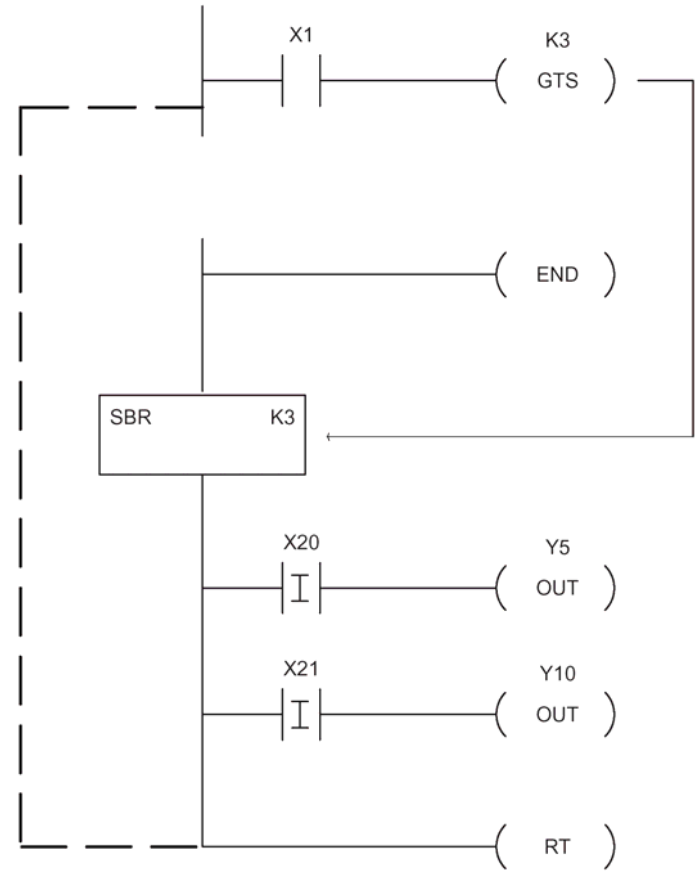
Набор на ручном программаторе

$ STR	→	B 1	ENT				
SHFT	G 6	T MLR	S RST	→	D 3	ENT	

SHFT	E 4	N TMR	D 3	ENT			
SHFT	S RST	SHFT	B 1	R ORN	→	D 3	ENT
$ STR	SHFT	I 8	→	C 2	A 0	ENT	
GX OUT	SHFT	I 8	→	F 5	ENT		
$ STR	SHFT	I 8	→	C 2	B 1	ENT	
GX OUT	SHFT	I 8	→	B 1	A 0	ENT	
$ STR	SHFT	I 8	→	D 3	F 5	ENT	
SHFT	R ORN	T MLR	C 2	ENT			
SP STRN	SHFT	I 8	→	D 3	F 5	ENT	
S RST	SHFT	I 8	→	A 0	→	B 1	H 7	ENT
SHFT	R ORN	T MLR	ENT				

В следующем примере, когда X1 включен, будет вызвана подпрограмма К3. ЦПУ перейдет к метке подпрограммы К3, и будет выполняться релейная логика в подпрограмме. ЦПУ вернется в главное тело программы после выполнения команды RT.

Direct SOFT32



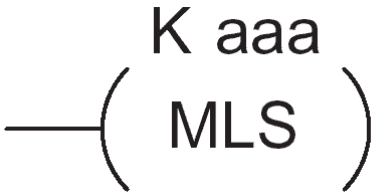
Набор на ручном программаторе

$ STR	→	B 1	ENT				
SHFT	G 6	T MLR	S RST	→	D 3	ENT	

SHFT	E 4	N TMR	D 3	ENT			
SHFT	S RST	SHFT	B 1	R ORN	→	D 3	ENT
$ STR	SHFT	I 8	→	C 2	A 0	ENT	
GX OUT	→	F 5	ENT				
$ STR	SHFT	I 8	→	C 2	B 1	ENT	
GX OUT	→	B 1	A 0	ENT			
SHFT	R ORN	T MLR	ENT				

Master Line Set (MLS)

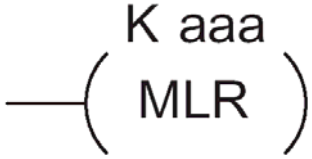
Команда Master Line Set позволяет программе управлять разделами релейной логики, формируя новую шину питания, управляемую главной левой шиной питания. Главная левая шина всегда имеет номер 0. Когда используется команда MLS K1, на уровне 1 создается новая шина. Команды Master Line Set и Master Line Reset могут быть использованы для того, чтобы создать вложенное множество шин вплоть до семи уровней вложенности.



Тип данных операнда	Диапазон DL06
	aaa
Константа K	1-7

Master Line Reset (MLR)

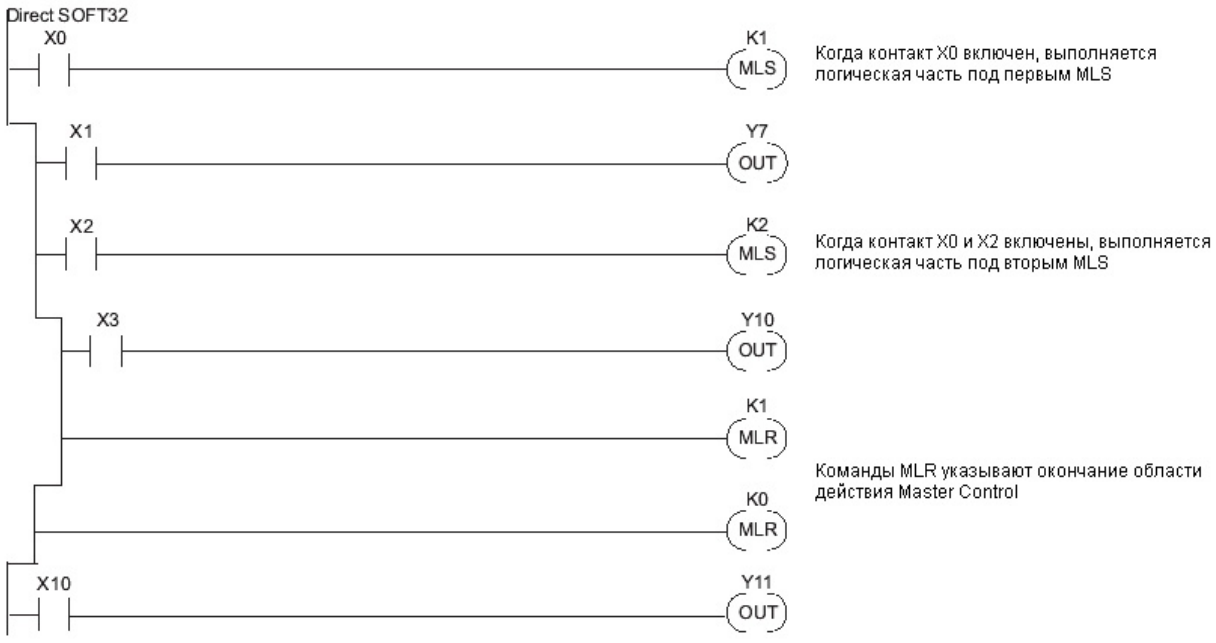
Команда Master Line Reset отмечает конец элемента управления для соответствующей команды MLS. Команда MLR имеет ссылку на единицу меньше, чем соответствующая команда MLS



Тип данных операнда	Диапазон DL06
	aaa
Константа K	1-7

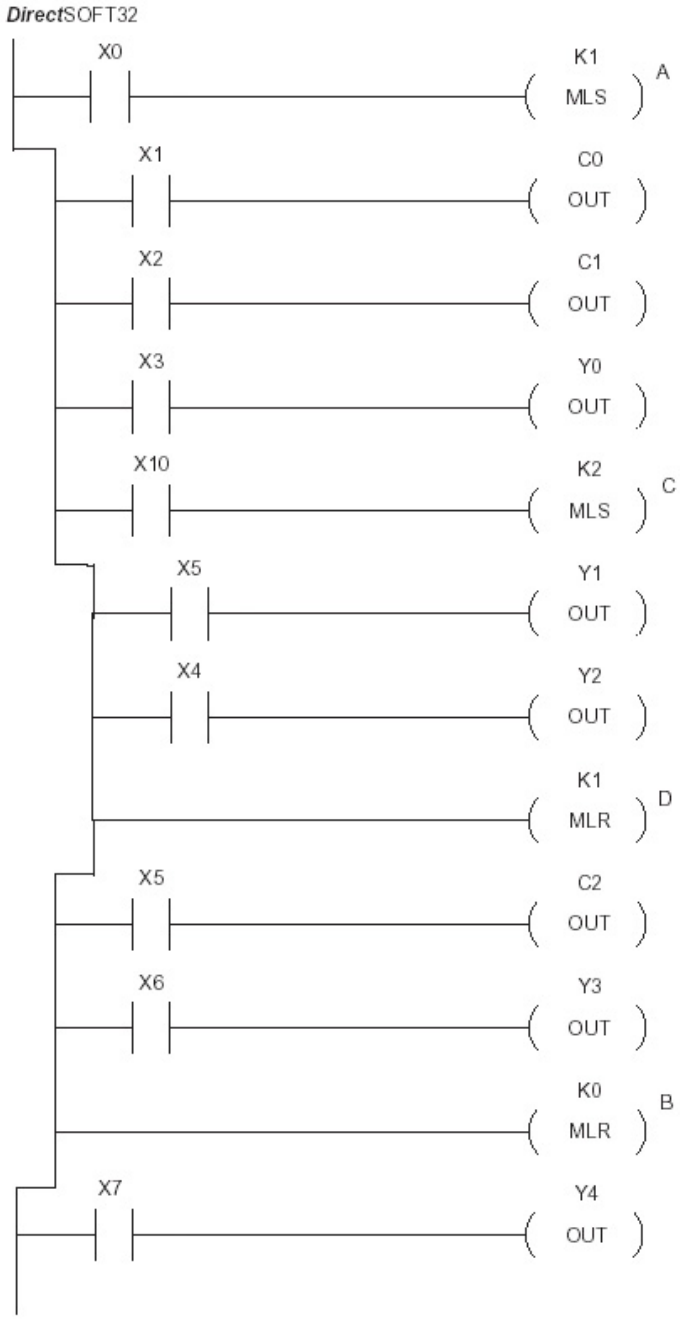
Объяснение работы Master Control Relays

Master Line Set (MLS) и Master Line Reset (MLR) позволяют Вам быстро включать (или отключать) разделы программы RLL. Это обеспечивает гибкость управления программой. Следующий пример показывает, как работают команды MLS и MLR, создавая шины питания для управляющей логики.



Пример MLS/MLR

В следующем примере MLS/MLR логика между первой MLS K1 (A) и MLR K0 (B) будет выполняться только тогда, когда вход X0 включен. Логическая часть между MLS K2 (C) и MLR K1 (D) будет выполняться только тогда, когда входы X10 и X0 включены. Последнее звено не управляется ни одной из команд MLS.

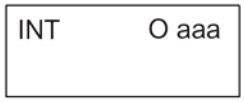


Набор на ручном программаторе

$ STR	→	A 0	ENT		
Y MLS	→	B 1	ENT		
$ STR	→	B 1	ENT		
GX OUT	→	SHFT	C 2	A 0	ENT
$ STR	→	C 2	ENT		
GX OUT	→	SHFT	C 2	B 1	ENT
$ STR	→	D 3	ENT		
GX OUT	→	A 0	ENT		
$ STR	→	B 1	A 0	ENT	
Y MLS	→	C 2	ENT		
$ STR	→	F 5	ENT		
GX OUT	→	B 1	ENT		
$ STR	→	E 4	ENT		
GX OUT	→	C 2	ENT		
T MLR	→	B 1	ENT		
$ STR	→	F 5	ENT		
GX OUT	→	SHFT	C 2	C 2	ENT
$ STR	→	G 6	ENT		
GX OUT	→	D 3	ENT		
T MLR	→	A 0	ENT		
$ STR	→	H 7	ENT		
GX OUT	→	E 4	C 2	ENT	

Команды прерывания

Interrupt (INT)



Команда Interrupt позволяет поместить раздел релейной логики вне главного тела программы и выполнять его при необходимости. Высокоскоростной Ввод/вывод в режимах 10, 20 и 40 может генерировать прерывание.
В режиме 40, Вы можете выбрать внешнее прерывание (вход X0) или прерывание по времени (3-999мс).
Обычно прерывания используются в прикладной программе, когда требуется быстрый отклик на вход, или раздел программы должен выполниться быстрее, чем обычный цикл сканирования ЦПУ. Метка прерывания и вся связанная логика должны быть помещены в программе после команды End. Когда встретиться прерывание, ЦПУ завершит выполнение текущей команды, которую оно обрабатывает в релейной логике, и затем будет выполнять указанную процедуру прерывания. Выполнение программы продолжится с того места, где оно было прервано.
Подробности о работе с прерыванием смотри См. Главу 3, раздел - Режим 40 (Прерывания). У контроллера DL06, только одно программное прерывание доступно. Программное прерывание использует прерывание #00 (INT 0), которое имеет значение аппаратного прерывания #0. Программное и аппаратное прерывание не может использоваться вместе. Аппаратные прерывания помечены в восьмеричном формате, чтобы соответствовать входному сигналу аппаратных средств (например, X1 инициализирует INT 1).

Тип данных операнда	Диапазон DL06
	aaa
Константа O	0-3

Interrupt Return (IRT)

Interrupt Return ставится последней командой в процедуре прерывания. ЦПУ вернется в ту точку главного тела программы, из которой программа прерывания была вызвана. Interrupt Return автономная команда (в звене нет входного контакта).

Interrupt Return Conditional (IRTC)

Команда Interrupt Return Conditional — дополнительная команда, используемая с входным контактом, для выполнения условного возврата из программы прерывания. Команда Interrupt Return требуется для завершения процедуры прерывания.

Enable Interrupts (ENI)

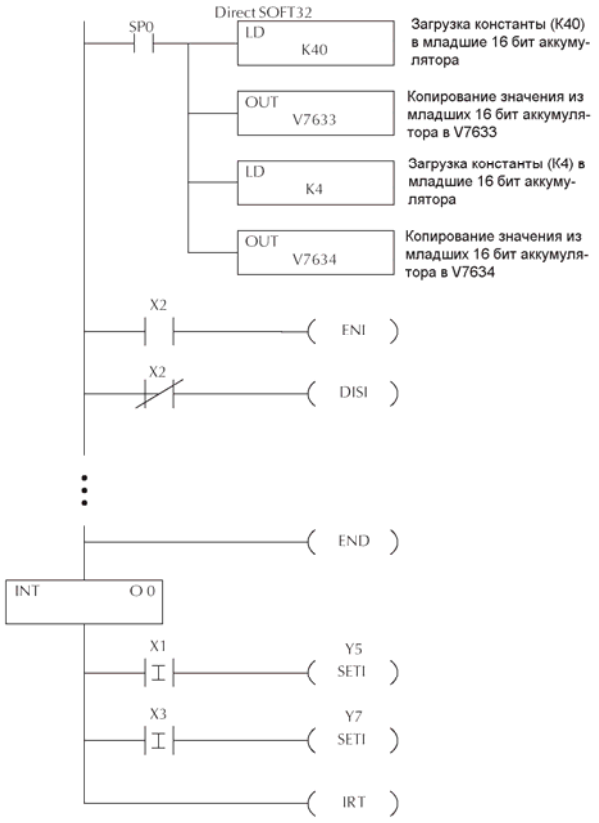
Команда Enable Interrupt применяется в главном теле прикладной программы (перед командой End), чтобы разрешить аппаратные или программные прерывания. Прерывания разрешены, пока они не будут запрещены командой Disable Interrupt.

Disable Interrupts (DISI)

Команда Disable Interrupt применяется в главном теле прикладной программы (перед командой End), чтобы блокировать прерывания (внешнее или прерывание по времени). Прерывания запрещены, пока они не будут разрешены командой Enable Interrupt.

Пример программы внешнего прерывания.

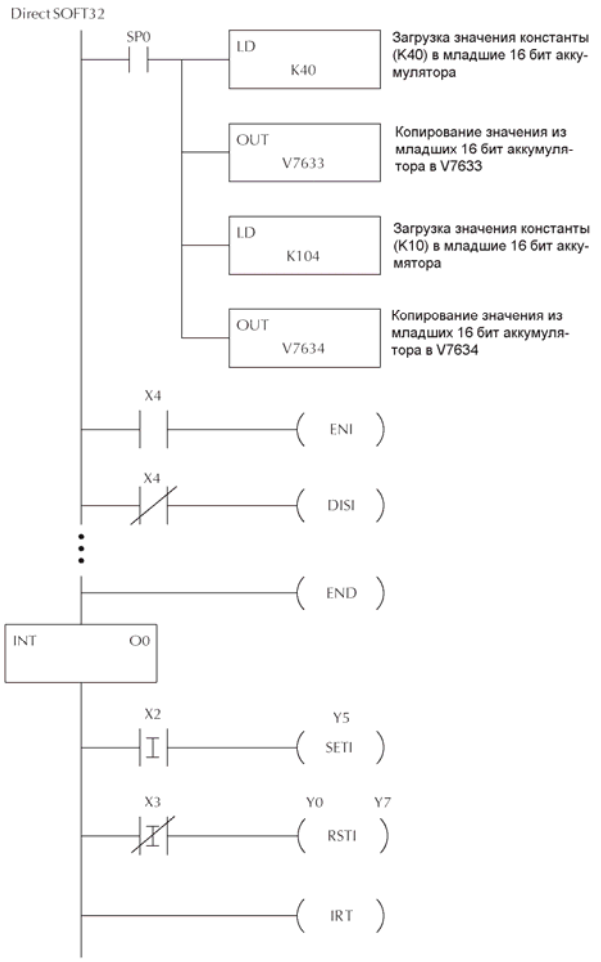
В следующем примере, мы делаем некую инициализацию при первом сканировании, используя контакт первого сканирования SP0. Вариант прерывания - Высокоскоростной Ввод/вывод в режиме 40. Затем мы конфигурируем X0, как внешнее прерывание, делая запись в регистр конфигурации V7634. Подробности: смотри Главу 3, Режим 40.
Во время исполнения программы, когда X2 включен, прерывания разрешены. Когда X2 выключен, прерывания запрещены. После получении сигнала прерывания(X0), ЦПУ перейдет к метке прерывания INT O0. В процедуре прерывания будет выполняться релейная логика. ЦПУ вернется к главному телу программы после выполнения команды IRT



Набор на ручном программаторе

Пример программы прерывания по времени

В следующем примере мы делаем инициализацию на первом сканировании, используя контакт первого сканирования SP0. Вариант прерывания - Высокоскоростной Ввод/Вывод в режиме 40. Затем мы конфигурируем таймер Высокоскоростного Ввода/Вывода, как прерывание 10 мс, записывая константу К104 в регистр конфигурации Х0 (V7634). Подробности: смотри Главу 3, Режим 40.
Когда Х4 включен, прерывание допускается. Когда Х4 выключен, прерывание заблокировано. Каждые 10 мс ЦПУ будет переходить к метке прерывания INT О0. В процедурее прерывания будет выполняться прикладная задача релейной логики. Если Х35 не включен, то Y0-Y17 будут сброшены и затем ЦПУ вернется к главному телу программы



Набор на ручном программаторе

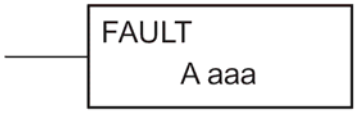
$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	SHFT	K JMP	E 4	A 0	ENT	
GX OUT	→	SHFT	V AND	H 7	G 6	D 3	D 3	ENT	
SHFT	L ANDST	D 3	→	SHFT	K JMP	B 1	A 0	E 4	ENT
GX OUT	→	SHFT	V AND	H 7	G 6	D 3	E 4	ENT	
$ STR	→	E 4	ENT						
SHFT	E 4	N TMR	I 8	ENT					
SP STRN	→	E 4	ENT						
SHFT	D 3	I 8	S RST	I 8	ENT				

SHFT	E 4	N TMR	D 3	ENT			
SHFT	I 8	N TMR	T MLR	→	A 0	ENT	
$ STR	SHFT	I 8	→	C 2	ENT		
X SET	SHFT	I 8	→	F 5	ENT		
SP STRN	SHFT	I 8	→	D 3	ENT		
X SET	SHFT	I 8	→	A 0	→	H 7	ENT
SHFT	I 8	R ORN	T MLR	ENT			

Команды вывода сообщений

Fault (FAULT)

Команда Fault используется для отображения сообщения на ручном программаторе, дополнительном жидкокристаллическом дисплее или в строке текущего состояния DirectSOFT. Сообщение может иметь максимум 23 знака и может быть либо данными V-памяти, либо числовой константой или ASCII текстом.



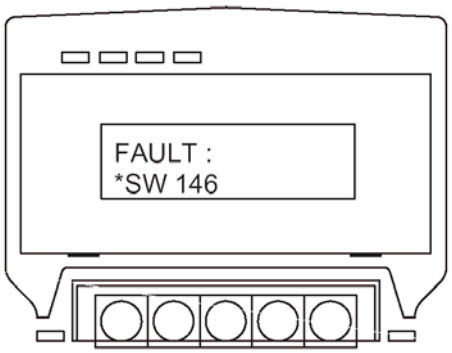
Чтобы отобразить значение в ячейке V-памяти, укажите в команде ячейку V-памяти. Чтобы отобразить данные в командах ACON (ASCII константа) или NCON (числовая константа), укажите значение константы (K) для соответствующей области метки данных.

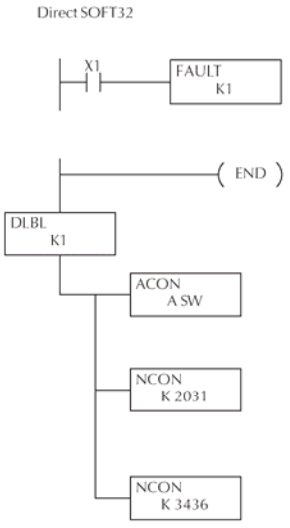
Тип данных операнда		Диапазон DL06
	A	**aaa**
V-память	V	Смотри карту памяти
Константа	K	0-FFFF

Флаги	Описание
SP50	«1», когда команда FAULT выполнена

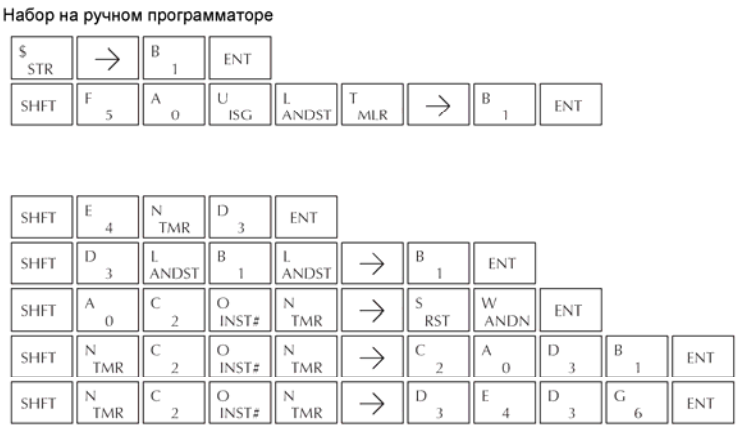
Пример команды Fault

В следующем примере, когда X1 включен, сообщение SW 146 будет отображено на дисплее ручного программатора. Команды NCON используют шестнадцатиричный ASCII эквивалент текста для отображения. (HEX эквивалент ASCII для пробела — 20, для 1 - 31, для 4 - 34 …)





Набор на ручном программаторе



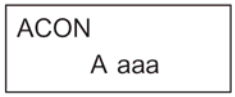
Data Label (DLBL)

Команда Data Label отмечает начало ASCII / числовой области данных. Команды DLBL располагаются в программе после оператора End. В программе может использоваться максимум 64 команды DLBL. Команды NCON и ACON могут многократно использоваться в области DLBL

Тип данных операнда		Диапазон DL06
		aaa
Константа	K	1-FFFF

ASCII Constant (ACON)

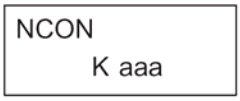
Команда ASCII Constant используется с командой DLBL, чтобы сохранить ASCII текст для использования с другими командами. Два ASCII символа могут быть сохранены в команде ACON. Если в ACON хранится только один символ, то первым знаком будет вставлен пробел.



Тип данных операнда		Диапазон DL06
		aaa
ASII	A	0-9 A-Z

Numerical Constant (NCON)

Команда Numerical Constant используется с командой DLBL, чтобы сохранить шестнадцатиричный ASCII эквивалент цифровых данных для использования с другими командами. Две цифры могут быть сохранены в команде NCON.

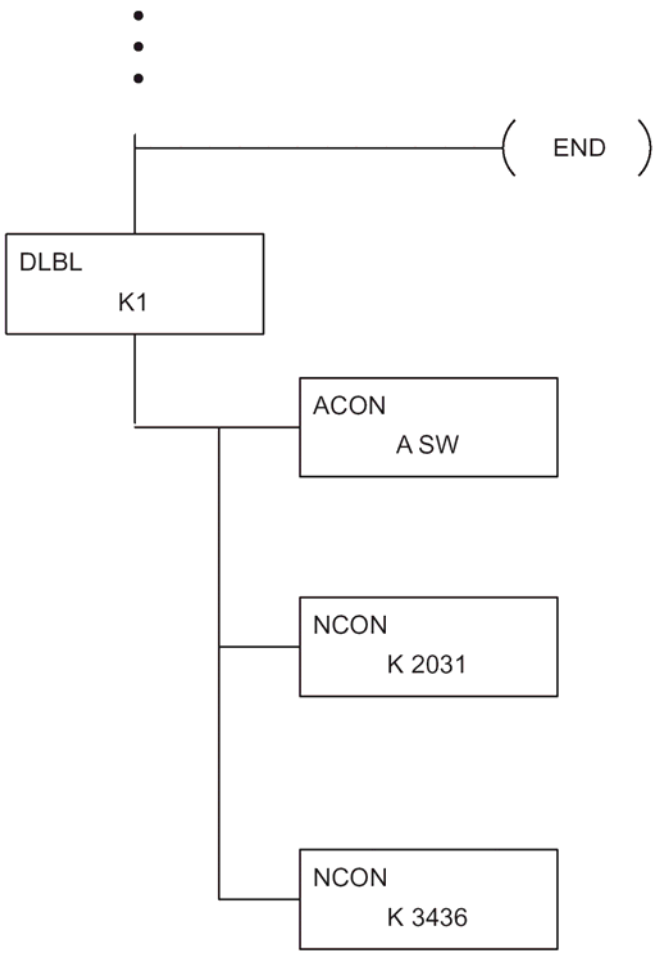


Тип данных операнда		Диапазон DL06
		aaa
Константа	K	0-FFFF

Пример команды Data Label

В следующем примере, команда ACON и две команды NCON используются внутри команды DLBL для построения текстового сообщения. Смотрите команду FAULT для информации по отображению сообщений. В руководстве на панель DV-1000 также есть информация о показываемых сообщениях

Direct SOFT32

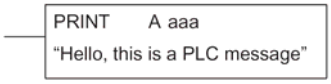


Набор на ручном программаторе

SHFT	E 4	N TMR	D 3	ENT						
SHFT	D 3	L ANDST	B 1	L ANDST	→	B 1	ENT			
SHFT	A 0	C 2	O INST#	N TMR	→	S RST	W ANDN	ENT		
SHFT	N TMR	C 2	O INST#	N TMR	→	C 2	A 0	D 3	B 1	ENT
SHFT	N TMR	C 2	O INST#	N TMR	→	D 3	E 4	D 3	G 6	ENT

Print Message (PRINT)

Команда Print Message печатает вложенный текст или сообщение с текстом / переменными в указанный коммуникационный порт (для DL06 – порт 2), который должен быть конфигурируемым коммуникационным портом.

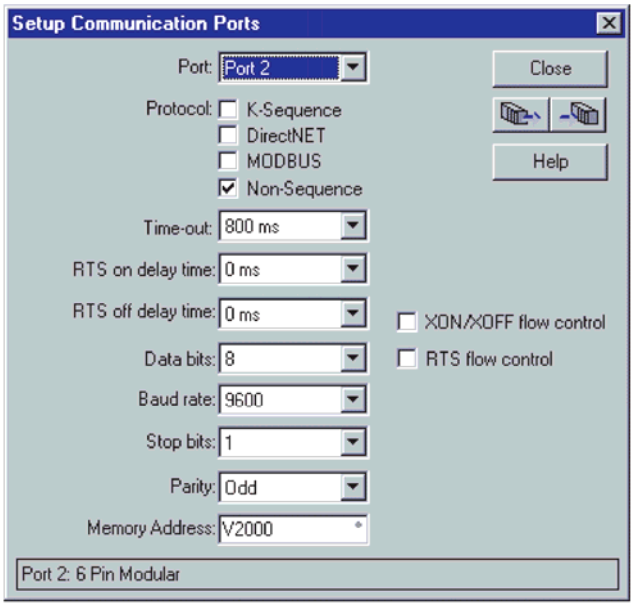


Тип данных операнда	Диапазон DL06
	aaa
Константа K	2

Вы можете вспомнить из спецификации процессора в Главе 3, что порты DL06 способны работать с несколькими протоколами. Порт 1 не может быть сконфигурирован для непроцедурного(Non-sequence) протокола. Для настройки порта 2 с помощью Ручного Программатора, используйте AUX 56 и следуйте подсказкам, делая те же действия, как показано ниже на этой странице. Для настройки порта при помощи DirectSOFT, войдите в меню PLC, затем в Setup, затем Setup Secondary Comm Port.

Port. В списке номеров портов выберите «Port 2».

Protocol. Щелкните по окошку слева от «Non-sequence», и затем Вы увидите диалоговое окно, показанное ниже



- **Baud Rate.** Выберите скорость передачи, которая соответствует Вашему принтеру.
- **Stop Bits, Parity**. Выберите число стоп-бит и установке контроля четности в соответствии с Вашим принтером
- **Memory Address.** Выберите адрес V-памяти для DirectSOFT32 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 необходимо.

Затем нажмите показанную клавишу, чтобы отправить установки Порта 2 в процессор, и нажмите Close. Потом посмотрите Главу 3 о том, как соединить Ваш принтер с DL06

Порт 2 в DL06 имеет стандартные RS232 уровни сигналов и должен работать с большинством последовательных портов.

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. Два знака, стоящие после знака $ интерпретируются в соответствии со следующей таблицей:

#	Знаковый код	Описание
1	$$	Знак доллара ($)
2	$”	Двойные кавычки (”)
3	$L или $l	Перевод строки (LF)
4	$N или $n	Возврат каретки, перевод строки (CRLF)
5	$P или $p	Перевод страницы
6	$R или $r	Возврат каретки (CR)
7	$T или $t	Табуляция

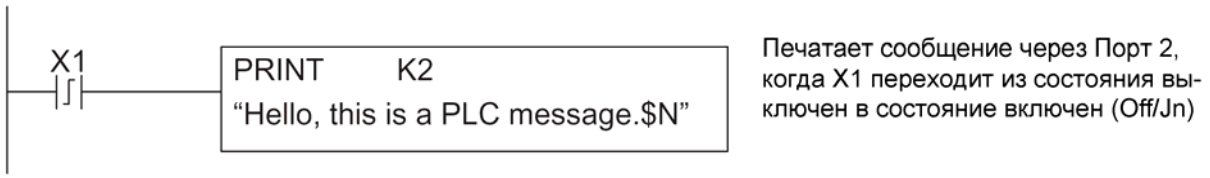
Следующие примеры показывают различные синтаксические соглашения и длину строки, выводимой на принтер.

Пример:

" "	Длина 0 без символа
"A"	Длина 1 с символом «A»
" "	Длина 1 с символом «пробел»
" $" "	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, Вы будете должны включить двойные кавычки перед и после текстовой строки. Если команда печати содержит некорректный текст или отсутствуют кавычки, то в ЦПУ возникнет код ошибки 499. Важно проверять Ваши данные в команде PRINT во время разработки приложения.

Следующий пример печатает сообщение в порт 2. Мы используем PD контакт, который выполняет команду Message только в одном цикле сканирования. Заметьте 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.



Элемент V-памяти — используется для печати содержимого V-памяти в целом или вещественном формате. Используйте номер V-памяти или номер V-памяти с «:» и тип данных. Типы данных показаны в таблице ниже. Символьный код должен состоять из заглавных букв.

ПРИМЕЧАНИЕ. Чтобы отделить адрес V-памяти от текстовой строки, перед и после адреса должен стоять пробел. Если этого не сделать, то возникнет код ошибки 499.

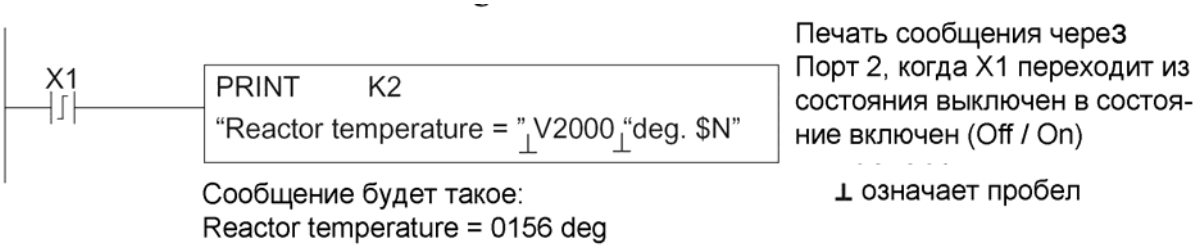
#	Знаковый код	Описание
1	нет	16-битный двоичный (десятичное число)
2	: B	4- цифровое BCD число
3	: D	32-битный двоичный (десятичное число)
4	: D B	8-цифровое BCD число

Пример:

V2000	Печать двоичных данных в V2000 для десятичного числа
V2000 : B	Печать BCD данных в V2000
V2000 : D	Печать двоичного числа в V2000 и V2001 для десятичного числа
V2000 : D B	Печать BCD данных в V2000 и V2001

Пример:

Следующий пример печатает сообщение, содержащее текст и переменную. «Температура реактора» обозначает данные, которые находятся в V2000. Вы можете использовать классификатор :B после V2000, если, например, данные в BCD формате. В конце строки добавляются 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 % перед числом знаков после номера V-памяти для представления текста. Если Вы присвоили «0» числу знаков, функция печати начнет отсчет знаков с первой ячейки. Затем она начнет со следующей ячейки V-памяти и будет читать число ASCII кодов в тексте из памяти.

Пример:

V2000 % 16	Печать 16 знаков в V2000-V2007
V2000 % 0	Печать знаков в V2001-Vxxxx (определяется числом в V2000)

Битовый элемент — используется для печати состояния обозначенного бита в V-памяти или релейных битов. Битовый элемент может быть назначен указанием точки (.) и номера бита вслед за номером V-памяти или номером реле (C). Тип вывода описывается как показано в таблице ниже.

#	Знаковый код	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 1/0 формате
C100 Печать состояния C100 в 1/0 формате
C100 : BOOL Печать состояния C100 в TRUE/FALSE формате
C100 : ON/OFF Печать состояния C100 в TRUE/FALSE формате
V2000 . 15 : BOOL Печать состояния бита 15 в V2000 в TRUE/FALSE формате

Максимальное число знаков, которое Вы можете напечатать — 128. Число знаков для каждого элемента показано в таблице ниже

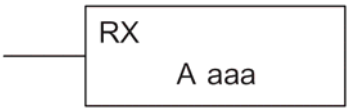
Тип элемента	**Максимальное число знаков**
Текст, 1 символ	1
16-битный двоичный	6
32-битный двоичный	11
4-знаковый BCD	4
8-знаковый BCD	8
С плавающей точкой (вещественное число)	12
С плавающей точкой (вещественное число в экспоненциальной форме)	12
V-память / текст	2
Бит (1/0 формат)	1
Бит (TRUE/FALSE формат)	5
Бит (ON/OFF формат)	3

Функция «PRINT» на ручном программаторе следует за клавишей DEF.
Флаги специальных реле SP116 и SP117 показывают состояние портов ЦПУ DL06 (занят, или ошибка связи). Для подробного описания смотрите приложение по специальным реле.

ПРИМЕЧАНИЕ. Вы должны использовать соответствующие специальные реле в соединении командой PRINT, чтобы гарантировать, что релейная программа не будет пытаться печатать в порт, который все еще занят предыдущими командами PRINT, WX или RX.

Read from Network (RX)

Команда Read from Network используется ведущим (master) устройством в сети для чтения блока данных с подчиненного устройства 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 Read from Network



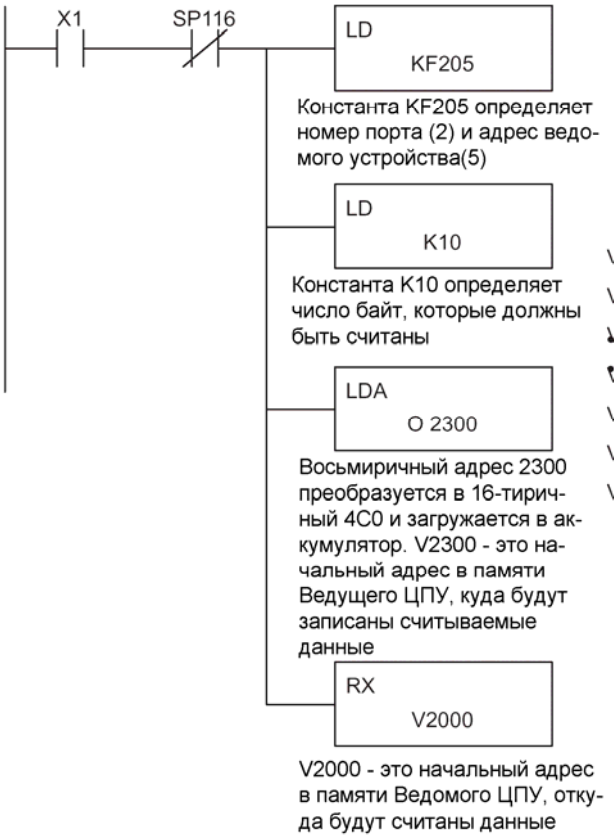
- **Шаг 1:** загрузите адрес подчиненного (slave) устройства (0 -- 90 BCD) в первый байт и внутренний порт ПЛК (KF2) или номер слота ведущего модуля DCM или ECOM (0--7) во второй байт второго уровня стека аккумулятора
- **Шаг 2:** загрузите число считываемых байт в первый уровень стека аккумулятора.
- **Шаг 3:** загрузите адрес, куда будут помещены считываемые данные, в аккумулятор данных. Этот параметр должен быть шестнадцатиричным значением.
- **Шаг 4:** вставьте команду RX, которая указывает стартовую ячейку V-памяти (Vaaa), откуда данные будут считываться из подчиненного устройства.

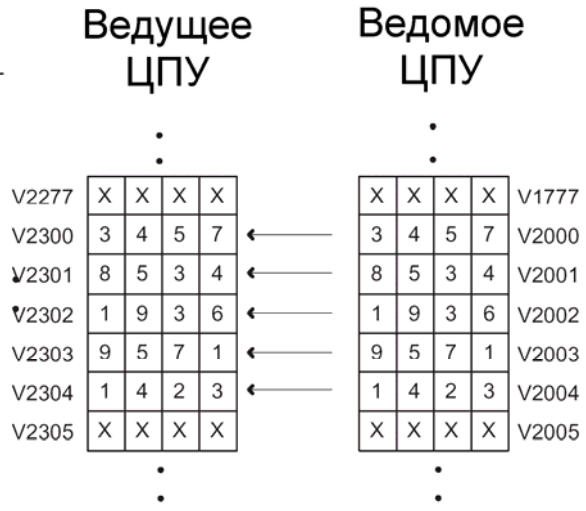
Полезная подсказка: Для параметров, которые требуют шестнадцатиричных значений, может быть использована команда LDA для преобразования восьмеричного адреса в шестнадцатиричный эквивалент и загрузки значения в аккумулятор.

Тип данных операнда		**Диапазон DL06**
	A	**aaa**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Входы	X	0-777
Выходы	Y	0-777
Реле управления	C	0-1777
Стадии	S	0-1777
Таймер	T	0-377
Счетчик	CT	0-177
Специальные реле	SP	0-777
Память программы	$	0-7680 (2К памяти программы)

В следующем примере, когда X1 включен, а реле занятости порта SP116 (смотрите «Специальные реле») не включено, команда RX обратится к порту 2, работающему как ведущий (master). Десять последовательных байт данных (V2000-V2004) будут считываться из устройства сети с адресом 5 и копироваться в ячейки V-памяти V2300-V2304 ведущего (master) устройства.

Direct SOFT32

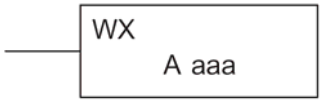




Набор на ручном программаторе

$ STR	→	B 1	ENT									
W ANDN	→	SHFT	SP STRN	B 1	B 1	G 6	ENT					
SHFT	L ANDST	D 3	→	SHFT	K JMP	SHFT	F 5	SHFT	C 2	A 0	F 5	ENT
SHFT	L ANDST	D 3	→	SHFT	K JMP	B 1	A 0	ENT				
SHFT	L ANDST	D 3	A 0	→	C 2	D 3	A 0	A 0	ENT			
SHFT	R ORN	X SET	→	C 2	A 0	A 0	A 0	ENT				

Write to Network (WX)



Команда Write to Network используется для записи блока данных с ведущего (master) устройства на подчиненное (slave) устройство в той же самой сети. Функциональные параметры загружаются в аккумулятор, в первый и второй уровень стека. Ниже перечислены шаги, необходимые для программирования функции Write to Network.

Шаг 1: Загрузите адрес подчиненного (slave) устройства (0-90 BCD) в младший байт и F2 в старший байт аккумулятора (следующие две команды помещают это слово во второй уровень стека аккумулятора).

Шаг 2: Загрузите число перемещаемых байтов в первый уровень стека аккумулятора(следующая команда помещает это слово на вершину стека).

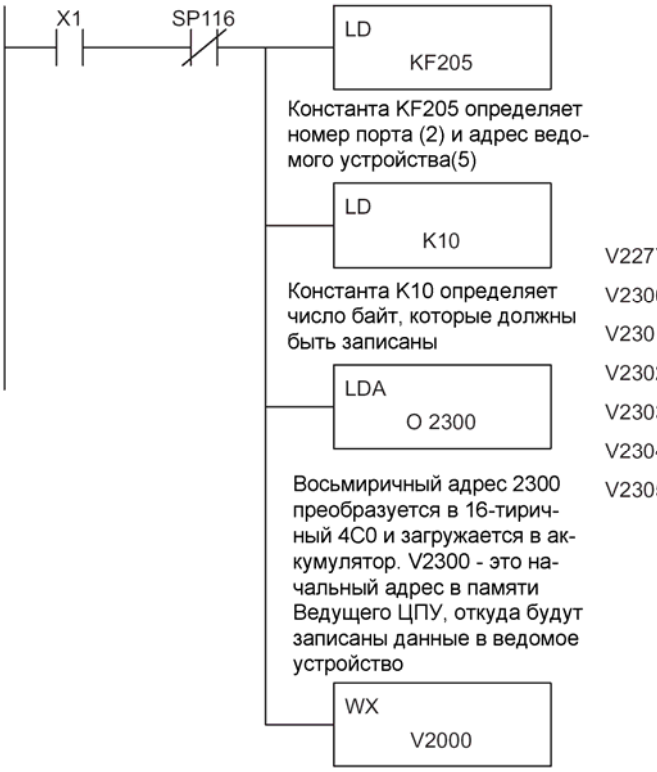
Шаг 3: Загрузите начальный адрес Ведущего устройства в аккумулятор. Это - ячейка памяти, откуда будут записаны данные. Этот параметр должен быть шестнадцатиричным значением.

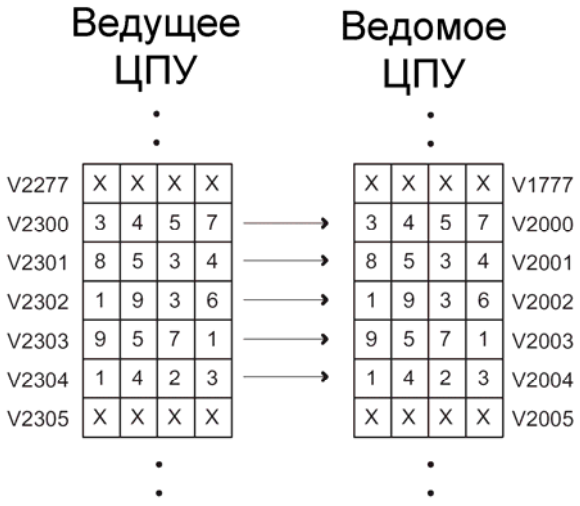
Шаг 4: Вставьте команду WX, которая указывает стартовую ячейку V-памяти (Аааа), куда данные будут записываться в подчиненном устройстве.

Полезная подсказка: Для параметров, которые требуют шестнадцатиричных значений, может быть использована команда LDA для преобразования восьмеричного адреса в шестнадцатиричный эквивалент и загрузки значения в аккумулятор.

Тип данных операнда		**Диапазон DL06**
	A	**ааа**
V-память	V	Смотри карту памяти
Указатель	P	Смотри карту памяти
Входы	X	0-777
Выходы	Y	0-777
Реле управления	C	0-1777
Стадии	S	0-1777
Таймер	T	0-377
Счетчик	CT	0-177
Специальные реле	SP	0-777
Память программы	$	0-7680 (2К памяти программы)

В следующем примере, когда X1 включен, а реле «Порт занят» SP116 (смотрите «Специальные реле») не включено, команда WX обратится к порту 2, работающему как ведущее (master) устройство. Десять последовательных байт данных будут считываться из Ведущего (master) ЦПУ и копироваться в ячейки V-памяти V2300-V2304 в ЦПУ Ведомого (slave) устройства с указанным адресом 5.



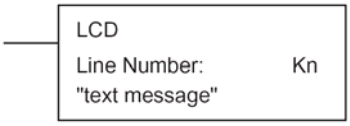


Набор на ручном программаторе

$ STR	→	B 1	ENT									
W ANDN	→	SHFT	SP STRN	B 1	C 1	E 6	ENT					
SHFT	L ANDST	D 3	→	SHFT	K JMP	SHFT	F 5	SHFT	C 2	A 0	F 5	ENT
SHFT	L ANDST	D 3	→	SHFT	K JMP	B 1	A 0	ENT				
SHFT	L ANDST	D 3	A 0	→	C 2	D 3	A 0	A 0	ENT			
SHFT	W ANDN	X SET	→	C 2	A 0	A 0	A 0	ENT				

LCD

Когда запускается, команда LCD то вызывается определяемое пользователем текстовое сообщение и отображается на индикаторной LCD-панели контроллера. Дисплей имеет ширину 16 символов и 2 строки в высоту, так что общее количество символов, которые могут быть одновременно отображены на панели равно 32. Каждая строка адресуется отдельно; поэтому максимальное число символов, которое допускает команда - 16.



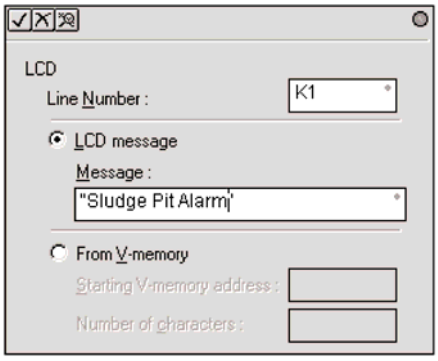
Текстовое сообщение может быть введено непосредственно в диалоге ввода команды, или в пользовательской ячейке V-памяти. Если текст находится в V-памяти, команда LCD используется, чтобы указать ячейку памяти, где располагается желательный текст. Также требуется ввести длину текстовой строки.

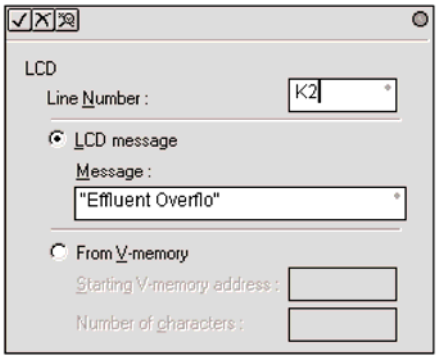
Используйте окно просмотра Instruction Browser, из папки проекта DirectSOFT32, для поиска команды LCD. Когда Вы выбирете инструкцию LCD и нажмете OK, появится диалоговое окно ввода команды LCD, как показано в примерах. Инструкция LCD вставляется в релейную программу через это окно задания параметров.

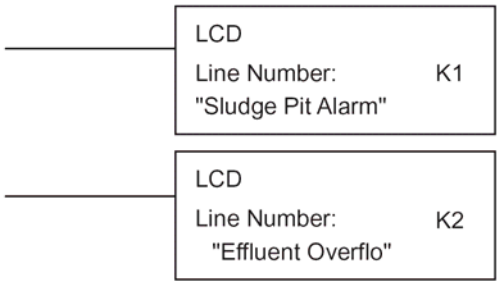
Отображаемые текстовые строки могут включать встроенные переменные. Установки даты и времени, а также значения ячеек V-memory могут быть встроенные в отображаемый текст. Примеры показаны ниже.

Непосредственный ввод текста

Два диалоговых окна справа показывают выбор, необходимый для создания двух команд релейной логики, представленных ниже. Двойные кавычки требуются, чтобы выделить текстовую строку. В первом окне, текст “Sludge Pit Alarm“ (Тревога Шламоотстойника) использует шестнадцать символов и будет показываться в строке 1, при выполнении команды. Обратите внимание, что номер строки - K1. Нажатие кнопки “check” (галочка) вставляет команду в программу релейной логики.





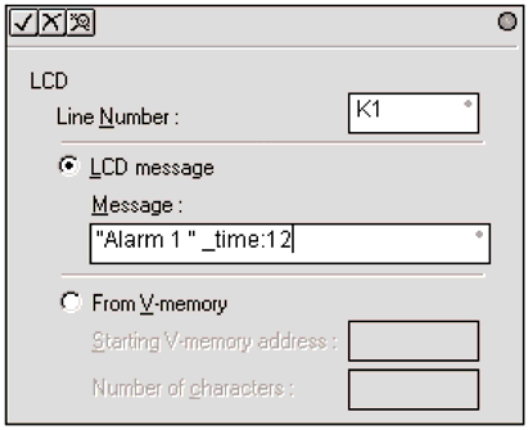


Определяем вторую строку K2, как текстовую строку “Effluent Overflow” (Переполнение Стока), которая будет появляться во второй строке дисплея, при выполнении второй команды LCD.

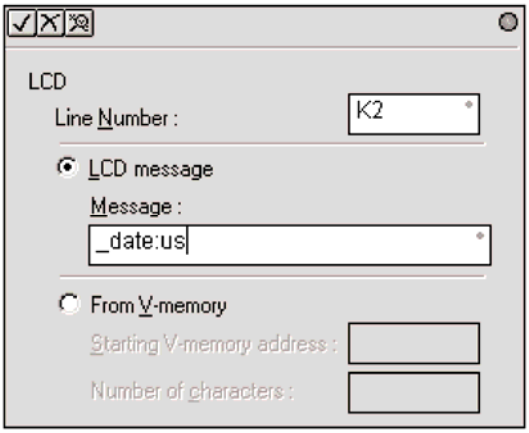
S	l	u	d	g	e		P	i	t		A	l	a	r
E	f	f	l	u	e	n	t		O	v	e	r	f	l

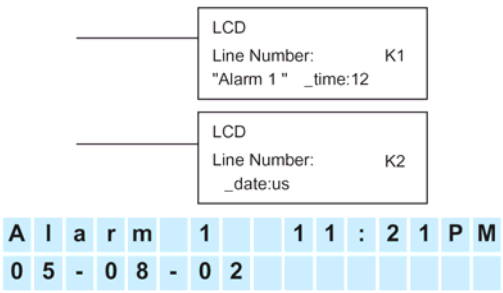
Встраивание значений даты и/или времени

Дата и/или время может быть встроенно в отображаемом тексте, используя переменные, перечисленные в таблице ниже. Эти переменные могут быть включены в сообщение LCD-панели при помощи диалогового окна ввода команды LCD. В примере переменная времени (12-часовой формат) встроенна, при помощи переменной _time:12. Этот формат времени использует максимум семь символов. Второе диалоговое окно создает команду, которая, при выполнении, печатает дату во второй строке дисплея.



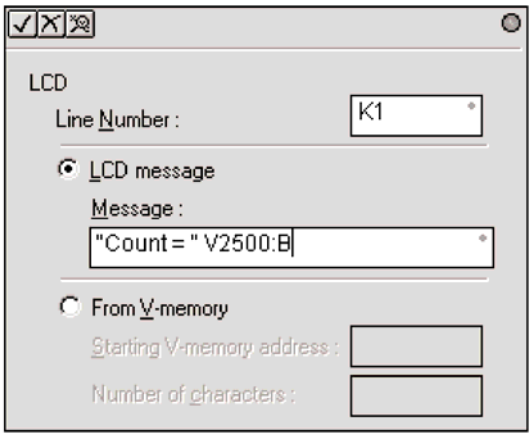
Переменные даты и времени и их форматы		
_date:us	Американский формат	ММ/ДД/ГГ
_date:e	Европейский формат	ДД/ММ/ГГ
_date:a	Азиатский формат	ГГ/ММ/ДД
_time:12	12–часовой формат	ЧЧ:ММ AM/PM
_time:24	24–часовой формат	ЧЧ:ММ:СС

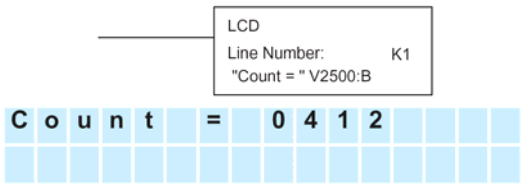




Встраивание значений V-памяти

Любые данные V-памяти могут отображаться в любом из шести доступных форматов данных. Пример показан справа. Список форматов данных и модификаторов находится на следующей странице. Обратите внимание, что различные форматы данных требуют различное число символов на дисплее.





Суффиксы форматов данных для встраивания значений из V- памяти.

Отдельные форматы данных доступны для отображения данных V-памяти на LCD-панели. Варианты показаны в следующей таблице. Двоеточие используется, чтобы отделить встроенную ячейку V-памяти от суффикса формата данных и модификатора. Пример показан на предыдущей странице.

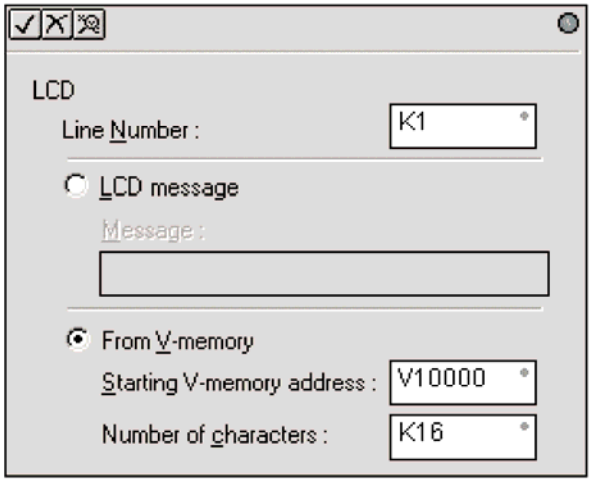
Формат данных	Модифи-катор	Пример	Показываемые символы												
Нет (16-битный формат)		V2000 = 0000 0000 0001 0010	1	2	3	4									
		V2000			1	8									
	[:S]	V2000:S	1	8											
	[:C0]	V2000:C0	0	0	1	8									
	[:0]	V2000:0			1	8									
: B (4-разрядный BCD)		V2000 = 0000 0000 0001 0010	1	2	3	4									
	[:B]	V2000:B	0	0	1	2									
	[:BS]	V2000:BS	1	2											
	[:BC0]	V2000:BC0	0	0	1	2									
	[:B0]	V2000:B0			1	2									
: D (32-битный десятичный)		V2000 = 0000 0000 0000 0000	Двойное слово												
		V2001 = 0000 0000 0000 0001	1	2	3	4	5	6	7	8	9	10	11		
	[:D]	V2000:D							6	5	5	3	6		
	[:DS]	V2000:DS	6	5	5	3	6								
	[:DC0]	V2000:DC0	0	0	0	0	0	0	6	5	5	3	6		
	[:D0]	V2000:D0							6	5	5	3	6		
: DB (8-разрядный BCD)		V2000 = 0000 0000 0000 0000	Двойное слово												
		V2001 = 0000 0000 0000 0011	1	2	3	4	5	6	7	8					
	[:DB]	V2000:DB	0	0	0	3	0	0	0	0					
	[:DBS]	V2000:DBS	3	0	0	0	0								
	[:DBC0]	V2000:DBC0	0	0	0	3	0	0	0	0					
	[:DB0]	V2000:DB0				3	0	0	0	0					
: R (число с плавающей точкой размером в два слова)		V2001/V2000 = 222.11111	Двойное слово												
		(вещественное число)	1	2	3	4	5	6	7	8	9	10	11	12	13
	[:R]	V2000:R				f	2	2	2	.	1	1	1	1	1
	[:RS]	V2000:RS	f	2	2	2	.	1	1	1	1	1			
	[:RC0]	V2000:RC0	f	0	0	0	2	2	2	.	1	1	1	1	1
	[:R0]	V2000:R0				f	2	2	2	.	1	1	1	1	1
: E (число с плавающей точкой размером в два слова в экспоненц. форме)		V2001/V2000 = 222.1	Двойное слово												
		(вещественное число)	1	2	3	4	5	6	7	8	9	10	11	12	13
	[:E]	V2000:E		f	2	.	2	2	1	0	0	E	+	0	2
	[:ES]	V2000:ES	f	2	.	2	2	1	0	0	E	+	0	2	
	[:EC0]	V2000:EC0	f	2	.	2	2	1	0	0	E	+	0	2	
	[:E0]	V2000:E0	f	2	.	2	2	1	0	0	E	+	0	2	
f = флаг плюс / минус (плюс = нет символа, минус = -)															

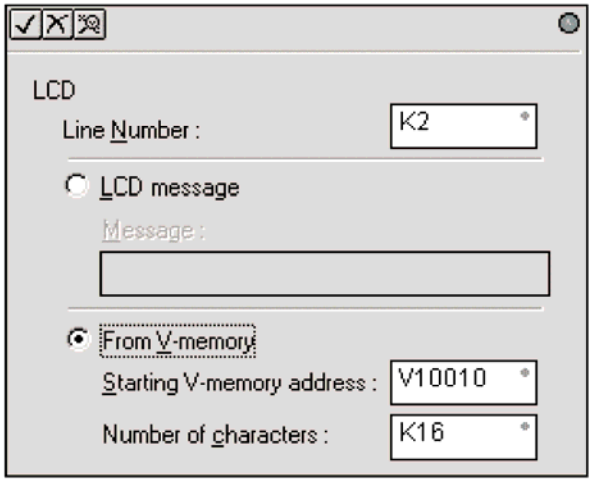
Модификаторы S, C0, и 0 изменяют представление вывода нулей и пробелов. S удаляет символы пробела, и выравнивает результат по левой границе. C0 заменяет, символы пробела нулями. 0 - модификатор C0. 0 заменяет любые нули в варианте формата C0 и преобразовывает их в пробелы.

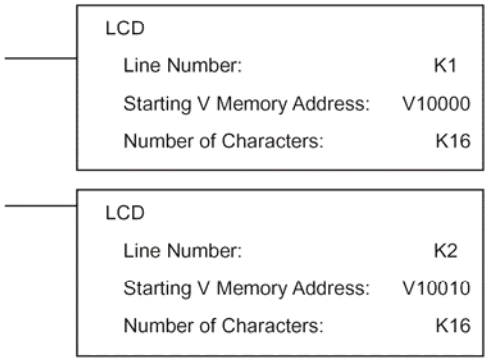
Ввод текста из V-памяти

В качестве альтернативы, текст, который постоянно находится в V-памяти, может отображаться на LCD-панели как показано в примере, приведенном ниже на этой страницы. Диалог ввода команды LCD используется дважды, один раз для каждой строки на дисплее. Диалог требует адрес первого символа и число символов, которое нужно отобразить.

Например, два диалоговых окна, показанные на этой странице создали бы две команды LCD, показанные ниже. Когда эти команды выполняются, то ASCII-символы в ячейках V10000 - V10020 будут отображаться на панели. ASCII-символы и их соответствующие ячейки памяти показаны в таблице ниже.







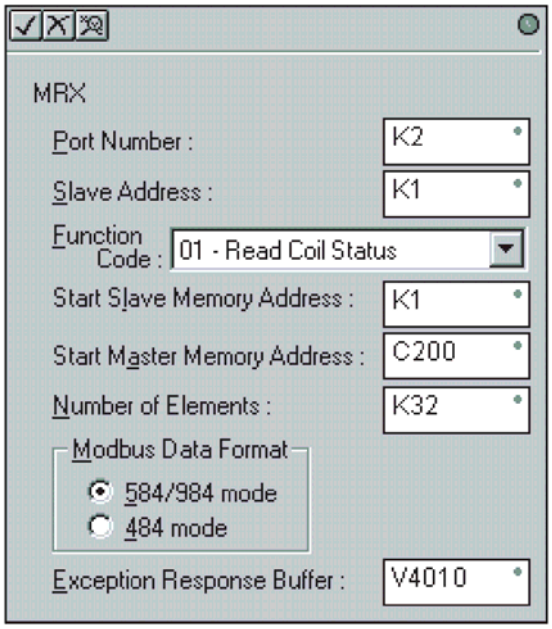
V10000	d	A
V10001	i	m
V10002		n
V10003	f	O
V10004	i	f
V10005	e	c
V10006		
V10007		
V10010	i	H
V10011	h	g
V10012	T	
V10013	m	e
V10014		p
V10015	l	A
V10016	r	a
V10017		m

A	d	m	i	n		O	f	f	i	c	e				
H	i	g	h		T	e	m	p		A	l	a	r	m	

Команды MODBUS RTU

MODBUS Read from Network (MRX)

Команда чтения сети MODBUS (MRX) используется ведущим сети (мастером) DL06, чтобы прочитать блок данных из подсоединенного ведомого устройства и записать данные в адреса V-памяти ведущего. Команда допускает определение пользователем, Кода Функции MODBUS, адреса ведомой станции, начальные адреса ведущего и ведомого, число пересылаемых элементов, формат данных MODBUS и буфер ошибок (Exception Response Buffer).



• **Port Number** (Номер порта): должен быть для DL06 Port 2 - (K2)
• **Slave Address** (Адрес ведомого): Определяет адрес ведомой станции (0–247)
• **Function Code** (Код функции): Следующие коды функции MODBUS поддерживаются командой MRX:

01 – Чтение группы реле
02 – Чтение группы входов
03 – Чтение регистров хранения (holding registers)
04 – Чтение входных регистров (input registers)
07 – Чтение Регистров состояния (Exception status)

• **Start Slave Memory Address:** определяет начальный адрес памяти ведомого устройства, откуда должны быть считаны данные. См. таблицу на следующей странице.
• **Start Master Memory Address:** определяет начальный адрес памяти ведущего устройства, куда должны быть записаны считанные данные. См. таблицу на следующей странице.
• **Number of Elements**: определяет число реле, входов, регистров хранения или входных регистров, которые должны быть считаны. См. таблицу на следующей странице.
• **MODBUS Data Format:** определяет используемый формат MODBUS 584/984 или 484
• **Exception Response Buffer:** определяет начальный адрес памяти ведущего устройства, куда должны быть записаны данные об ошибках (Exception Response). См. таблицу на следующей странице.

Диапазоны адресов ведомого команды MRX.

Код функции	Формат данных MODBUS	Диапазон адресов ведомого
01 – Read Coil	Режим 484	1–999
01 – Read Coil	Режим 584/984	1–65535
02 – Read Input Status	Режим 484	1001–1999
02 – Read Input Status	Режим 584/984	10001–19999 (5 цифр) или 100001–165535 (6 цифр)
03 – Read Holding Register	Режим 484	4001–4999
03 – Read Holding Register	Режим 584/984	40001–49999 (5 цифр) или 4000001–465535 (6 цифр)
04 – Read Input Register	Режим 484	3001–3999
04 – Read Input Register	Режим 584/984	30001–39999 (5 цифр) или 3000001–365535 (6 цифр)
07 – Read Exception Status	Режим 484 и 584/984	Нет доступа

Диапазоны адресов ведущего команды MRX

Тип данных операнда		Диапазон DL06
Входы	X	0–777
Выходы	Y	0–777
Управляющие реле	C	0–1777
Биты стадий	S	0–1777
Биты таймеров	T	0–377
Биты счетчиков	CT	0–177
Специальные реле	SP	0–777
V–память	V	Вся
Удаленные входы	GX	0–3777
Удаленные выходы	GY	0–3777

Число элементов

Тип данных операнда		Диапазон DL06
V–память	V	Вся
Константы	K	Биты: 1-2000, Регистры: 1-125

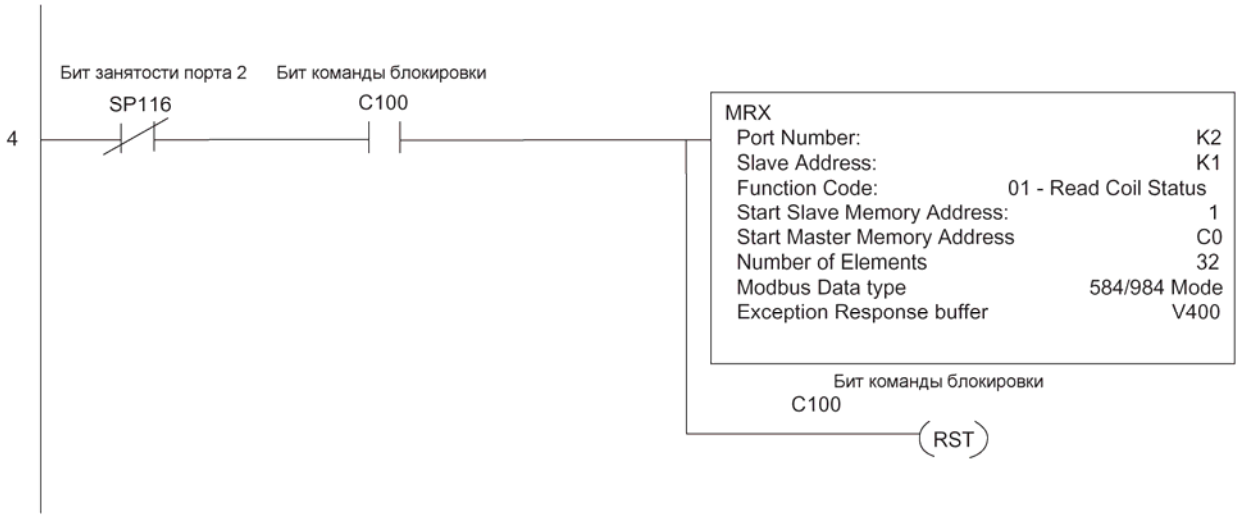
Буфер ошибок

Тип данных операнда		Диапазон DL06
V–память	V	Вся

Пример MRX

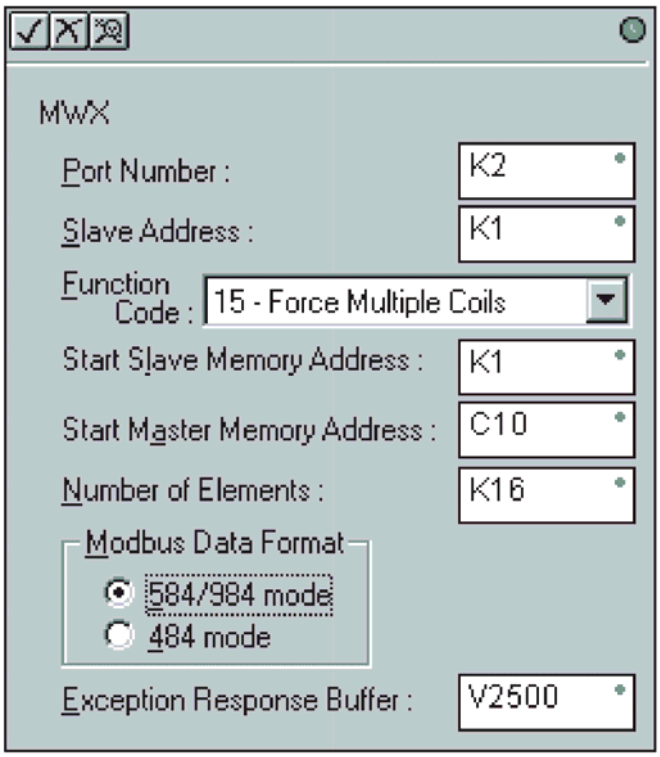
Порт 2 контроллера DL06 имеет два, связанных с ним, специальных реле (см. Приложение D. Специальные реле для порта связи). Одно (SP116) указывает, что “Порт занят”, а другое (SP117) - ”Ошибка Связи Порта”. Бит “Порт занят” включен, в то время когда PLC связывается с ведомым. Когда бит сброшен, то программа, может инициализировать следующий сетевой запрос. Бит “Ошибка связи порта” включается, когда PLC обнаружил ошибку. Этот бит используется дополнительно. Он должен использоваться перед любыми сетевыми командами, так как бит ошибки будет сброшен, если команда MRX или MWX выполнены. Обычно, связь в сети выполняется дольше, чем 1 сканирование процессора. Программа должна ждать освобождение линии связи, перед началом следующего запроса.

Эта цепь читает первые 32 реле MODBUS из ведомого устройства номер один. Она поместит значения считанных реле в 32 бита ведущего, начиная с C0.



MODBUS Write to Network (MWX)

Команда записи в сети MODBUS (MWX) используется для записи блока данных из памяти ведущего сети (мастера) DL06, в подсоединенное ведомое устройство. Команда допускает определение пользователем, Кода Функции MODBUS, адреса ведомой станции, начальные адреса ведущего и ведомого, число пересылаемых элементов, формат данных MODBUS и буфер ошибок.



• **Port Number** (Номер порта): должен быть DL06 Port 2 (K2)
• **Slave Address** (Адрес ведомого): Определяет адрес ведомой станции (0–247)
• **Function Code** (Код функции): Следующие коды функции MODBUS поддерживаются командой MWX:
  05 – Установка одного реле
  06 – Запись одного регистра
  15 – Установка нескольких реле
  16 – Запись нескольких регистров
• **Start Slave Memory Address:** определяет начальный адрес памяти ведомого устройства, куда должны быть записаны данные.
• **Start Master Memory Address:** определяет начальный адрес памяти ведущего устройства, откуда должны быть считанны записываемые данные.
• **Number of Elements:** определяет число реле или регистров , которые должны быть записаны. Это поле активно только тогда, когда выбран код функции 15 или 16.
• **MODBUS Data Format:** определяет используемый формат MODBUS 584/984 или 484
• **Exception Response Buffer:** определяет начальный адрес памяти ведущего устройства, куда должны быть записаны данные об ошибках (Exception Response).

Диапазоны адресов ведомого команды MWX.

Код функции	Формат данных MODBUS	Диапазон адресов ведомого
05 – Установка одного реле	Режим 484	1–999
05 – Установка одного реле	Режим 584/984	1–65535
06 – Запись одного регистра	Режим 484	4001–4999
06 – Запись одного регистра	Режим 584/984	40001–49999 (5 цифр) или 400001–465535 (6 цифр)
15 – Установка нескольких реле	Режим 484	1–999
15 – Установка нескольких реле	Режим 584/984	1–65535
16 – Запись нескольких регистров	Режим 484	4001–4999
16 – Запись нескольких регистров	Режим 584/984	40001–49999 (5 цифр) или 400001–465535 (6 цифр)

Диапазоны адресов ведущего команды MWX

Тип данных операнда		Диапазон DL06
Входы	X	0–777
Выходы	Y	0–777
Управляющие реле	C	0–1777
Биты стадий	S	0–1777
Биты таймеров	T	0–377
Биты счетчиков	CT	0–177
Специальные реле	SP	0–777
V–память	V	Вся
Удаленные входы	GX	0–3777
Удаленные выходы	GY	0–3777

Число элементов команды MWX

Тип данных операнда		Диапазон DL06
V–память	V	Вся
Константы	K	Биты: 1-2000, Регистры: 1-125

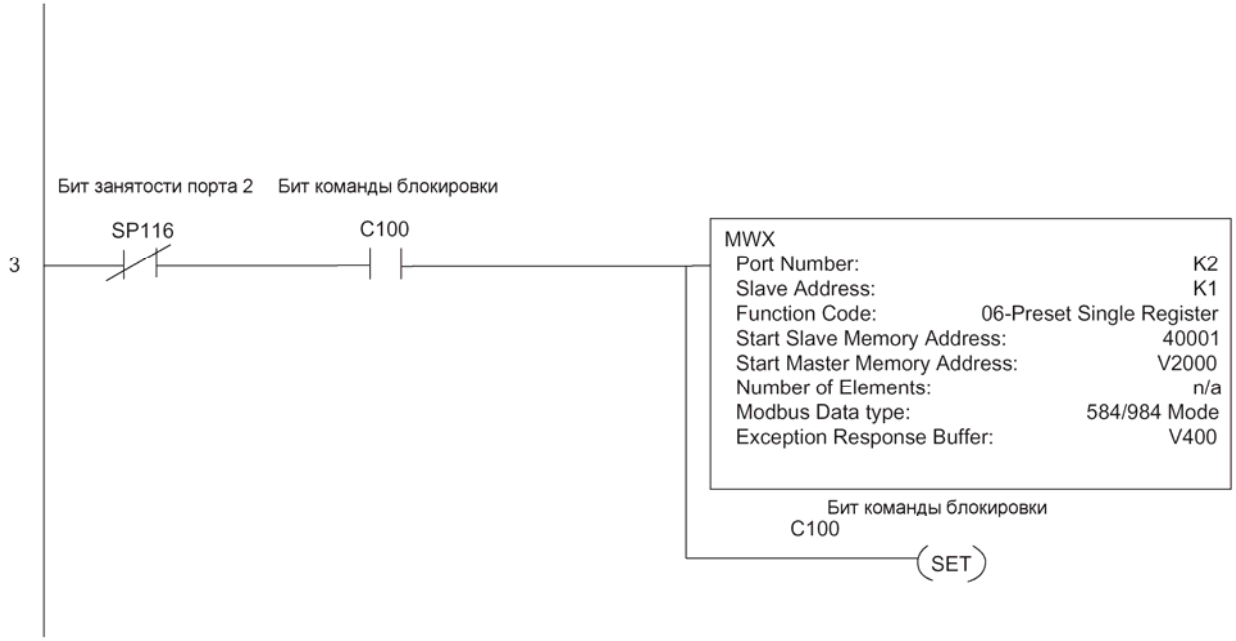
Буфер ошибок команды MWX

Тип данных операнда		Диапазон DL06
V–память	V	Вся

Пример MWX

Порт 2 контроллера DL06 имеет два, связанных с ним, специальных реле, (см. Приложение D. Специальные реле для порта связи). Одно (SP116) указывает, что “Порт занят”, а другое (SP117) указывает ”Ошибка Связи Порта”. Бит “Порт занят” включен, в то время когда PLC связывается с ведомым. Когда бит сброшен, то программа, может инициализировать следующий сетевой запрос. Бит “Ошибка связи порта” включается, когда PLC обнаружил ошибку. Этот бит используется дополнительно. Он должен использоваться перед любыми сетевыми командами, так как бит ошибки будет сброшен, если команда MRX или MWX выполнены. Обычно, связь в сети выполняется дольше, чем 1 сканирование процессора. Программа должна ждать освобождение линии связи, перед началом следующего запроса.

Эта цепь выполняет запись по MODBUS в первый регистр хранения 40001 в ведомом устройстве с адресом один. Записываемое значение хранится в ячейке V2000. Используется код функции 16, Этот код записывает только 1 регистр. Только один запрос по сети (WX, RX, MWX, MRX) возможен в одном цикле сканирования. Эта ситуация является причиной для использования бит взаимной блокировки. Для использования многих запросов в одной сети, рассмотрите использование команды Shift Register.



Команды ASCII

Процессор DL06 поддерживает несколько команд и способов, которые позволяют читать и записывать ASCII-строки в портах связи контроллера. А именно, порт 2 контроллера DL06 может использоваться для чтения или записи необработанных ASCII-строк. DL06 может также декодировать ASCII-символы, встроенные в пределах поддерживаемого протокола (K–Sequence, DirectNet, Modbus) через порт процессора.

Чтение входной ASCII-строки.

Имеется несколько способов, чтобы использовать DL06 для чтения входной ASCII -строки.

1) ASCII IN (AIN) – Эта команда настраивает порт 2 на прием необработанных ASCII-строк с такими параметрами как фиксированная и переменная длина ASCII-строки, символы окончания передачи, байт дополнительного обмена и служебных бит команд. Используется со сканерами штрихового кода, весами, и т.д. для записи необработанных ASCII-строк в порт 2 на основаннии параметров команды (AIN).

2) Запись встроенных ASCII-строк непосредственно в V-память из внешнего устройства операторского интерфейса или подобного ведущего устройства через поддерживаемый протокол связи, используя порты процессора. Команда AIN в этом случае не используется.

3) Если контроллер DL06 является ведущим сети, можно воспользоваться командой Network Read (RX), для чтения встроенных ASCII-данных из подчиненного устройства через поддерживаемый протокол связи, используя порт 2. Команда RX помещает данные непосредственно в V-память.

Запись выходной ASCII-строки

Следующие команды могут использоваться, для записи выходной ASCII-строки:

1) Print from V–memory (PRINTV) – Используйте эту инструкцию, чтобы вывести необработанные ASCII-строки в порт 2 к панели оператора или последовательному принтеру, и т.п. Команда показывает начальный адрес V-памяти, длину строки, байт дополнительного обмена, и т.д. Когда разрешающий бит команды устанавлен, строка записывается в порт 2.

2) Print to V–memory (VPRINT) – Используйте эту команду, для создания закодированных ASCII-строк в контроллере (т.е. сигнализирующих сообщений). Когда разрешающий бит команды устанавливается, сообщение загружается в определенный адрес ячейки V-памяти. Затем команда (PRINTV) может использоваться, чтобы записать закодированную ASCII-строку в порт 2. Поддерживаются американские, европейские и азиатские форматы времени / даты.

Дополнительно, если контроллер DL06 является ведущим сети, то может использоваться команда Network Write (WX), чтобы записать встроенные данные ASCII в устройство человеко-машинного интерфейса или подчиненное устройство непосредственно из V-памяти через поддерживаемый протокол связи, используя порт 2.

Управление ASCII-строками

Следующие команды могут быть полезны в работе с ASCII-строками в пределах V-памяти процессора:

• ASCII Find (AFIND) - Находит отдельный блок ASCII-строки в последовательных адресах V-памяти. Поддерживается поиск вперед и назад.

• ASCII Extract (AEX) - Извлекает отдельный блок (обычно некоторое значение данных) из найденной или другой известной ячейки данных ASCII.

• Compare V–memory (CMPV) - Эта команда используется, чтобы сравнить два блока адресов V-памяти и обычно используется, чтобы обнаружить изменения в ASCII-строке. Сравниваемые типы данных должны иметь один и тот же формат (т.е. BCD, ASCII,и т.д.).

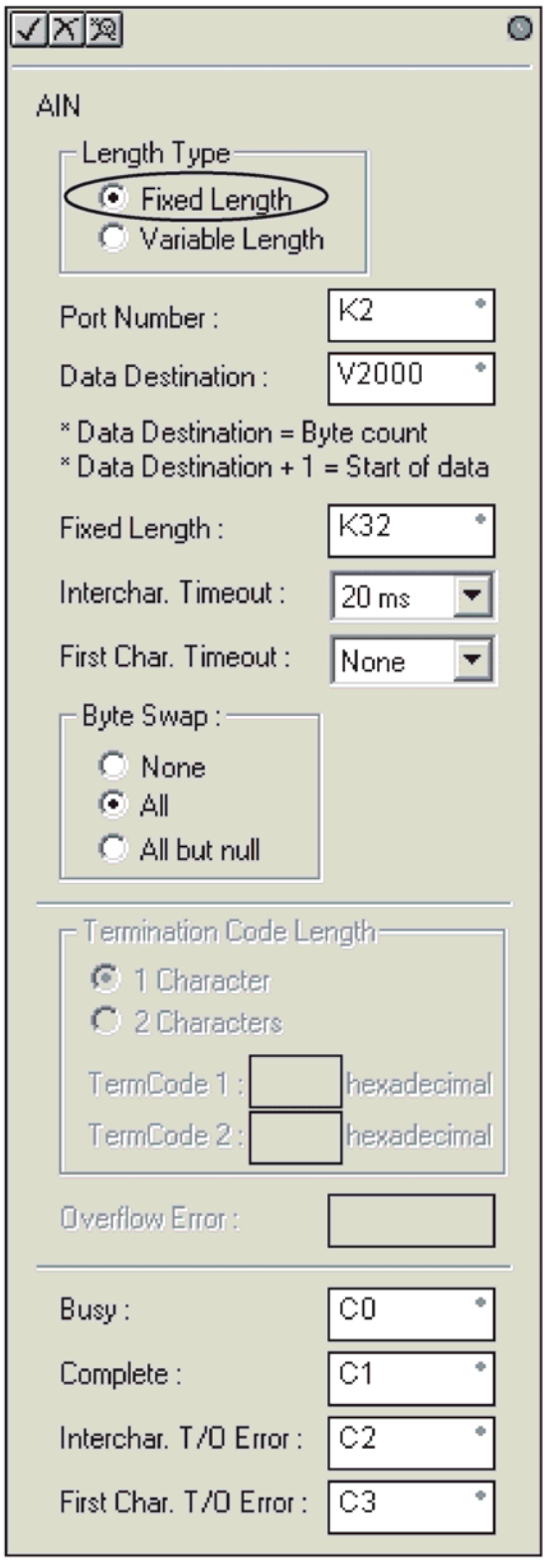
• Swap Bytes (SWAPB) – Обычно используется, чтобы обменять байты V-памяти на данные ASCII, которые были записаны непосредственно в V-память из внешнего устройства человеко-машинного интерфейса или подобного управляющего устройства через протокол связи. Команды AIN и AEX имеют встроенную возможность перестановки байта.

ASCII Input (AIN)

Команда ASCII Input разрешает процессору получить ASCII-строки через указанный порт связи и помещает строку в указанную последовательность регистров V-памяти. Данные ASCII могут быть получены как фиксированное число байт или как строка переменной длины с определенным символом окончания. Другие возможности включают, перестановка байтов «Byte Swap», настройка межсимвольного таймаута, и определяемые пользователем биты флагов для «порт занят», «окончания сообщения» и ошибки по таймауту.

Конфигурация команды AIN с фиксированной длиной

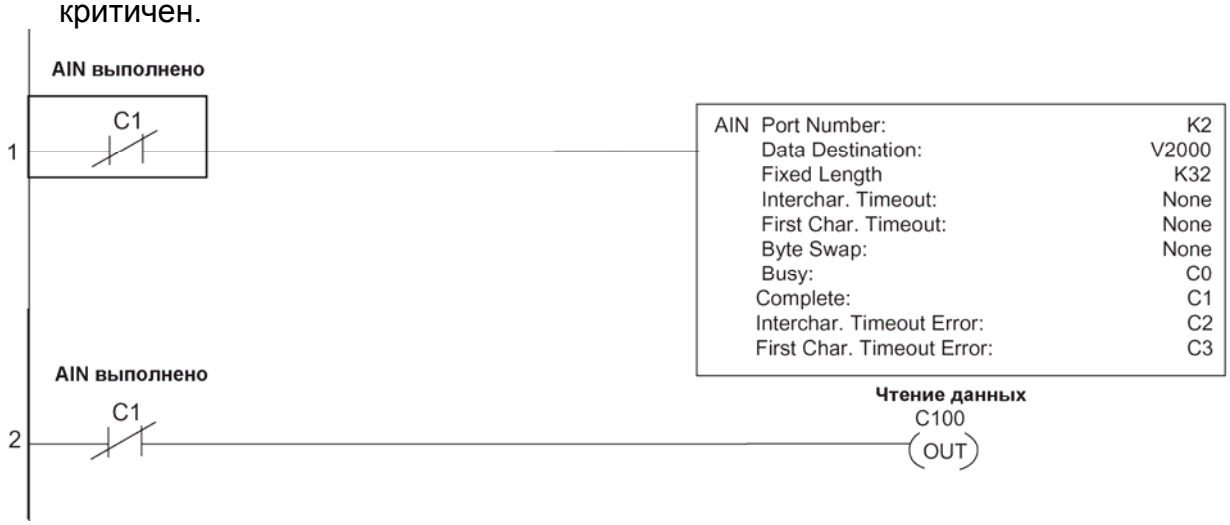
- **Length Type:** Выберите фиксированную длину, основанную на постоянной длине ASCII-строки, которая будет посылаться порту процессора.
- **Port Number:** должен быть DL06 Порт 2 (K2)
- **Data Destination:** Указывает, куда в V-памяти будет помещена строка ASCII
- **Fixed Length:** Определяет фиксированную длину, в байтах, ASCII-строки.
- **Inter–character Timeout:** Если количество времени между ASCII-символами превышает установленное время, то бит ошибка по таймауту (Timeout Error bit) будет установлен. Данные не будут сохранены в ячейке V-памяти адреса назначения. Бит сбрасывается, когда отключается разрешающий бит для выполнения команды AIN. Выбор 0мс отключает эту настройку.
- **First Character Timeout:** Если время с момента запуска команды AIN до получения первого символа, превышает указанное, то бит таймаута первого символа (First Character Timeout bit) будет установлен. Бит сбрасывается, когда отключается разрешающий бит для выполнения команды AIN. Выбор 0мс отключает эту настройку.
- **Byte Swap:** Меняет местами старший и младший байт в пределах каждого регистра V-памяти ASCII-строки фиксированной длины. См. команду SWAPB.
- **Busy Bit:** включен, когда команда AIN продолжает получать ASCII-данные
- **Complete Bit:** Установлен, когда ASCII данные указанной длины были получены и сбрасывается, когда отключается разрешающий бит для выполнения команды AIN.
- **Inter–character Timeout Error Bit:** Установлен, когда межсимвольный таймаут превышен.



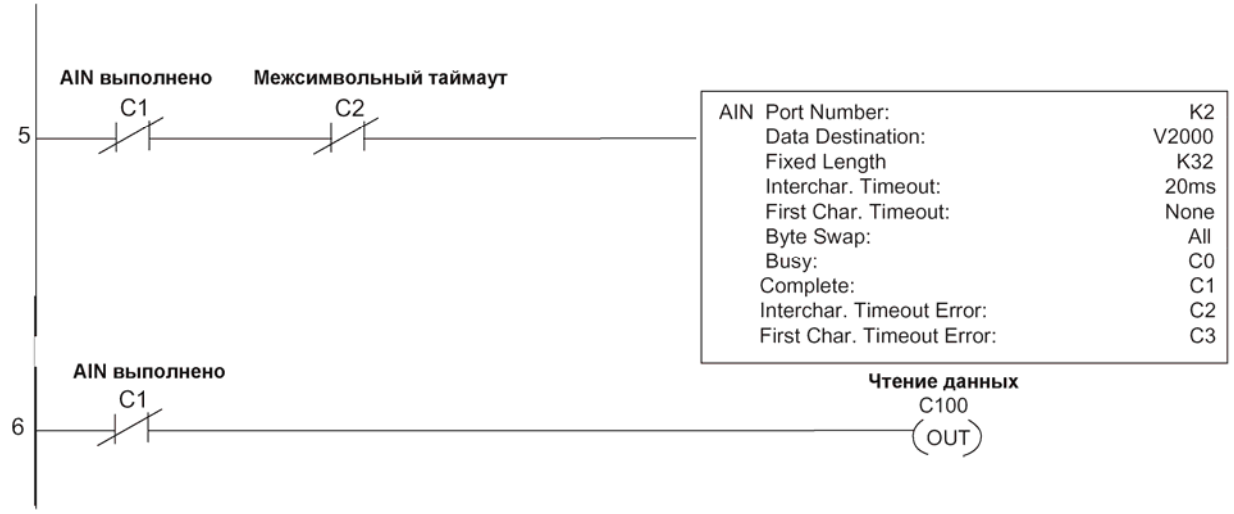
Параметры	
Размещение данных (Data Destination)	Вся V-память
Длина (Fixed Length)	K1–128
Биты: занят, выполнен, ошибка таймаута, переполнение	C0–1777

Пример команды AIN с фиксированной длиной

Пример чтения сообщения фиксированный длины, когда контроллер опрашивает порт непрерывно и выбор определенного момента времени не критичен.

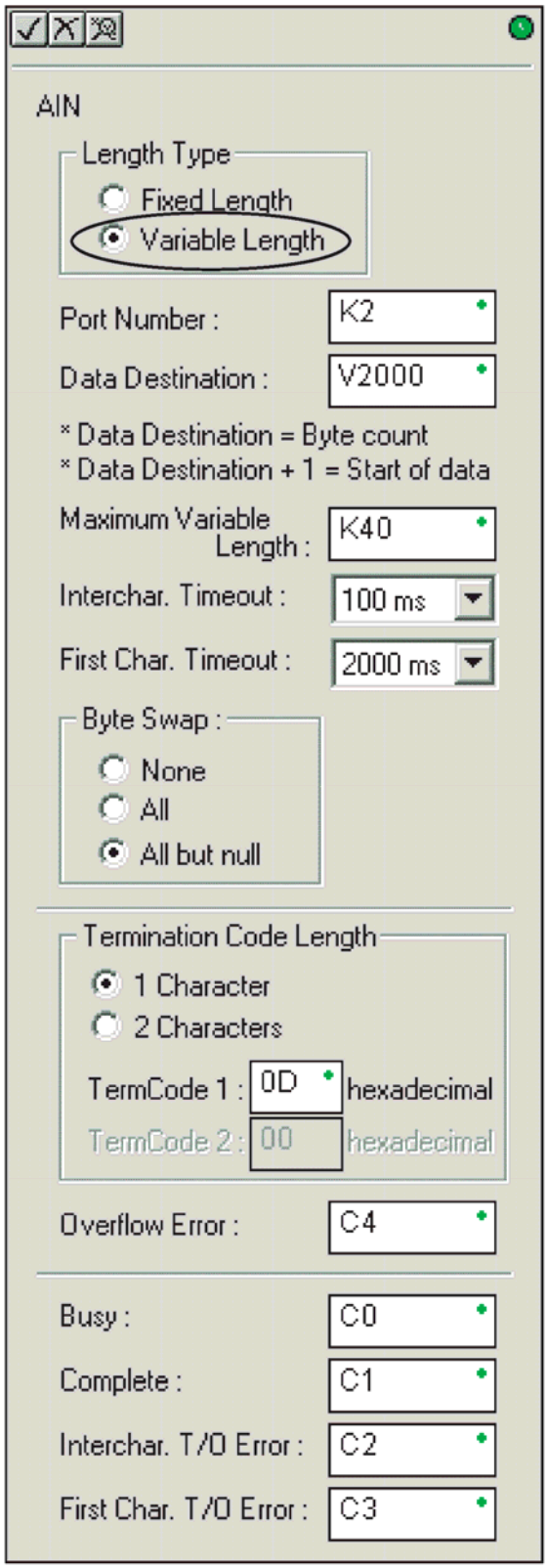


Пример чтения сообщения фиксированный длины, когда время между символами критично.



Конфигурация команды AIN с переменной длиной

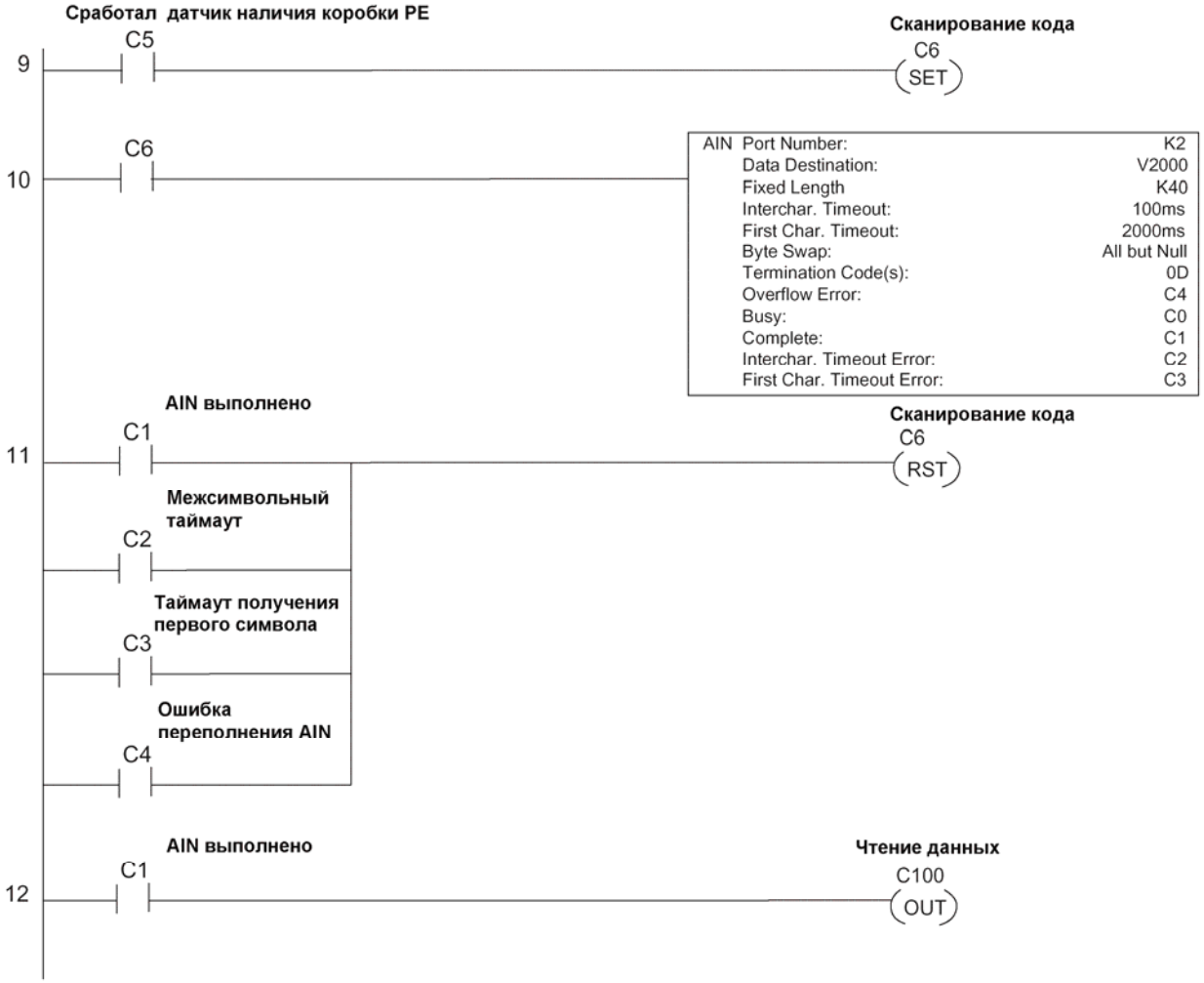
- **Length Type:** Выберите переменную длину сообщения, основанную на символе окончания ASCII-строки.
- **Port Number:** должен быть для DL06 Порт 2 - (K2)
- **Data Destination:** Указывает, куда в V-памяти будет помещена строка ASCII
- **Maximum Variable Length:** определяет максимальную длину ASCII-строки, в байтах, которую будет принимать порт.
- **Inter–character Timeout:** Если количество времени между ASCII-символами превышает установленное время, то бит ошибка по таймауту (Timeout Error bit) будет установлен. Данные не будут сохранены в ячейке V-памяти адреса назначения. Бит сбрасывается, когда отключается разрешающий бит для выполнения команды AIN. Выбор 0мс отключает эту настройку.
- **First Character Timeout:** Если время с момента запуска команды AIN до получения первого символа, превышает указанное, то бит таймаута первого символа (First Character Timeout bit) будет установлен. Бит сбрасывается, когда отключается разрешающий бит для выполнения команды AIN. Выбор 0мс отключает эту настройку.
- **Byte Swap:** Меняет местами старший и младший байт в пределах каждого регистра V-памяти ASCII-строки переменной длины. См. команду SWAPB.
- **Termination Code Length:** состоит из 1 или из 2 символов. Смотрите таблицу ASCII в приложении G
- **Busy Bit:** включен, когда команда AIN продолжает получать ASCII-данные
- **Complete Bit:** Установлен, когда ASCII-данные были получены вместе с символом окончания строки и сбрасывается, когда отключается разрешающий бит для выполнения команды AIN.
- **Inter–character Timeout Error Bit:** Установлен, когда межсимвольный таймаут превышен.
- **First Character Timeout Error Bit:** устанавливается когда таймаут первого символа превышен. О таймауте первого символа смотрите выше.
- **Overflow Error Bit**: устанавливается когда ASCII-данный превышают параметр Maximum Variable Length.



Параметры	
Размещение данных (Data Destination)	Вся V-память
Длина	K1–128
Биты: занят, выполнен, ошибка таймаута, переполнение	C0–1777

Пример команды AIN с переменной длиной

Пример чтения командой AIN сообщения переменной длины, используется для чтения штрих-кодов на коробках (PE = фотодатчик).

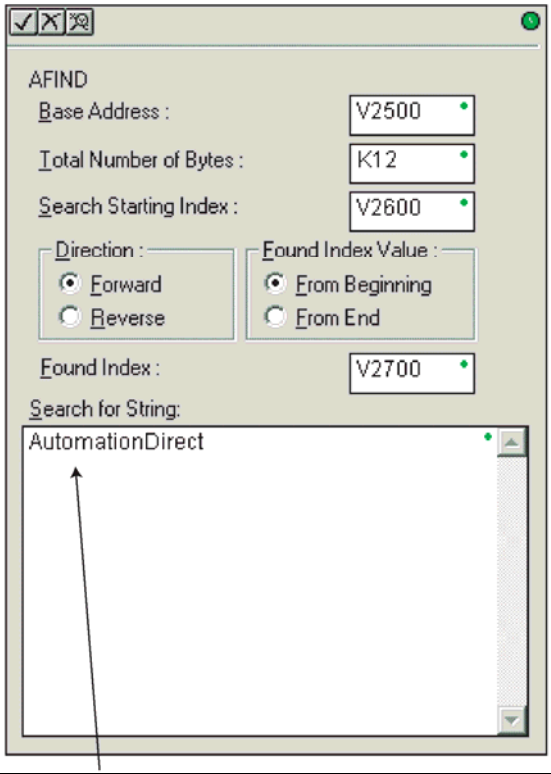


ASCII Find (AFIND)

Команда ASCII Find обнаруживает отдельную строку ASCII или часть строки ASCII в пределах диапазона V-памяти, регистрирует и размещает номер строки (Found Index) (номер байта, где требуемая строка найдена), в шестнадцатиричном формате, в определенный регистр V-памяти. Дополнительные параметры команды включают в себя: Search Starting Index (начальный индекс поиска) для пропуска лишних байт перед началом работы FIND, направление поиска Forward (вперед) или Reverse (назад), и варианты выбора ссылки From Beginning (от начала) и From End (от конца) для заначение Found Index.

- Base Address: определяет начало регистра V-памяти, где осуществляется поиск ASCII-строки.
- Total Number of Bytes: определяет общее число байт, в которых производится поиск требуемой строки ASCII
- Search Starting Index: определяет сколько байт пропускать (относительно Base Address) перед началом поиска
- Direction: Forward начинает поиск от младших регистров V-памяти к старшим. Reverse производит поиск от старших регистров V-памяти к младшим.
- Found Index Value: устанавливает Beginning (начальный) или End (конечный) байт найденной строки ASCII будет загружен в регистр Found Index
- Found Index: определяет регистр V-памяти, где значение Found Index будет сохранено. Если требуемая строка не обнаружена в указанных регистрах памяти, результатом будет являтся значение в FFFF.
- Search for String: искомый текст до 128 символов

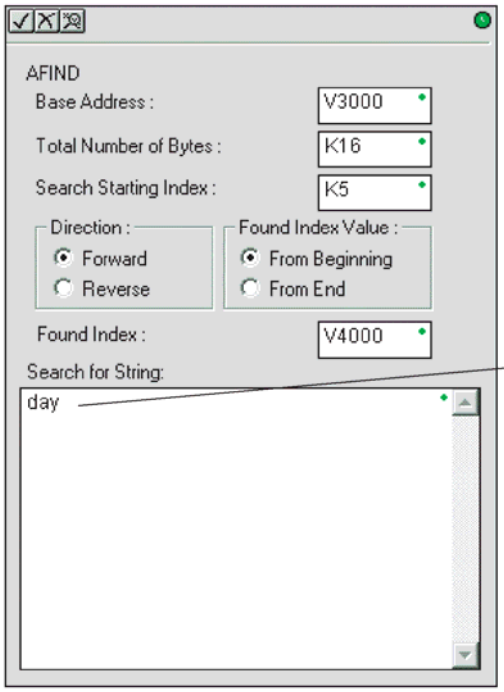
Параметр	Диапазон DL06
Base Address	Вся V–память
Total Number of Bytes	Вся V– память или K1–128
Search Starting Index	Вся V– память или K0–127
Found index	Вся V– память



ПРИМЕЧАНИЕ. Кавычки вокруг элемента Search String не требуются. Кавычки - допустимые символы, которые AFIND может искать.

Пример поиска AFIND

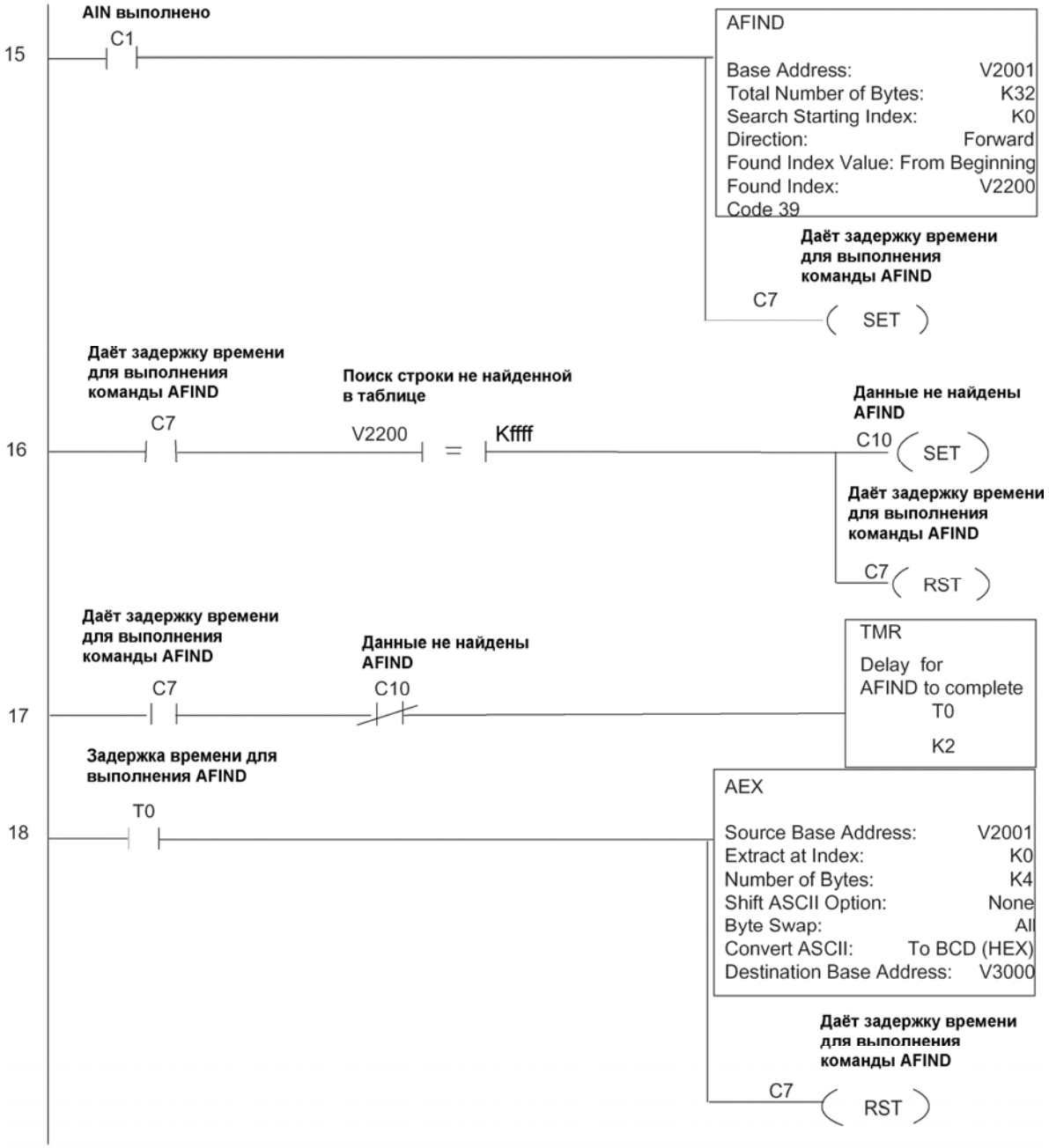
В следующем примере, команда AFIND используется для поиска части “day” слова “Friday” в ASCII-строке “Today is Friday.”, которая предварительно была загружена в V-память. Обратите внимание, использование начального индекса поиска Search Starting Index, являющегося константой (K)5, вместе с направлением поиска Direction Search – Forward(вперед) необходимо для того, чтобы предотвратить нахождение части “day” в слове “Today”. Результат поиска Found Index будет помещен в V4000.



Внимание, искомая строка Search String не заключается в кавычки. Используйте кавычки только тогда, когда они являются частью искомой строки Search String.

Пример объединения AFIND с командой AEX

Когда команда AIN выполнилась, её бит завершенния может быть использован для вызова команды AFIND для поиска требуемуй части ASCII-строки. Когда строка найдена, команда AEX может использоваться для выделения сохраненной строки.



ASCII Extract (AEX)

Команда ASCII Extract извлекает определенное число байт ASCII-данных из одной серии регистров V-памяти и помещает их в другую серию регистров V-памяти. Дополнительные параметры команды включают в себя: Extract at Index - для пропуска ненужных байт перед началом работы Extract, опцию Shift ASCII - для смещения на один байт влево или один байт вправо, Byte Swap – для изменения порядка следования байт и Convert – для преобразования данных в число BCD-формата.

• Source Base Address: определяет начало регистра V-памяти, где в памяти сохранена вся строка ASCII.

• Extract at Index: определяет какой байт пропускать (относительно Source Base Address) перед извлечением данных

• Number of Bytes: определяет число байт, которые будут извлечены

• Опция Shift ASCII: смещает все извлеченные данные на один байт влево или один байт вправо, замещая “нежелательные” символы по необходимости

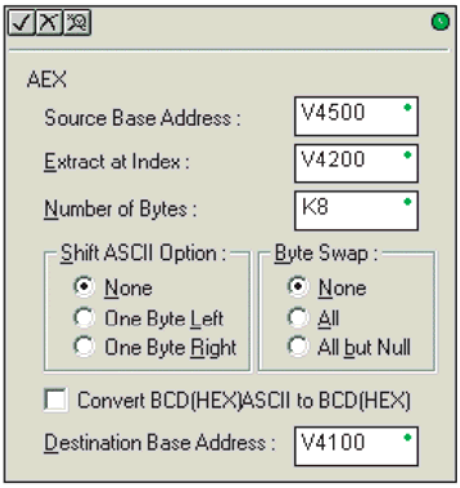
• Byte Swap: переставляет в извлеченных данных старший и младший байт в пределах каждого регистра V -памяти. Для уточнения смотри команду SWAPB.

• Convert BCD(Hex) ASCII to BCD (Hex): если данная опция разрешена, то преобразует численные символы ASCII в шестнадцатеричные числа.

• Destination Base Address: определяет регистр V-памяти, где будут сохранены извлеченные данные

Смотри предыдущую страницу для примера использования команды AEX.

Параметр	Диапазон DL06	
Source Base Address	Вся V–память	
Extract at Index	Вся V– память или K0–127	
Number of Bytes (“Convert BCD (HEX) ASCII” **Не** включено)	Диапазон константы K1-128	Ячейка V– памяти содержащая BCD число 1-128
Number of Bytes (“Convert BCD (HEX) ASCII” включено)	Диапазон константы K1-4	Ячейка V– памяти содержащая BCD число 1-4
Destination Base Address	Вся V–память	



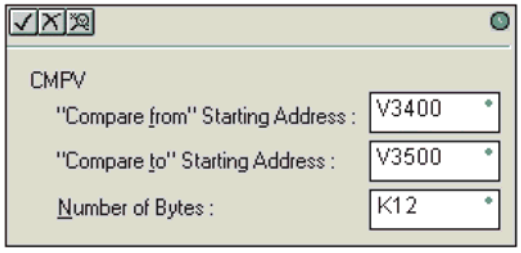
ASCII Compare (CMPV)

Команда ASCII Compare сравнивает две группы регистров V–памяти. CMPV сравнивает любые типы данных (ASCII с ASCII, BCD с BCD, и т.д.) одной серии (группы) регистров V– памяти с другой для указанной в байтах длины.

• “Compare from” Starting Address: указывает начальный регистр V–памяти первой из сравниваемых групп.

• “Compare to” Starting Address: указывает начальный регистр V–памяти второй из сравниваемых групп..

• Number of Bytes: указывает длину каждой из указанных групп регистров V–памяти.

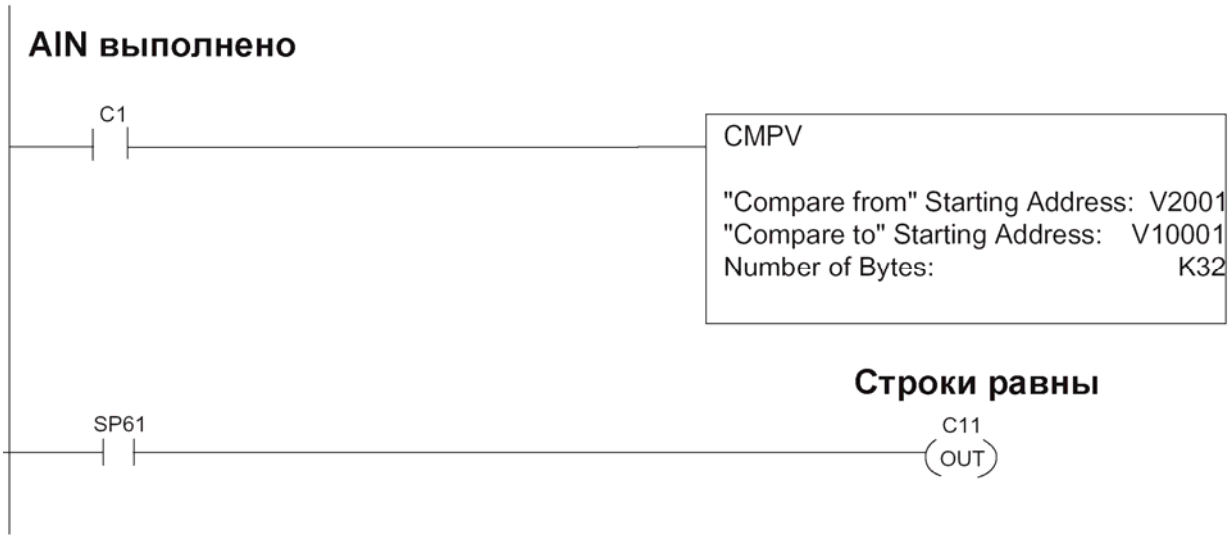


SP61 = 1, результат - равны
SP61 = 0, результат – не равны

Параметр	Диапазон DL06
Compare from Starting Address	Вся V–память
Compare to Starting Address	Вся V– память
Number of Bytes	K0–127

Пример CMPV

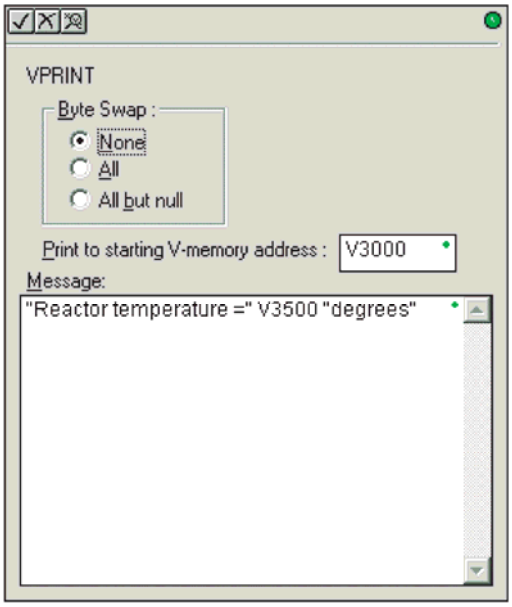
Команда CMPV выполняется после выполнения команды AIN. Если сравниваемые таблицы V-памяти равны, то SP61 будет установлен.



ASCII Print to V–memory (VPRINT)

Команда ASCII Print to V–memory записывает указанную ASCII-строку в последовательные регистры V–памяти.
Дополнительными параметрами команды являются варианты Перестановки байт в слове - Byte Swap: отсутствие; полная перестановка, перестановка за исключением нулей или пробелов; и различные форматы даты (американский, европейский, азиатский) и времени (12-ти или 24-х часовой).

• Byte Swap: меняет местами старший и младший байт в пределах каждого регистра V-памяти напечатанной строки ASCII. Для уточнения смотри команду SWAPB.
• Print to Starting V–memory Address: определяет начало последовательных адресов памяти, где командой VPRINT будет размещена строка ASCII.•



ПРИМЕЧАНИЕ. Starting V–memory Address: первый регистр V-памяти, указанных регистров, будет содержать длину строки ASCII в байтах. Starting V–memory Address +1: 2-ые и последующие регистры будут содержать строку ASCII, напечатанную в V-памяти.

Параметр	Диапазон DL06
Starting V–memory Address	Вся V–память

Тип времени/даты – коды в таблице ниже могут использоваться в VPRINT сообщении ASCII-строки для «печати в V-памяти» текущего времени и/или даты

#	Код знака	Тип времени/даты
1	_date:us	Американский формат (месяц/день/2 цифры года)
2	_date:e	Европейский формат (день /месяц/ 2 цифры года)
3	_date:a	Азиатский формат (2 цифры года/ месяц/день)
4	_time:12	12-часовой формат (0-12 часы : минуты AM/PM)
5	_time:24	24-часовой формат (0-23 часы : минуты AM/PM)

VPRINT V-memory element – Следующие модификаторы могут использоваться использоваться в VPRINT сообщении ASCII-строки для «печати в V-памяти» регистра в целочисленном или вещественном формате. Используйте номер регистра V-памяти с разделителем «:» и модификатором типа данных. Модификатор типа данных показан в таблице ниже. Символьный код должен быть заглавной буквой.

ПРИМЕЧАНИЕ. До и после номера регистра V-памяти обязательно должен находится пробел, чтобы отделить его от текстовой строки. В случае если это требование не выполнено возникает ошибка с кодом 499.

#	Символьный код	Описание
1	нет	16-битное двоичное (десятичное число)
2	: B	4–х разрядное BCD
3	: D	32- битное двоичное (десятичное число)
4	: D B	8-ми разрядное BCD
5	: R	Число с плавающей запятой (вещественное число)
6	: E	Число с плавающей запятой (вещественное число в экспоненциальном виде)

Примеры:
V2000 напечатаются двоичные данные из V2000 в виде десятичного числа
V2000 : B напечатаются BCD данные из V2000
V2000 : D напечатаются двоичные данные из V2000 и V2001 в виде десятичного числа
V2000 : D B напечатаются BCD данные из V2000 и V2001
V2000 : R напечатается число с плаваящей запятой из V2000/V2001 как вещественное число
V2000 : E напечатается число с плаваящей запятой из V2000/V2001 как вещественное число в экспоненциальном виде

Следующие модификаторы могут добавляться к любому из вышеперечисленных, чтобы удалять или преобразовывать начальные нули или пробелы. Символлный код должен быть заглавной буквой.

#	**Символьный код**	**Описание**
1	S	Удаляет начальные пробелы
2	C0	Преобразует начальные пробелы в нули
3	0	Удаляет начальные нули

Например V2000 = 0018 (двоичный формат)

Регистр V-памяти с модификатором	Число символов			
	1	2	3	4
V2000	0	0	1	8
V2000:B	0	0	1	2
V2000:B0	1	2		

Например V2000 = пробел пробел 18 (двоичный формат)

Регистр V-памяти с модификатором	Число символов			
	1	2	3	4
V2000	пробел	пробел	1	8
V2000:B	пробел	пробел	1	2
V2000:BS	1	2		
V2000:BC0	0	0	1	2

Текстовый элемент V-памяти VPRINT – следующий элемент используется для «печати в V-память» текста сохраненного в регистре. Используйте символ % и число символов после номера V-памяти для отображения текста. Если Вы используете “0”, как число символов, то то функция считает количество символов из первой указанной ячейки, а текст начнет извлекать из следующей ячеки памяти.

Например:
V2000 % 16 напечатает 16 символов с V2000 по V2007.
V2000 % 0 напечатает символы с V2001 по Vxxxx (определяется числом символов в ячейке V2000)

Битовый элемент V-памяти VPRINT – следующий элемент используется для «печати в V-память» состояния указанных бит в V-памяти или биты управляющих реле. Битовый элемент может быть определен номером ячейки V-памяти, затем точкой (.) и следующим за ним номера бита или номером управляющего реле. Тип сообщений выводимых на печать приведен в таблице ниже.

№	Формат вывода	Описание
1	нет	печатает 1 при включенном состоянии и 0 при отключенном
2	: BOOL	печатает «TRUE» при включенном состоянии и «FALSE» при отключенном
3	: ONOFF	печатает «ON» при включенном состоянии и «OFF» при отключенном

Например:
V2000 . 15 Печатает состояние бита 15 ячейки V2000, в формате 1/0
C100 Печатает состояние бита C100, в формате 1/0
C100 : BOOL Печатает состояние бита C100, в формате TRUE/FALSE
C100 : ON/OFF Печатает состояние бита C100, в формате ON/OFF
V2000.15 : BOOL Печатает состояние бита 15 ячейки V2000, в формате TRUE/FALSE

Максимальное число символов, которое Вы можете напечатать командой VPRINT равно 128. Число символов, требуемых для каждого элемента, независимо от того, используются ли модификаторы :S,:C0 или:0, перечислены в таблице ниже.

Тип элемента	Максимальное число символов
Текст, 1 символ	1
16-битное двоичное	6
4–х разрядное BCD	11
32- битное двоичное	4
8-ми разрядное BCD	8
Вещественное число	3
Вещественное число в экспоненциальном виде	13
Текст V-памяти	2
Бит (формат 1/0)	1
Бит (формат TRUE/FALSE)	5
Бит (формат ON/OFF)	3

Текстовый элемент – следующий элемент используется для «печати в V-память» символьной строки. Символьные строки являются последовательностью символов (больше чем 0), располагающимися между двойными кавычками. Два шестнадцатиричных числа, которым предшествует знак доллара представляют собой 8-разрядный ASCII-код символа. Также, два символа, которые начинаются со знака доллара интерпретируются согласно следующей таблице:

№	Символьный код	Описание
1	$$	Знак доллара ($)
2	$”	Двойные кавычки (”)
3	$Lor $l	Перевод строки (LF)
4	$N or $n	Возврат каретки с переводом строки(CRLF)
5	$P or $p	Перевод страницы
6	$R or $r	Возврат каретки (CR)
7	$T or $t	Табуляция

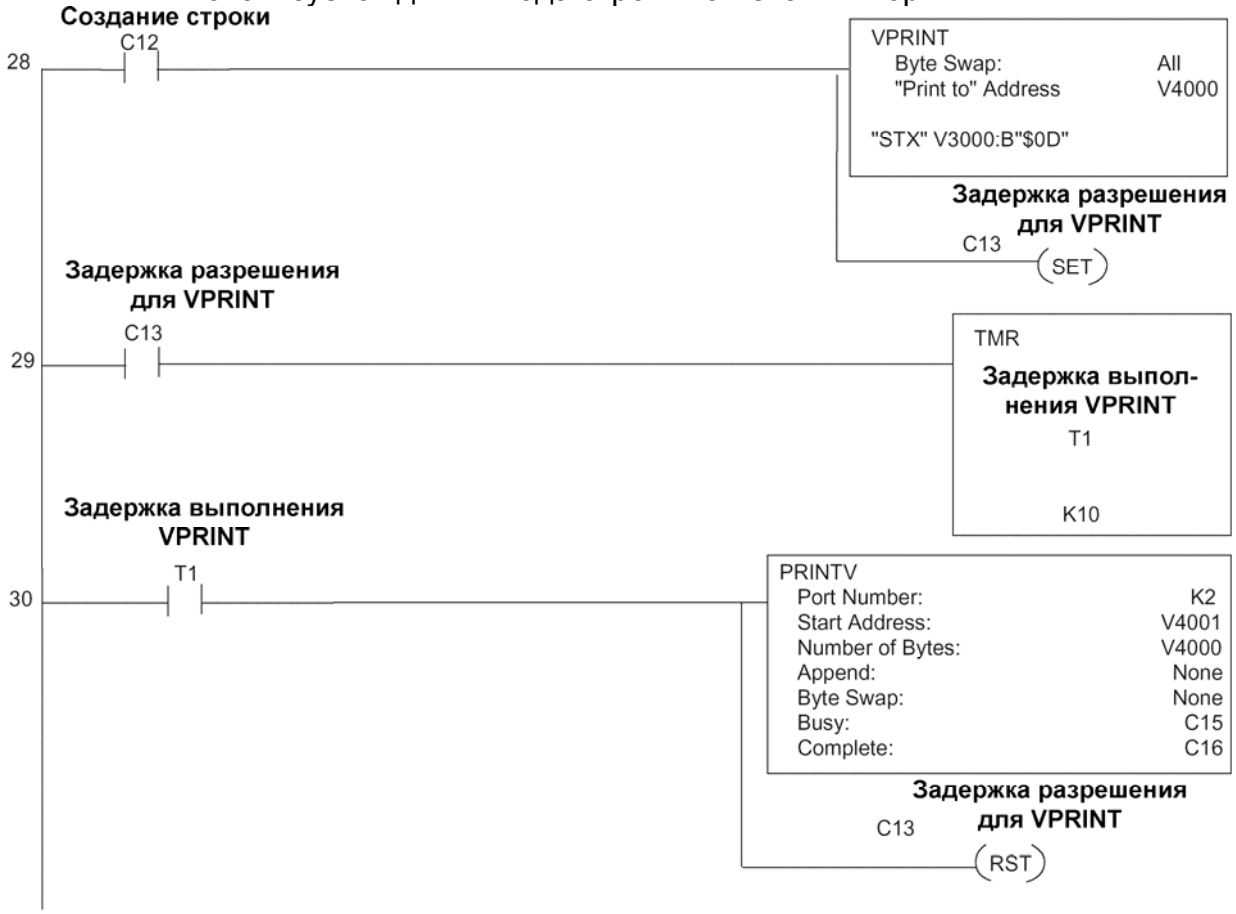
Следующие примеры показывают различные синтаксические соглашения и длину вывода на печать.

” ”	Длина 0 без символа
”A”	Длина 1 с символом A
” ”	Длина 1 с пробелом
” $” ”	Длина 1 с двойной кавычкой
” $ 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 знаком $

Печатая обычную строку текста, Вам надо включить двойные кавычки до и после текстовой строки. Код ошибки 499 будет возникать в процессоре, когда инструкция печати содержит недопустимый текст или отсутствуют кавычки. Важно проверить данные для команды VPRINT при разработке приложения.

Пример комбинации команд VPRINT и PRINTV

Команда VPRINT используется для создания строки в V-памяти. Команда PRINTV используется для вывода строки на печать в порт 2.



ASCII Print from V–memory (PRINTV)

Команда ASCII Print from V–memory используется для вывода ASCII-строки на указанный порта связи из определенной последовательности регистров V-памяти с определенной длиной в байтах. Другие возможности команды включают в себя определяемые пользователем дополнительные символы(Append Characters), которые будут помещены после требуемой строки данных для устройств, которые требуют специального символа (ов) окончания, а также вожможности Перестановки Байтов(Byte Swap), и определяемые пользователем флаги занятости и окончания выполнения.

• Port Number: Должен быть для DL06 порт 2 (K2)

• Start Address: Определяет начало последовательности регистров V-памяти, которые содержат ASCII-строку для печати.

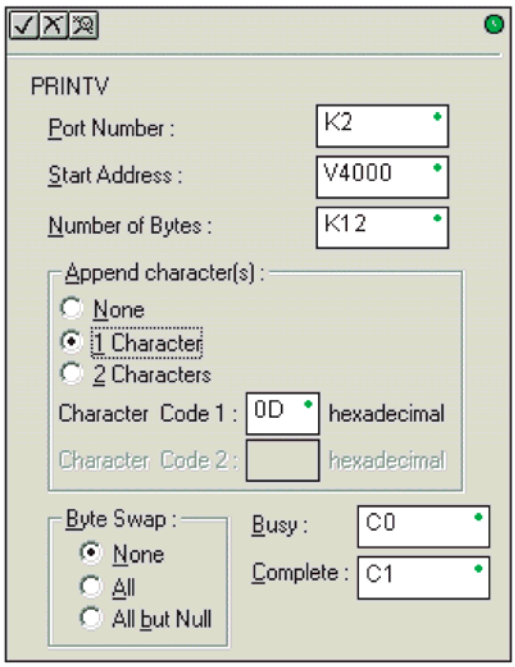
• Number of Bytes: Определяет длину строки для печати

• Append Characters: Определяет ASCII-символы, которые нужно добавить к концу строки для устройств, которые требуют специальных символов окончания передачи

• Byte Swap: обмен местами сташего байта с младшим в каждом регистре V-памяти при печати строки. Смотри команду SWAPB для дополнительных сведений.

• Busy Bit: будет включен, когда команда продолжает печатать ASCII-данные.

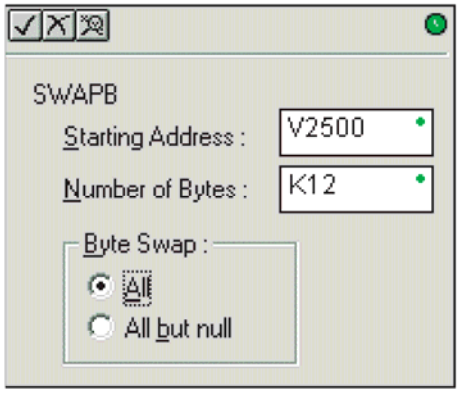
• Complete Bit: будет установлен только после того, как все ASCII-данные будут распечатаны и сбрасывается, когда разрешающий бит для команды PRINTV отключает ветвь команды.



Смотри предыдущую страницу с примером, использующим команду PRINTV.

Параметр	Диапазон DL06
Номер порта	Порт 2 (K2)
Начальный адрес	Вся V-память
Число байт	Вся V-память или K1-128
Биты занятости и окончания печати	C0-1777

ASCII Swap Bytes (SWAPB)



Команда ASCII Swap Bytes обменивает местами позиции байт (старший байт в младший и, наоборот, младший байт в старший) в каждом регистре V-памяти в указанной последовательности регистров V-памяти.

• Starting Address: Определяет начальный адрес последовательных регистров V-памяти, для обмена байт.

• Number of Bytes: Определяет число байт, начиная со стартового адреса для обмена байт.

• Byte Swap:

All – обмен всех байт

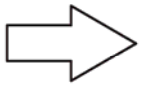
All but null - обмен всех байт, кроме неполных

Параметр	Диапазон DL06
Начальный адрес	Вся V-память
Число байт	Вся V-память или K1-128

Варианты обмена байт

Нет обмена байт

(AIN, AEX, PRINTV, VPRINT)

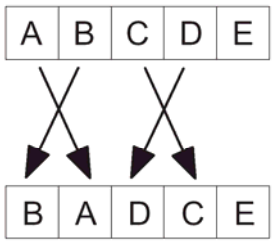


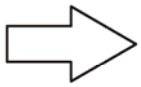
Байт

Старший Младший

V2000	0005h	
V2001	B	A
V2002	D	C
V2003	xx	E

Перестановка всех байт



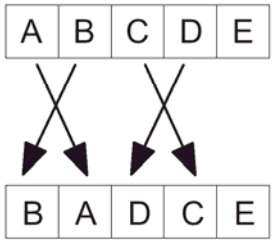


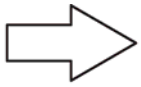
Байт

Старший Младший

V2000	0005h	
V2001	A	B
V2002	C	D
V2003	E	xx

Обмен всех байт, кроме неполных





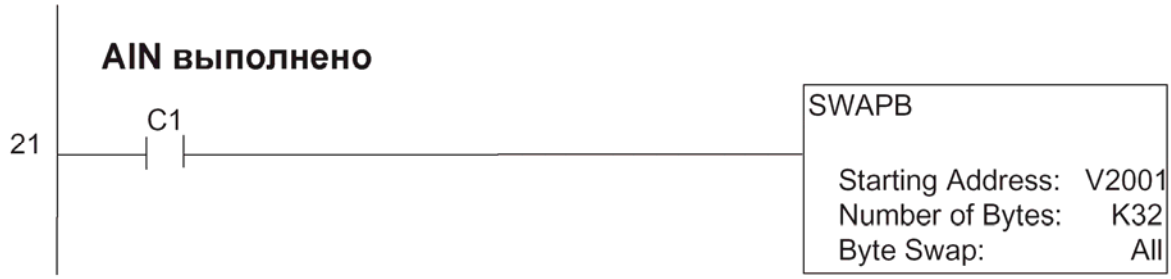
Байт

Старший Младший

V2000	0005h	
V2001	A	B
V2002	C	D
V2003	xx	E

Пример SWAPB

Бит окончания выполнения команды AIN используется, чтобы вызвать команду SWAPB. Используйте команду однократного импульса, так чтобы команда SWAPB выполнялась только один раз.



ASCII Clear Buffer (ACRB)

Команда ASCII Clear Buffer очищает премный ASCII буфер на указанном коммуникационном порту. Port Number: должен быть для DL06 порт 2 - (K2)

Пример ACRB

Бит окончания выполнения команды AIN используется, чтобы вызвать команду очистки ASCII буфера.