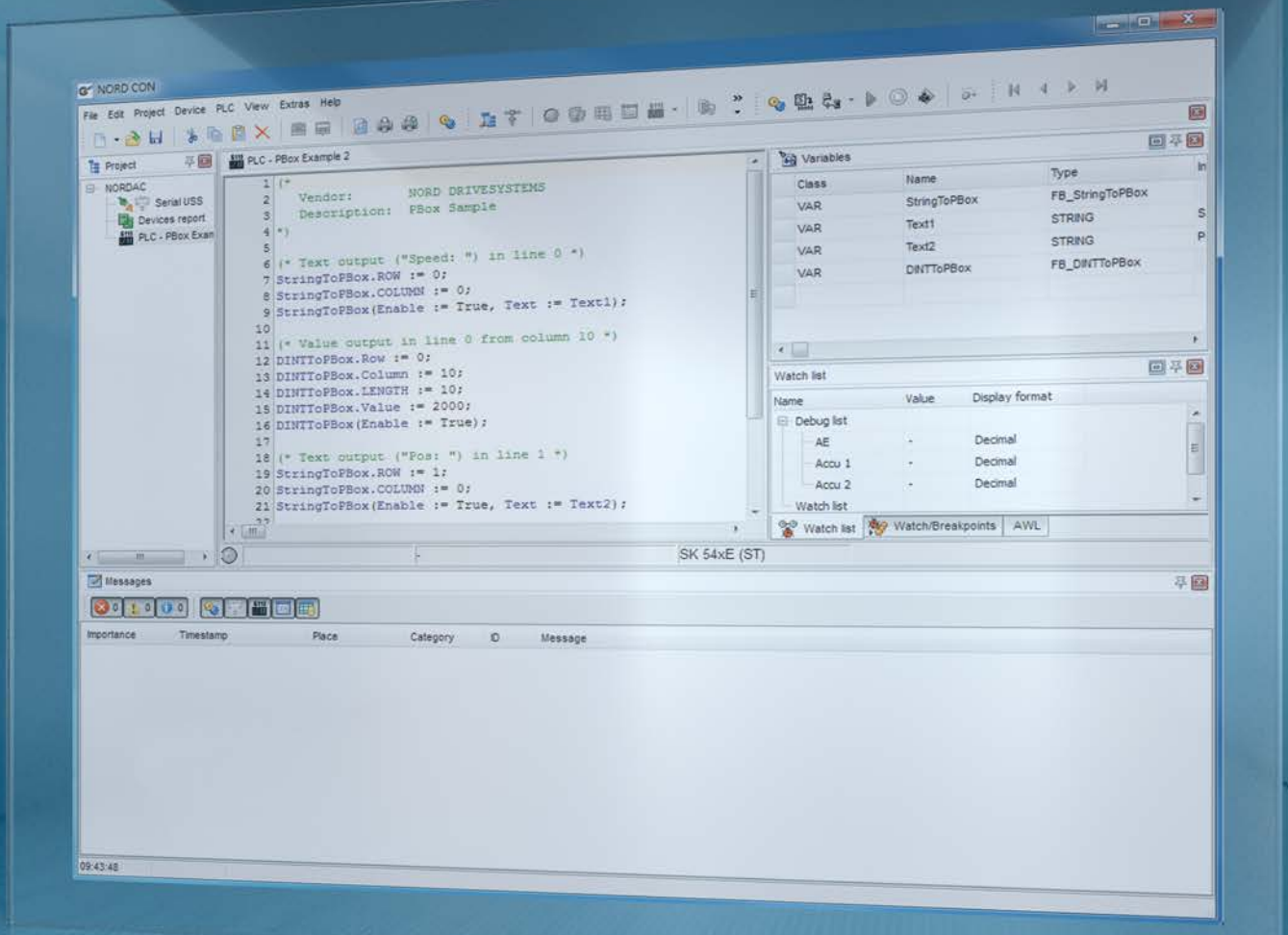
INTELLIGENT DRIVESYSTEMS, WORLDWIDE SERVICES



BU 0550 – ru

Функции ПЛК

Дополнительное руководство для устройств NORDAC

NORD® DRIVESYSTEMS

Оглавление

1 Введение

1.1 Общая информация

1.1.1 Документация

Наименование: **BU 0550**

Артикул: **6075507**

Серия: **Функциональность ПЛК частотного преобразователя и пускового устройства двигателя для оборудования серий**

NORDAC *PRO* (SK 500P … SK 550P)
(SK 520E … SK 545E)

NORDAC *FLEX* (SK 200E … SK 235E)

NORDAC *Base* (SK 180E / SK 190E)

NORDAC *Link* (SK 250E-FDS … SK 280E-FDS)

NORDAC *Link* (SK 155E-FDS / SK 175E-FDS)

1.1.2 Изменения документа

Редакция **Номер заказа**	**Модельный ряд**	**Версия** **Программное обеспечение**	**Примечания**
BU 0550, Сентябрь 2011 года **6075507**/ 3911	SK 540E … SK 545E	V 2.0 R0	Первое издание
Другие редакции: октябрь 2011, февраль 2013, февраль 2017 Краткий перечень изменений, внесенных в вышеуказанные издания, приводится в отдельном документе			
BU 0550, май 2019 года	SK 500P … SK 550P	V 1.0 R1	• Добавлено описание устройств типов NORDAC *PRO* SK 500P … SK 550P • Изменения и исправления
	SK 540E … SK 545E	V 2.4 R0	
6075507/ 1919	SK 520E … SK 535E	V 3.2 R0	
	SK 200E … SK 235E	V 2.2 R0	
	SK 180E / SK 190E	V 1.3 R0	
	SK 250E-FDS … SK 280E-FDS	V 1.3 R0	
	SK 155E-FDS / SK 175E-FDS	V 1.2 R0	

1.1.3 Авторское право

1.1.4 Издатель

Getriebebau NORD GmbH & Co. KG

Getriebebau-Nord-Straße 1

22941 Bargteheide, Germany

http://www.nord.com/

Тел. +49 (0) 45 32 / 289-0

Факс +49 (0) 45 32 / 289-2253

1.1.5 Об этом руководстве

Это руководство содержит информацию по вводу в эксплуатацию функционала ПЛК частотного преобразователя и пускового устройства двигателя Getriebebau NORD GmbH & Co. KG (коротко NORD). Оно предназначено для специалистов-электротехников, выполняющих работы по планированию, проектированию, настройке и внедрению программ для ПЛК (📖 раздел 2.1 "Квалификация персонала"). При этом предполагается, что специалисты-электротехники, отвечающие за выполнение этих задач, знакомы с особенностями электронной приводной техники и, в частности, с оборудованием NORD.

В настоящем руководстве содержится информация и описание функциональных возможностей ПЛК, а также дополнительная информация о приборах Getriebebau NORD GmbH & Co. KG, связанная с использованием функционала ПЛК.

1.2 Применяемая документация

Это руководство следует использовать только вместе с инструкцией по эксплуатации, прилагаемой к соответствующему преобразователя частоты, так как в ней содержится вся информация, необходимая для безопасного ввода в эксплуатацию преобразователя и надежной работы приводной установки. Список соответствующих документов приводится в 📖 главе 5.2 "Документы и программы".

Все необходимые документы можно также найти на сайте www.nord.com.

1.3 Условные обозначения

1.3.1 Указания

В документе указания, относящиеся к безопасности оператора или использованию шинных интерфейсов, отмечены следующим образом:

⚠ ОПАСНО

Это указание сообщает о прямой опасности, угрожающей жизни и здоровью персонала.

⚠ ПРЕДУПРЕЖДЕНИЕ

Это указание сообщает об опасности, которая может угрожать жизни и здоровью персонала.

⚠ ОСТОРОЖНО

Это указание на незначительную опасность,которая может привести к травмам легкой или средней степени тяжести.

ВНИМАНИЕ

Указание на возможное повреждение оборудования.

1.3.2 Другие указания

ⓘ Информация

Указание на важную или полезную информацию.

1.4 Использование по назначению

ПЛК -производства Getriebebau NORD GmbH & Co. KG является устройством, дополняющим программные и аппаратные функции частотного преобразователя и пускового устройства двигателя NORD. Он подключается к соответствующему прибору и не может работать отдельно от него. В связи с этим необходимо в полной мере соблюдать указания по технике безопасности, указанные в руководстве, прилагаемом к соответствующему прибору (📖 раздел 5.2 "Документы и программы"Dokumente und Software</dg_ref_source_inline).

Функциональный модуль -ПЛК применяется, как правило, в сложных приводных системах, с одним или несколькими электронными приводными приборами, а также для облегчения выполнения приводных функций управления и контроля за счет использования прибора с соответствующим оснащением.

2 Безопасность

2.1 Квалификация персонала

Работы по вводу в эксплуатацию функционального модуля ПЛК -разрешается выполнять только квалифицированным специалистам-электрикам, которые обладают необходимыми знаниями о функциональном модуле ПЛК, электронной приводной технике и средствах конфигурирования (например, NORD CON) и в достаточной степени знакомы с периферийным оборудованием, связанным с приводной установкой (датчики, контроллеры).

Эти лица, кроме того, должны уметь выполнять ввод в эксплуатацию электронной приводной техники и периферийных устройств, связанных с ней.

2.1.1 Квалифицированный персонал

Квалифицированным персоналом называются лица, которые благодаря своему специальному образованию и профессиональному опыту обладают специализированными знаниями и которые хорошо знают действующие стандарты по технике безопасности и охране труда, а также общие правила по работе с соответствующим оборудованием.

Эти лица могут выполнять работы на установке только с разрешения владельца установки.

2.1.2 Специалист-электрик

Квалифицированным электриком считается специалист, который благодаря своему профессиональному образованию и опыту обладает знаниями, позволяющими

- выполнять включение, выключение, изолирование, заземление и маркировку электрических цепей и устройств,
- выполнять работы по техническому обслуживанию и использовать защитные устройства в соответствии с установленными нормами безопасности.
- обеспечивать аварийное электроснабжение

2.2 Инструкции по технике безопасности

Технологический модуль **Функции ПЛК** и оборудование производства Getriebebau NORD GmbH & Co. KG разрешается использовать только для целей, для которых они предназначены, 🕮 раздел 1.4 "Использование по назначению".

Во избежание опасных ситуаций при использовании технологического модуля выполнять все требования и условия, перечисленные в настоящем руководстве.

Разрешается эксплуатировать устройство, если его конструкция не изменена и на устройстве установлены все защитные панели и крышки. Убедиться в отсутствии повреждений и исправном состоянии соединений и кабелей.

К работам на устройстве и к эксплуатации устройства допускается только квалифицированный персонал, 🕮 глава 2.1 "Квалификация персонала".

3 ПЛК

3.1 Общая информация

Частотный преобразователь NORD серий SK 180E/SK 190E, SK 2xxE, SK 2xxE-FDS, SK 520E – SK 545E и SK 5xxP, а также пусковое устройство двигателя серии SK 155E-FDS/SK 175E-FDS имеют встроенную логическую систему обработки данных реализуемую на базе стандарта IEC61131-3, действующего для программируемых логических контроллеров (ПЛК/PLC). Вычислительная мощность и время отклика системы такого ПЛК позволяет ему выполнять простые вспомогательные задачи периферийного оборудования преобразователя. В частности, он может осуществлять контроль информации, поступающей через входы преобразователя или по полевой шине, анализировать ее и передавать на преобразователь уже обработанные расчетные значения. Совместно с другими приборами NORD он также позволяет оператору получать в графическом виде информацию о состояниях установки и вводить специальные параметры. Таким образом, обеспечивается экономия за счет отказа от использования внешних модулей ПЛК. Устройство поддерживает язык программирования AWL. AWL представляет собой машинный текстовый язык программирования, содержание и применение которого описываются стандартом IEC61131-3.

Информация

Программирование и установка на прибор осуществляются исключительно при помощи программного обеспечения NORD NORD CON.

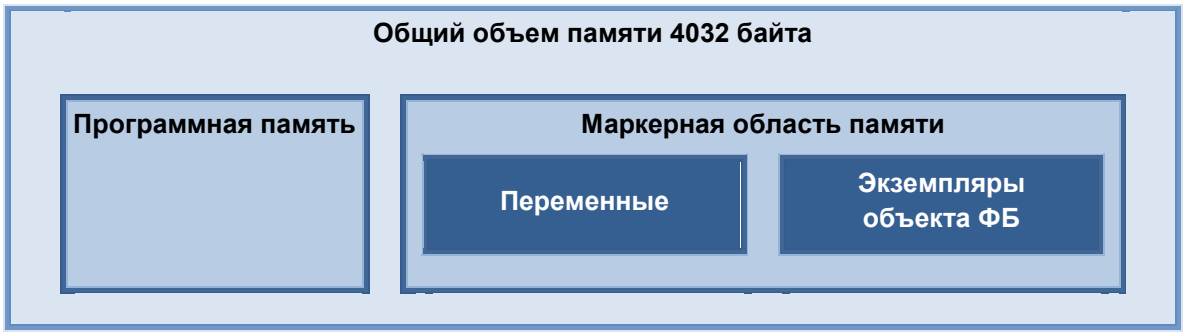
3.1.1 Спецификация ПЛК

Функция	Спецификация		
Стандартная конфигурация	На базе IEC61131-3		
Язык	Instruction List (IL), strukturierter Text (ST)		
Задание	Циклическая задача, вызов программы каждые 5 мс		
Вычислительная мощность	Около 200 к.команд AWL за 1 мс		
Запоминающее устройство	SK 5xxP, SK 520E … SK 545E, SK 2xxE, SK 2x0E-FDS	SK 190E / SK 180E	SK 155E-FDS / SK 175E-FDS
	8128 байт для маркеров, функций и программы ПЛК	2032 байт для маркеров, функций и программы ПЛК	2028 байт для маркеров, функций и программы ПЛК
Максимальное количество команд	ок. 2580 команд	ок. 660 команд	ок. 660 команд
	Примечание: Данное значение является средним, использование большого числа маркеров, данных процессов и функций значительно уменьшает возможное количество строк, см. раздел "Ресурсы".		
Доступные буферы сообщений CAN Mailboxen	20		
Поддерживаемые приборы	SK 5xxP SK 54xE SK 53xE / SK 52xE от V3.0 SK 2xxE от V2.0 SK 2x0E-FDS SK 180E / SK 190E SK 155E-FDS / SK 175E- FDS		

3.1.2 Конструкция ПЛК

3.1.2.1 Запоминающее устройство

Запоминающее устройство ПЛК подразделяется на программную и идентификационную (маркерную) части. В маркерной части, помимо переменных, хранятся экземпляры объектов функциональных блоков (ФБ). Экземпляр объекта представляет собой область памяти, в которой хранятся все внутренние входные и выходные переменные функционального блока. Для описания каждого функционального блока требуется собственный экземпляр объекта. Граница между программной и маркерной областями памяти является динамической и зависит от размера маркерной области.



В маркерной памяти в области переменных сохраняются два различных класса:

[VAR]

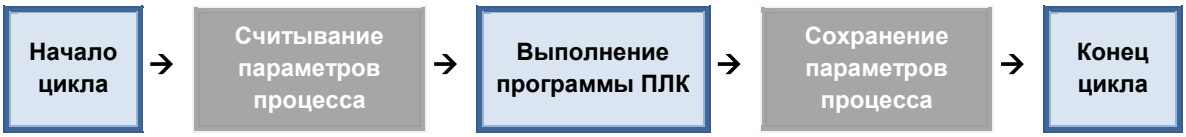
Переменные памяти для записи вспомогательной информации и состояний. Переменные этого типа заново устанавливаются при каждом запуске ПЛК. Содержимое памяти сохраняется во время выполнения цикла ПЛК.

[VAR_ACCESS]

Служит для считывания и описания данных процессов (входы, выходы, уставки и т.д.) частотного преобразователя. Эти значения заново создаются при каждом цикле ПЛК.

3.1.2.2 Образ процесса

Работа прибора характеризуется множеством физических параметров, таких как вращающий момент, частота вращения, положение, входы, выходы и т.д. Эти параметры делятся на уставки и фактические значения. Образ процесса позволяет загружать и обрабатывать эти параметры. Требуемые параметры процессов определяются в списке переменных класса VAR_ACCESS. Все заданные в списке переменных данные процессов преобразователя заново считываются при выполнении каждого цикла ПЛК. В конце каждого цикла ПЛК записываемые данные процессов снова передаются на преобразователь, см. рисунок далее.



Такой порядок выполнения определяет необходимость использования циклов при создании программы. Программирование циклов на ожидание определенного события (например, изменения уровня на входе) не дает желаемого результата. Функциональные блоки, которые

обращаются к параметрам процессов, ведут себя иначе. Здесь параметры процессов считываются при вызове функционального блока и записываются сразу по окончании блока.

Информация

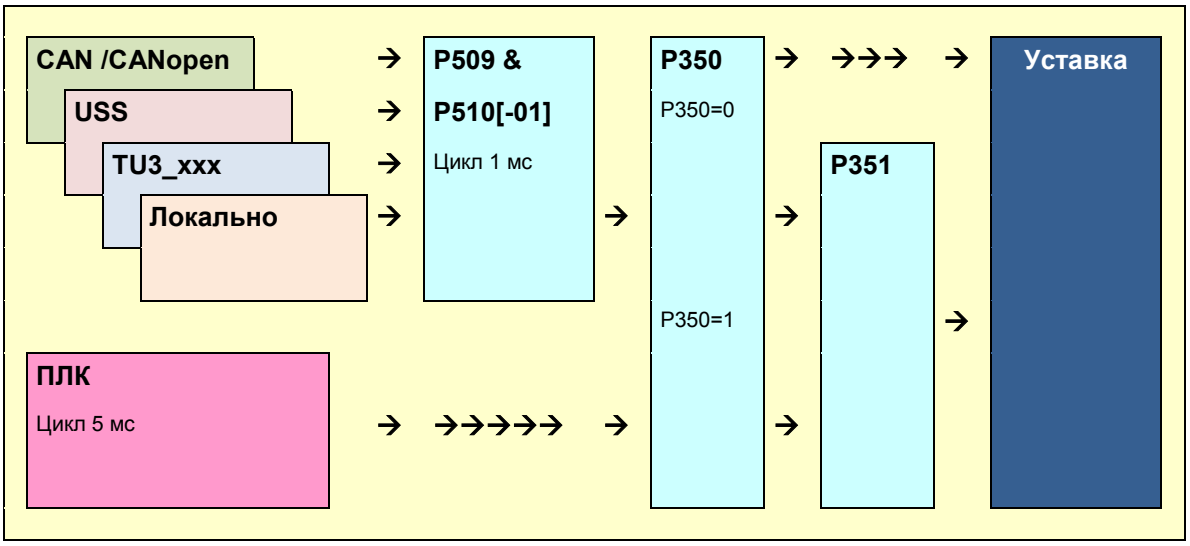
При использовании блоков управления движением (Motion) MC_Power, MC_Reset, MC_MoveVelocity, MC_Move, MC_Home или MC_Stop, не допускается использование параметра „PLC_Control_Word“ и параметров от „PLC_Set_Val1“ до „PLC_Set_Val5“. Иначе изменения функциональных блоков будут всегда перезаписываться значениями из списка переменных.

3.1.2.3 Программная задача

Выполнение программы ПЛК происходит в рамах одной задачи. Обращение к задаче производится циклически каждые 5 мс, а максимальная длительность ее обработки не превышает 3 мс. Если большая программа не успевает обработаться за это время, то ее выполнение прерывается и возобновляется с началом следующего интервала в 5 мс для следующей задачи.

3.1.2.4 Обработка уставки

Преобразователь получает уставки от множества источников, которые в конечном итоге соединяются друг с другом в единую результирующую частотного преобразователя посредством нескольких параметров.



При активированном ПЛК (P350=1) предустановка поступающих внешних уставок (главных уставок) осуществляется с помощью P509 & P510[-01]. Параметр P351 окончательно определяет, какие уставки будут использованы: от ПЛК или из входящих значений от P509/P510[-01]. Допускается смешанное использование уставок из обоих источников. Для вспомогательных уставок (P510[-02]) функция ПЛК не вызывает никаких изменений. Все источники вспомогательных уставок и ПЛК обладают равными правами при передаче своих вспомогательных уставок на частотный преобразователь.

3.1.2.5 Обработка данных через накопительный регистр

Накопительный регистр образует центральный вычислительный блок ПЛК. Почти все команды языка AWL выполняются исключительно совместно с накопительным регистром. ПЛК NORD имеет сразу три накопительных регистра: Akku1 и Akku2 размером 32 бита, а также текущий

результат AE в формате BOOL. Переменная AE используется для всех булевых операций загрузки, сохранения и сравнения. При загрузке булевой величины она отображается при помощи AE. Операнды сравнения отправляют результат в переменную AE, и на основании ее значения выполняются условные переходы. Akku1 и Akku2 применяются для всех операндов в формате данных BYTE, INT и DINT. Akku1 представляет собой главный накопительный регистр, тогда как Akku2 выполняет вспомогательные функции. Все операнды загрузки и сохранения выполняются через Akku1. В Akku1 также сохраняются результаты всех арифметических операций. В Akku2 при выполнении каждой команды загрузки смещается содержимое Akku1. Следующий далее оператор может объединить оба регистра, либо произвести вычисления и снова сохранить результат в Akku1, далее в общем случае именуемый также „accumulator“.

3.1.3 Функционал

ПЛК поддерживает множество операторов, функций и стандартных функциональных блоков, описанных в стандарте IEC1131-3. Подробное описание представлено в следующих главах. Далее также описываются дополнительно поддерживаемые функциональные блоки.

3.1.3.1 Библиотека управления движением Motion Control Lib

Библиотека Motion Control Lib создана на базе спецификации PLCopen „Function blocks for motion control“. В ней преимущественно содержатся функциональные блоки, описывающие порядок работы привода. Дополнительно библиотека также предоставляет функциональные блоки для чтения и записи параметров оборудования.

3.1.3.2 Электронные редукторы с летучей пилой

Частотный преобразователь обладает функциями электронного редуктора (синхронизация движения в режиме позиционирования) и летучей пилы. Благодаря этим функциям преобразователь может перемещаться синхронно с другим приводом. Кроме того дополнительная функция летучей пилы обеспечивает возможность позиционно точной синхронизации с движущимся приводом. Запустить и отключить режим электронного редуктора можно в любой момент. Команды перемещения и функции редуктора могут комбинироваться с классическим контролем положения. Для работы редуктора на ведущем устройстве обязательно должен быть установлен частотный преобразователь NORD с внутренней шиной CAN-Bus.

3.1.3.3 Визуализация

С помощью модулей ControlBox или ParameterBox может осуществляться визуализация рабочего состояния и параметризация частотного преобразователя. В качестве альтернативы для отображения информации может также использоваться функционал ведущего устройства CANopen панели ПЛК с шиной CAN-Bus.

ControlBox

Самым простым способом визуализации является использование модуля ControlBox. Доступ к 4-разрядному дисплею и состоянию клавиатуры может осуществляться посредством двух параметров. Это позволяет очень быстро создавать простые приложения человеко-машинного интерфейса (HMI). Чтобы ПЛК мог получать доступ к индикации для параметра P001 должно быть установлено значение „PLC-Controlbox Value“. Следует обратить внимание на то, что меню параметров больше не будет вызываться при помощи кнопок со стрелками. Вместо этого нужно будет одновременно нажимать кнопки „On“ и „Enter“.

ParameterBox

В режиме визуализации посредством ПЛК на дисплее P-Box может использоваться каждый из 80 символов (4 строки по 20 символов). С их помощью могут отображаться как цифры, так и

текст. Кроме того, ПЛК может передавать сигналы ввода с клавиатуры на P-Box. Это позволяет реализовывать сложные функции интерфейса HMI (индикация фактических значений, смена изображений, передача уставок и пр.). Доступ к индикации P-Box осуществляется посредством функциональных блоков в ПЛК. Для визуализации используются индикаторы рабочего состояния ParameterBox. Содержание индикаторов рабочего состояния настраивается при помощи параметра P1003 в P-Box. Этот параметр располагается в пункте основного меню „Display“. Для параметра P1003 должно быть установлено значение „PLC Display“. После этого можно снова осуществлять выбор индикаторов рабочих значений при помощи кнопок со стрелками "вправо" и "влево". Теперь здесь будет отображаться дисплей, управляемый ПЛК. После перезапуска оборудования настройки сохраняются.

3.1.3.4 Регулятор технологического процесса

Регулятор технологического процесса представляет собой ПИД регулятор типа PID-T1 с ограниченной выходной величиной. За счет данного функционального блока сложные регуляторы, управляющие различными процессами, например регулировкой давления, могут быть сформированы в ПЛК значительно более эффективным способом, чем с помощью часто используемых двухточечных регуляторов.

3.1.3.5 Передача данных через CANopen

Помимо стандартных имеющихся каналов передачи данных ПЛК предлагает также дополнительные способы коммуникации. Он позволяет устанавливать дополнительные коммуникационные связи с другими приборами через интерфейс CAN Bus частотного преобразователя, либо через системную шину. Для этого используется протокол CANopen. Коммуникационные связи при этом ограничиваются передачей сообщений PDO и командами NMT. Такая функция ПЛК не влияет на стандартный для частного преобразователя обмен данными по протоколу CANopen с помощью SDO, PDO1, PDO2 и Broadcast.

PDO (Process Data Objects)

Сообщения PDO позволяют управлять и контролировать работу других частотных преобразователей. К ПЛК могут подключаться также приборы других производителей. Это могут быть модули входов-выходов, датчики CANopen, панели и пр. Это позволяет увеличить количество входов/выходов преобразователя, а также использовать аналоговые выходы.

NMT (Network Management Objects)

Все приборы CANopen должны быть переведены с помощью Busmaster в состояние „Operational“ для шины шины CANopen Bus. Обмен сообщениями PDO возможен только при таком состоянии шины. Если шина CANopen Bus не имеет опции Busmaster, то соответствующую настройку следует установить при помощи ПЛК. Для этого предназначен функциональный блок FB_NMT.

3.2 Составление программы для ПЛК

Составление программы для ПЛК происходит исключительно в программе для ПКNORD CON. Открыть редактор программы для ПЛК можно через пункт меню „ File/New/PLC program “ или с помощью символа  . Эта кнопка доступна только если в обзоре устройств выбран прибор с функциями ПЛК.

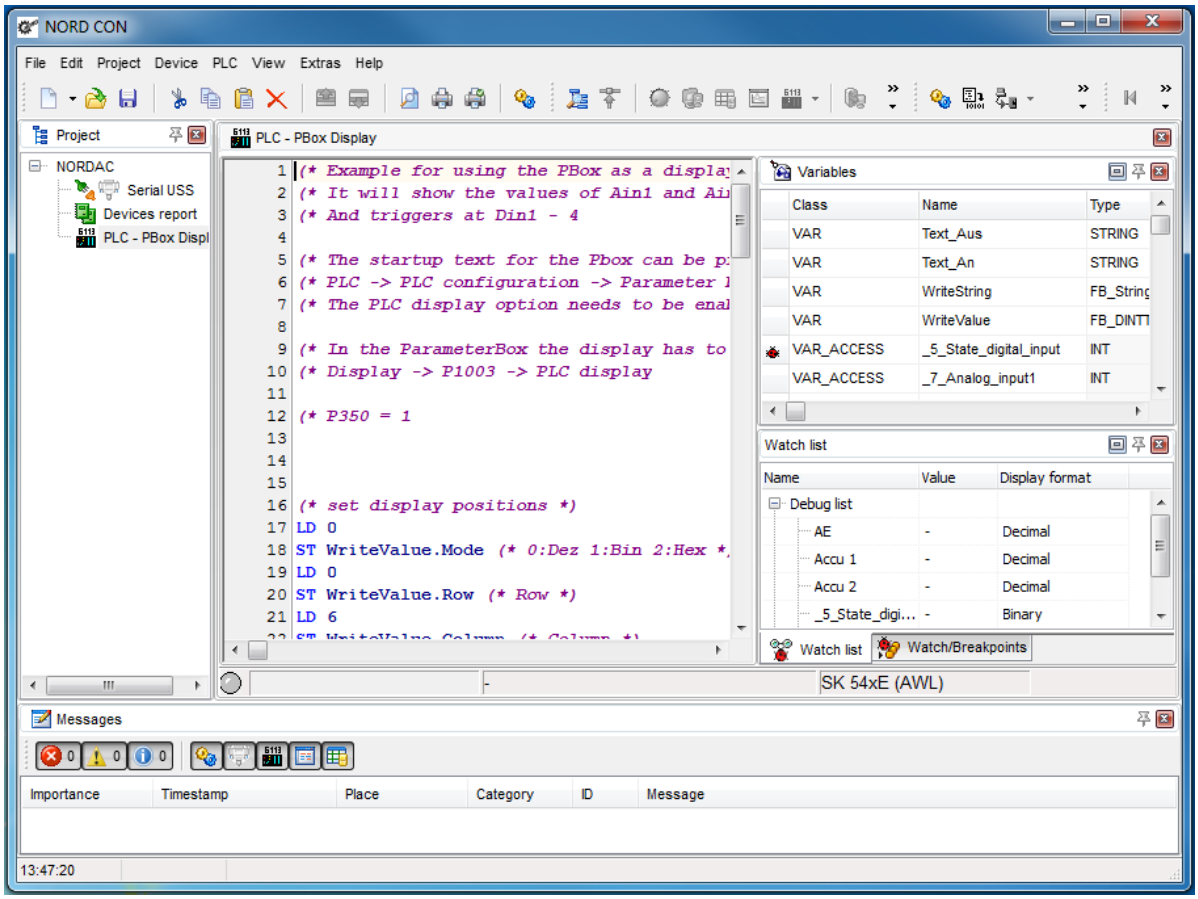
3.2.1 Загрузка, сохранение и печать

Функции загрузки, сохранения и печати реализуются при помощи соответствующих пунктов главного меню или символов. При открытии в диалоговом окне „Open“ рекомендуется установить тип файла „PLC program“ (*.awlx, *.nstx). При этом в окне будут показаны только те файлы, которые могут быть прочитаны редактором ПЛК. Для сохранения составленной программы для ПЛК должно быть активировано соответствующее окно редактора ПЛК. Сохранение выполняется при помощи команд „Save“ или „Save as“. Команда „Save as“ позволяет задать тип файла при сохранении (программа для ПЛК (*.awlx*.nstx)). Для вывода программы ПЛК на печать также следует активировать соответствующее окно. Для начала печати выбрать пункт „ File/Print“ или нажать соответствующий символ.

Кроме того, программа для ПЛК может быть сохранена как защищенная программа для ПЛК. Для этого в меню выбора файла следует установить "Backed-up AWL files" или "Backed up ST files". В результате этого будет сохранена программа для ПЛК в зашифрованном формате (*.awls или *.nsts), а также обычная версия программы Version (*.awlx, *.nstx). Зашифрованная программа может быть передана на прибор (см.).

3.2.2 Редактор

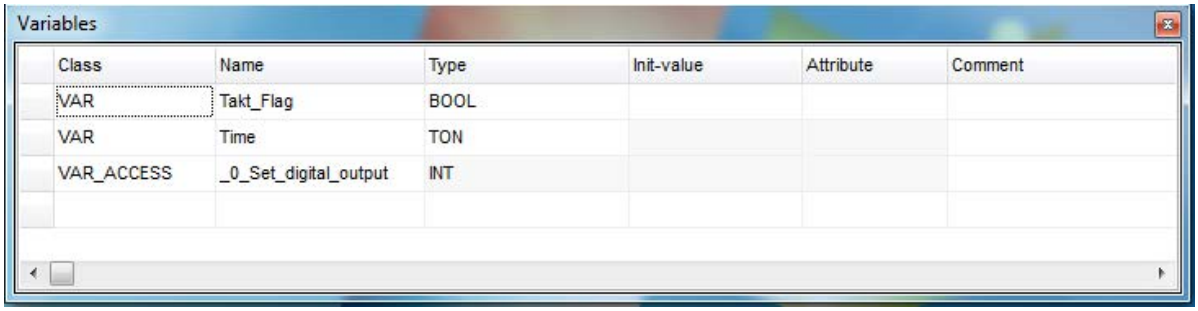
Редактор программы для ПЛК имеет четыре окна.



Отдельно о каждом окне рассказывается в следующих разделах руководства.

3.2.2.1 Переменные и описание функционального блока

В этом окне задаются все необходимые программе переменные, параметры процессов и функциональные блоки.



Переменные

Переменные создаются путем выбора „VAR“ в поле класса. Переменная может иметь любое имя. Для переменной следует выбрать один из типов: BOOL, BYTE, INT или DINT. Для переменной может быть задано начальное значение.

Параметры процессов

Для создания параметра в поле класса следует установить значение „VAR_ACCESS“. Имя параметра не может быть задано произвольно, а поле начального значения для него недоступно.

Функциональные блоки

В поле класса выбрать значение „VAR“. Для соответствующего экземпляра объекта ФБ может быть выбрано любое имя. Требуемый ФБ выбирается в поле типа. Начальное значение для ФБ не устанавливается.

Все пункты меню, связанные с окном переменных, вызываются через контекстное меню. С их помощью можно добавлять и удалять записи, а также активировать переменные и переменные процессов для отслеживания (функция Watchpoint) или отладки (Breakpoint).

3.2.2.2 Окно ввода

Окно ввода служит для ввода программы, а также представления программы на языке AWL. Оно обладает следующими функциями:

- Выделение синтаксиса
- Закладки
- Описание переменных
- Отладка

Выделение синтаксиса

Если редактор распознал команду и соответствующую ей переменную, то команда выделяется синим, а переменная черным цветом. Пока этого не произошло, то текст отображается тонким, наклонным, черным шрифтом.

Закладка

Так как программы в редакторе могут достигать значительных размеров, функция закладки позволяет отметить в тексте программы важные участки для быстрого перехода к ним. Чтобы отметить строку следует установить на нее курсор. Выбрать пункт меню „Switch bookmark“ (меню правой кнопки мыши). На строку будет установлена требуемая закладка. Для перехода к закладке использовать пункт меню „Go to bookmark“.

Описание переменных

Новые переменные можно описать с помощью редактора, используя меню редактора „Add Variable“ (меню правой кнопки мыши).

Отладка

Для функции отладки в редакторе устанавливаются позиции контрольных точек и точек прерывания. Для этого используются пункты меню „Switch breakpoint“ (точки прерывания) и „Switch monitoring point“ (контрольные точки). Задать положение точки прерывания можно также нажатием клавишей мышки на левой границе окна редактора. Переменные и параметры процессов, которые во время отладки требуется считывать с частотного преобразователя, должны быть выделены. Это можно сделать в редакторе с помощью пунктов меню „Debug variable“ и „Watch variable“. Нужные переменные следует выделить перед переходом в соответствующий пункт меню.

3.2.2.3 Окна отображения контрольной точки и точки прерывания (Watch- & Breakpoint)

В этом окне имеется две вкладки, описанных далее.

Точки прерывания

В этом окне отображаются все установленные точки прерывания и контрольные точки. Их включение/отключение выполняется при помощи установки и снятия флажков в соответствующих ячейках. Для удаления используется „Delete key“. Соответствующее меню открывается нажатием правой кнопки мыши.

Список отслеживания

Здесь отображаются все переменные, выбранные для отслеживания. В поле значения представлено текущее содержимое переменной. В поле индикации можно выбрать формат представления.

3.2.2.4 Окно сообщений ПЛК

В этом окне вводятся все сообщения о состоянии и ошибках ПЛК. Если трансляция программы выполнена без ошибок в окне появляется сообщение „Translated without error“. На строку ниже представлена информация об использовании ресурсов. При наличии ошибок в программе ПЛК появляется сообщение „Error X“, значение X соответствует количеству ошибок. В следующих строках отображается конкретное сообщение об ошибке в формате:

[номер строки]: описание ошибки

3.2.3 Передача программы ПЛК на прибор

Существует несколько способов передачи программы ПЛК на прибор.

Передача программы ПЛК напрямую:

1. Выбрать прибор в дереве проекта.
2. Открыть контекстное меню (нажать правую клавишу мыши)
3. Выполнить функцию "Transfer PLC program to device"
4. Выбрать файл в окне выбора файла и нажать "Open"

Передача программы ПЛК при помощи редактора ПЛК (оффлайн):

1. Открыть программу ПЛК при помощи функции "Open" (Datei->Öffnen)
2. Соединить редактор ПЛК с прибором (PLC->Verbinden)
3. Транслировать программу ПЛК
4. Передать программу ПЛК на прибор

Передача программы ПЛК при помощи редактора ПЛК (онлайн):

1. Отметить прибор в дереве проекта.
2. Запустить редактор ПЛК
3. Открыть программу ПЛК
4. Импортировать программу ПЛК в онлайн-вид
5. Транслировать программу ПЛК
6. Передать программу ПЛК на прибор 

Информация

SK 1xxE-FDS - ограниченное количество циклов записи

В приборах SK 155E-FDS / SK 175E-FDS в качестве среды хранения используется флеш-накопитель. Количество циклов записи для флеш-накопителей строго ограничено. Поэтому стандартная загрузка программы выполняется только в оперативную память RAM. Она может быть запущена и протестирована. Если после этого требуется перезапустить ПЛК, то программу следует повторно загрузить на прибор, чтобы установить переменные ПЛК. Если требуется сохранить программу на приборе на длительное время, пользователь должен выполнить действие "Transfer and store program to device".

3.2.4 Отладка

Поскольку очень редко программы запускаются с первого раза, в ПЛК NORD предусмотрен ряд возможностей для обнаружения ошибок. Их можно разделить на два пункта, о которых далее будет рассказано подробно.

3.2.4.1 Контрольные точки (Watchpoints)

Функция Watchpoint является самым простым вариантом отладки. Она обеспечивает быструю проверку поведения отдельных переменных. Для этого в любом месте программы назначается контрольная точка. Когда ПЛК обрабатывает эту программную строку, функция сохраняет до 5 значений и отображает в списке отслеживания (окно „Observation list“). Выбрать 5 отслеживаемых значений можно в окне ввода или окне переменных через контекстное меню. Если контрольная точка назначена в месте без программного кода, NORD CON выполняет поиск предшествующей строки кода. При достижении этой строки кода в ходе выполнения программы происходит обновление значений. Если контрольная точка была пропущена в результате выполнения перехода (JMP, IF, Switch), то значения не обновляются.

Информация

В текущей версии переменные функциональных блоков не могут быть добавлены в список отслеживания!

3.2.4.2 Точки прерывания (Breakpoints)

Точки прерывания позволяют остановить выполнение программы ПЛК на определенной строке. Когда ПЛК доходит до точки прерывания происходит считывания текущих значений Akku1 и Akku2, а также всех переменных, выбранных через пункт меню „Debug variable“ (контекстное меню). В программе ПЛК можно установить до 5 точек прерывания. Запуск функции осуществляется при помощи символа  . Программа выполняется до тех пор пока не сработает точка прерывания. При повторном нажатии кнопки с символом программа запускается далее и выполняется до следующей точки прерывания. Если требуется продолжить выполнение программы без прерываний используется символ  .

3.2.4.3 Пошаговое выполнение (Single Step)

Этот метод отладки позволяет обрабатывать программу для ПЛК строка за строкой. При выполнении каждого отдельного шага программы все выбранные переменные считываются с прибора- и отображаются в окне „Observation list“. Выбрать отслеживаемые значения можно в окне ввода или окне переменных с помощью меню правой кнопки мыши. Обязательным условием работы пошагового метода является назначение минимум одной точки прерываниям перед началом отладки. Для включения режима отладки используется символ  . После достижения программой первой точки прерывания начинается отладка последующих строк программы в пошаговом режиме при помощи символа  . За некоторыми командными строками скрываются несколько отдельных команд. Из-за этого может произойти так, что два или более отдельных шага будут обработаны до того как в окне ввода произойдет переход к отображению следующего шага. Текущее положение указывается маленькой стрелкой в левом окне редактора ПЛК. При нажатии на символ  выполнение программы продолжается до следующей точки прерывания. Если требуется продолжить выполнение программы без прерываний используется символ  .

3.2.5 Конфигурация ПЛК

Диалоговое окно конфигурации ПЛК открывается при помощи символа  . Здесь устанавливаются некоторые основные настройки ПЛК, о которых подробно рассказывается далее.

Контроль времени цикла

Функция контролирует максимальное время обработки для цикла ПЛК. Это позволяет отделять ошибочно запрограммированные непрерывные циклы в программе для ПЛК. В случае превышения допустимой продолжительности цикла на частотном преобразователе возникает ошибка E22.4.

Разрешить использование ФБ ParameterBox

Если программа ПЛК использует для визуализации модуль ParameterBox, следует активировать данную опцию. Иначе в соответствующих функциональных блоках при запуске частотного преобразователя будет возникать ошибка компилятора.

Недопустимые управляющие данные

ПЛК может анализировать команды управления, поступающие через возможные имеющиеся системы шин. При этом команды пропускаются только в том случае, если установлен бит „PZD valid“ (бит 10). Чтобы ПЛК мог анализировать также команды управления, не соответствующие протоколу USS, следует активировать данную опцию. Тогда бит 10 первой команды опрашиваться не будет.

"Теплый" запуск после ошибки

При запуске ПЛК все переменные всегда загружаются с начальным значением "0". При этом не важно, выполняется ли запуск после остановки, загрузки программы или ошибки ПЛК. При использовании опции теплого запуска содержимое переменных не изменяется. Теплый запуск выполняется после срабатывания команды остановки ПЛК или ошибки.

Не останавливать системное время в точке прерывания

В ходе отладки, если ПЛК находится в точке прерывания или в пошаговом режиме, системное время останавливается. Системное время служит основой для работы всех таймеров ПЛК. Если требуется не останавливать системное время во время отладки, необходимо активировать данную функцию.

3.3 Функциональные блоки

Функциональные блоки - это небольшие программы, которые могут сохранять свои параметры состояния во внутренних переменных. По этой причине для каждого функционального блока в списке переменных должен быть создан свой экземпляр объекта NORD CON. Например, если таймер должен параллельно контролировать три времени, то и в списке переменных он должен быть задан три раза.

Информация **Определение фронта сигнала**

Чтобы последующие функциональные блоки могли распознавать фронт сигнала на входе нужно чтобы вызов функции был произведен на входе с различными состояниями.

3.3.1 CANopen

С помощью функциональных блоков ПЛК может конфигурировать каналы сообщений PDO, управлять ими и отправлять на них сигналы. Сообщения PDO позволяют ПЛК отправлять или принимать до 8 бит информации о процессах. Обращение к каждому из таких сообщений PDO выполняется через собственный адрес (COB-ID). ПЛК может конфигурировать до 20 сообщений PDO. Для более эффективной работы эти сообщения не отправляются напрямую COB-ID. Вместо этого адрес прибора и номер PDO передаются в ФБ. Адрес COB-ID определяется в итоге на основании заранее определенных связей (CiA DS301). Таким образом для ПЛК могут быть получены нижеследующие COB-ID.

Отправка сообщений PDO		**Контролируемые PDO**	
PDO	**COB-ID**	**PDO**	**COB-ID**
PDO1	200h + адрес устройства	PDO1	180h + адрес устройства
PDO2	300h + адрес устройства	PDO2	280h + адрес устройства
PDO3	400h + адрес устройства	PDO3	380h + адрес устройства
PDO4	500h + адрес устройства	PDO4	480h + адрес устройства

NORD Частотный преобразователь используют PDO1 для передачи данных процессов, а PDO2 используется только для передачи уставок/фактических значений 4 и 5.

3.3.1.1 Обзор

Функциональный блок	**Описание**
FB_PDOConfig	Конфигурация PDO
FB_PDOSend	Отправка PDO
FB_PDOReceive	Получение PDO
FB_NMT	Разблокировка и блокировка PDO

3.3.1.2 FB_NMT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

После выполнения *Power UP* все абоненты CAN находятся в состоянии шины Pre-Operational. В этом состоянии они не могут принимать или отправлять сообщения PDO. Чтобы ПЛК мог осуществлять обмен данными с другими абонентами они должны находиться в состоянии Operational. Как правило за это отвечает Busmaster. Если Busmaster отсутствует, то эту задачу может выполнять блок FB_NMT. Входы **PRE**, **OPE** или **STOP** позволяют менять состояние всех абонентов, подключенных к шине. Положительный фронт переводит входы в состояние **EXECUTE**. Функция должна вызываться до тех пор, пока на выходе не будет получено значение 1 для **DONE** или **ERROR**.

Получение значения 1 на выходе **ERROR** означает либо отсутствие напряжения 24В на шине RJ45 CAN преобразователя, либо драйвер CAN преобразователя находится в состоянии *Bus off*. Отрицательный фронт **EXECUTE** возвращает 0 на все выходы.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
EXECUTE	Выполнение	BOOL	DONE	Отправка команды NMT	BOOL
PRE	Перевод всех абонентов в состояние Pre-Operational	BOOL	ERROR	Ошибка в ФБ	BOOL
OPE	Перевод всех абонентов в состояние Operational	BOOL			
STOP	Перевод всех абонентов в состояние Stopped	BOOL			

3.3.1.3 FB_PDOConfig

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X		

Данный ФБ используется для конфигурации PDO. Экземпляр объекта данной функции позволяет выполнить настройку всех нужных PDO. Для каждого PDO требуется вызвать ФБ только один раз. При этом может быть настроено до 20 PDO. Каждое PDO обладает своим набором параметров. Назначение PDO другим ФБ CANopen производится при помощи номера окна сообщений. **TARGETID** содержит адрес устройства. У NORD Частотный преобразователь для настройки используется P515 или DIP-переключатель. Для PDO указывается требуемый номер окна сообщений (см. введение). **LENGTH** содержит отправляемую длину PDO. **DIR** определяет направление отправки/приема. При положительном фронте на **EXECUTE** происходит передача данных. Запрос на выход **DONE** может быть отправлен сразу после вызова ФБ. Если **DONE** имеет значение 1, выполняется конфигурирование канала PDO. Если **ERROR** = 1, значит возникла проблема, причина которой хранится в **ERRORID**. Отрицательный фронт **EXECUTE** возвращает 0 на все выходы.

Отправка сообщений PDO		**Контролируемые PDO**	
PDO	**COB-ID**	**PDO**	**COB-ID**
PDO1	200h + адрес устройства	PDO1	180h + адрес устройства
PDO2	300h + адрес устройства	PDO2	280h + адрес устройства
PDO3	400h + адрес устройства	PDO3	380h + адрес устройства
PDO4	500h + адрес устройства	PDO4	480h + адрес устройства
PDO5	180h + адрес устройства	PDO5	200h + адрес устройства
PDO6	280h + адрес устройства	PDO6	300h + адрес устройства
PDO7	380h + адрес устройства	PDO7	400h + адрес устройства
PDO8	480h + адрес устройства	PDO8	500h + адрес устройства

VAR_INPUT			VAR_OUTPUT		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
EXECUTE	Выполнение	BOOL	**DONE**	Конфигурация PDO	BOOL
NUMBER	Номер окна сообщений Диапазон значений = от 0 до 19	BYTE	**ERROR**	Ошибка в ФБ	BOOL
TARGETID	Адрес прибора Диапазон значений = от 1 до 127	BYTE	**ERRORID**	Код ошибки	INT
PDO	PDO Диапазон значений = от 1 до 4	BYTE			
LENGTH	Длина PDO Диапазон значений = от 1 до 8	BYTE			
DIR	Отправка или прием Отправка = 1 / прием = 0	BOOL			

ERRORID	Описание
0	Отсутствие ошибки
1800h	Превышен диапазон значений
1801h	Превышен диапазон значений TARGETID
1802h	Превышен диапазон значений PDO
1803h	Превышен диапазон значений LENGT

Информация — Исключение двойного использования CAN ID

Не допускается параметрирование CAN-ID, который уже используется устройством!

Связанные адреса приема:

- CAN ID = 0x180 + P515[-01] PDO1
- CAN ID = 0x180 + P515[-01]+1 CAN ID для абсолютного энкодера
- CAN ID = 0x280 + P515[-01] PDO2

Связанные адреса отправки:

- CAN ID = 0x200 + P515[-01] PDO1
- CAN ID = 0x300 + P515[-01] PDO2

Пример на ST:

```
(* Конфигуратор PDO*)
PDOConfig(
    Execute := TRUE,
    (* Конфигурация окна сообщения Messagebox 1 *)
    Number := 1,
    (* Установить номер узла CAN *)
    TargetID := 50,
    (* Выбор PDO (стандарт для команды управления PDO1, Sollwert1, Sollwert2, Sollwert3) *)
    PDO := 1,
    (* Определить длину данных (стандарт для PDO1 равен 8 *)
    LENGTH := 8,
    (* Отправка *)
    Dir := 1);
```

или

```
(* Конфигуратор PDO*)
PDOConfig(
    Execute := TRUE,
    (* Конфигурация окна сообщения Messagebox 1 *)
    Number := 2,
    (* Установить номер узла CAN *)
    TargetID := 50,
    (* Выбор PDO (стандарт для команды управления PDO2 Sollwert4, Sollwert SK540E) *)
    PDO := 2,
    (* Определить длину данных (стандарт для PDO2 равен 4 *)
    LENGTH := 4,
    (* Отправка *)
    Dir := 1);
```

или

```
(* Конфигуратор PDO*)
PDOConfig(
    Execute := TRUE,
    (* Конфигурация окна сообщения Messagebox 2 *)
    Number := 2,
    (* Установить номер узла CAN *)
    TargetID := 50,
    (* Выбор PDO (стандарт для команды управления PDO1, Istwert1, Istwert2, Istwert3) *)
    PDO := 1,
    (* Определить длину данных (стандарт для PDO1 равен 8 *)
    LENGTH := 8,
    (* Получение *)
    Dir := 0);
```

3.3.1.4 FB_PDOReceive

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X		

Этот ФБ осуществляет контроль входящих сообщений для предварительно сконфигурированного канала PDO. Функция запускается когда входу **ENABLE** соответствует значение 1. После вызова функции выполняется проверка выхода **NEW**. Если сообщение поступило, то выходу будет соответствовать значение 1. Содержимое выхода **NEW** удаляется при следующем вызове функции. Переменные с **WORD1** по **WORD4** хранят полученные данные. Переменная **TIME** позволяет контролировать циклический прием сообщений через канал PDO. В **TIME** указывается значение от 1 до 32767 мс, которое соответствует временному промежутку приема сообщений. Иначе в ФБ переходит в состояние ошибки (**ERROR** = 1). Значение 0 отключает функцию Таймер контролирующей функции работает с интервалами 5 мс. В случае ошибки **ERROR** принимает значение 1. В этом случае значение **DONE** равно 0. **ERRORID** содержит соответствующий код ошибки. При отрицательном фронте на **ENABLE** производится сброс значений **DONE**, **ERROR** и **ERRORID**.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
ENABLE	Выполнение	BOOL	**NEW**	Получение нового PDO	BOOL
NUMBER	Номер окна сообщений Диапазон значений = от 0 до 19	BYTE	**ERROR**	Ошибка в ФБ	BOOL
TIME	Функция Watchdog Диапазон значений = от 0 до 32767 0 = выкл. от 1 до 32767 = время контроля	INT	**ERRORID**	Код ошибки	INT
			WORD1	Принимаемые данные Слово данных 1	INT
			WORD2	Принимаемые данные Слово данных 2	INT
			WORD3	Принимаемые данные Слово данных 3	INT
			WORD4	Принимаемые данные Слово данных 4	INT

ERRORID	**Описание**
0	Отсутствие ошибки
1800h	Превышен диапазон значений
1804h	Неверная конфигурация выбранного окна
1805h	Отсутствует напряжение 24 В на драйвере шины или он находится в состоянии „Bus off“
1807h	Timeout получения (функция Watchdog)

Информация **Время цикла ПЛК**

Цикл ПЛК составляет 5 мс, то есть при вызове функции в программе ПЛК каждые 5 мс может считываться только одно сообщение CAN. Если сообщения отправляются быстрее, то они могут перезаписываться.

Пример на ST:

```
IF bFirstTime THEN
  (* Перевести устройства в состояние Pre-Operational *)
  NMT(Execute := TRUE, OPE := TRUE);
  IF not NMT.Done THEN
    RETURN;
  END_IF;

  (* Конфигуратор PDO*)
  PDOConfig(
    Execute := TRUE,
    (* Конфигурация окна сообщения Messagebox 2 *)
    Number := 2,
    (* Установить номер узла CAN *)
    TargetID := 50,
    (* Выбор PDO (стандарт для команды управления PDO1, Istwert1, Istwert2, Istwert3) *)
    PDO := 1,
    (* Определить длину данных (стандарт для PDO1 равен 8 *)
    Length := 8,
    (* Получение *)
    Dir := 0);
END_IF;

(* Считывание состояний и факт.значений *)
PDOReceive(Enable := TRUE, Number := 2);
IF PDOReceive.New THEN
  State := PDOReceive.Word1;
  Sollwert1 := PDOReceive.Word2;
  Sollwert2 := PDOReceive.Word3;
  Sollwert3 := PDOReceive.Word4;
END_IF
```

3.3.1.5 FB_PDOSend

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X		

Этот ФБ позволяет отправлять сообщения PDO на предварительно сконфигурированный канал. Сообщения могут отправляться однократно или циклически. Данные для отправки заносятся в переменные **WORD1** - **WORD4**. Отправка PDO может выполнятся независимо от состояния CANopen частотного преобразователя. Выбор предварительно сконфигурированного канала производится при помощи **NUMBER**. В **WORD1** - **WORD4** заносятся данные для отправки. С помощью переменной **CYCLE** осуществляется выбор между однократной отправкой (значение=0) или циклической. При положительном фронте **EXECUTE** выполняется отправка PDO. Если **DONE** = 1, значит все данные были корректны и PDO успешно отправлены. Если **ERROR** = 1, значит произошла ошибка. Точная причина ошибки указывается в **ERRORID**. При отрицательном фронте **EXECUTE** значения всех выходов сбрасываются. Опорное время ПЛК составляет 5 мс, что также действительно для входа **CYCLE**. Циклы отправки могут выполнятся только с кратностью 5 мс.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
EXECUTE	Выполнение	**BOOL**	**DONE**	PDO отправлено = 1	BOOL
NUMBER	Номер окна сообщений Диапазон значений = от 0 до 19	**BYTE**	**ERROR**	Ошибка в ФБ	BOOL
CYCLE	Цикл отправки Диапазон значений = от 0 до 255 0 = выкл. от 1 до 255 = время цикла в мс	**BYTE**	**ERRORID**	Код ошибки	INT
WORD1	Отправляемые данные Слово данных 1	**INT**			
WORD2	Отправляемые данные Слово данных 2	**INT**			
WORD3	Отправляемые данные Слово данных 3	**INT**			
WORD4	Отправляемые данные Слово данных 4	**INT**			

ERRORID	**Описание**
0	Отсутствие ошибки
1800h	Превышен диапазон значений
1804h	Неверная конфигурация выбранного окна
1805h	Отсутствует напряжение 24 В на драйвере шины или он находится в состоянии „Bus off“

Когда **DONE** принимает значение 1, сообщение для отправки передается на модуль CAN, но не отправляется. Собственно отправка происходит фоново. Если ФБ предусматривает отправку нескольких сообщений одно за другим, может возникнуть ситуация, когда при новом вызове программы предыдущее сообщение еще не отправлено. Определить это можно по отсутствию значения 1 у сигналов **DONE** и **ERROR** после вызова **CAL**. В этом случае следует повторять вызов **CAL**, пока не будет получено значение 1 для одного из сигналов. Если один должен ФБ описывать несколько различных CAN-ID, это можно выполнить при помощи новой конфигурации ФБ. При этом это не должен быть тот же цикл ПЛК, что и у отправки. Иначе возникает опасность, что в процессе конфигурации предназначенное для отправки сообщение будет удалено функцией FB_PDOConfig.

Пример на ST:

```
IF bFirstTime THEN
  (* Перевести устройства в состояние Pre-Operational *)
  NMT(Execute := TRUE, OPE := TRUE);
  IF not NMT.Done THEN
    RETURN;
  END_IF;

  (* Конфигуратор PDO*)
  PDOConfig(
    Execute := TRUE,
    (* Конфигурация окна сообщения Messagebox 2 *)
    Number := 2,
    (* Установить номер узла CAN *)
    TargetID := 50,
    (* Выбор PDO (стандарт для команды управления PDO1, Istwert1, Istwert2, Istwert3) *)
    PDO := 1,
    (* Определить длину данных (стандарт для PDO1 равен 8 *)
    Length := 8,
    (* Получение *)
    Dir := 0);
END_IF;

(* Считывание состояний и факт.значений *)
PDOReceive(Enable := TRUE, Number := 2);
IF PDOReceive.New THEN
  State := PDOReceive.Word1;
  Sollwert1 := PDOReceive.Word2;
  Sollwert2 := PDOReceive.Word3;
  Sollwert3 := PDOReceive.Word4;
END_IF
```

3.3.2 Электронные редукторы с летучей пилой*

Для функции *электронного редуктора* („синхронизация с вращением на один и тот же угол за один и тот же промежуток времени“) и вспомогательной функции *летучей пилы* предусмотрено два функциональных блока, обеспечивающих управление этими функциями. Кроме того, для выполнения обоих функциональных блоков на ведущем и ведомом частотных преобразователях могут быть настроены разные параметры. В качестве примера в нижеследующей таблице рассмотрена реализация для SK 540E.

Ведущий ЧП			**Ведомый ЧП**		
Параметр	Настройка	Значение	Параметр	Настройка	Значение
P502[-01]	20	Уставка частоты по линейному изменению	P509	10 *	Широкое вещание CANopen Broadcast*
P502[-02]	15	Текущая позиция в инкр. High – Word	P510[-01]	10	Широкое вещание CANopen Broadcast
P502[-03]	10	Текущая позиция в инкр. Low – Word	P510[-02]	10	Широкое вещание CANopen Broadcast
P503	3	CANopen	P505	0	0,0 Гц
P505	0	0,0 Гц	P515[-02]	P515[-03]Master	Адрес Broadcast Slave
P514	5	250 Кбод (мин. 100 Кбод)	P546[-01]	4	Сложение частот
P515[-03]	P515[-02]Slave	Адрес Broadcast Master	P546[-02]	24	Уст.полож. в инкр. High – Word
			P546[-03]	23	Уст.полож. в инкр. Low – Word
			P600	1,2	Контроль положения
			Только для ФБ FB_Gearing		
			P553[-01]	21	Полож. Уставка полож. Low Word
			P553[-02]	22	Полож. Уставка полож. High Word
* (P509) не обязательно должен быть {10} „CANopen Broadcast“. (P502 [-01]) на ведущем устройстве для настройки {21} "Фактическая частота без проскальзывания".					

Информация **Формат передачи фактического положения**

Фактическое положение ведущего устройства должно передаваться в формате „Increments“ (Inc).

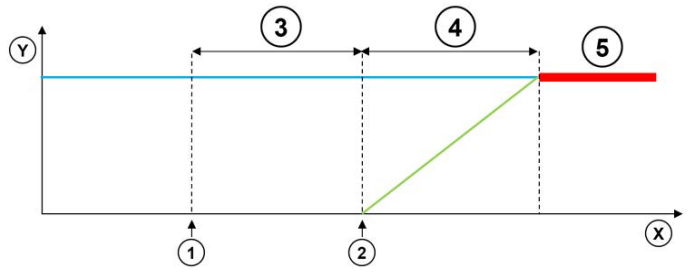
3.3.2.1 Обзор

Функциональный блок	Описание
FB_Gearing	ФБ для простой функции редуктора
FB_FlyingSaw	ФБ для функции редуктора с летучей пилой.

3.3.2.2 FB_FlyingSaw

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	

Функция летучей пилы представляет собой дополнение к функции редуктора. Функция летучей пилы обеспечивает возможность позиционно точной синхронизации с движущимся приводом. Синхронизация выполняется противоположно относительно функции FB_Gearing, то есть ведомая ось перемещается синхронно в позицию ведущего, занимаемую при запуске функции „Flying Saw“. Процесс синхронизации представлен на следующем рисунке.



1	Положение инициатора
2	Начальная точка ведомого устр-ва
3	Перемещение инициатора в начальную точку ведомого ЧП-
4	Ускорение
5	Синхронизация обоих приводов
X	Положение
Y	Скорость

При запуске функции ведомый частотный преобразователь ускоряется до скорости ведущей оси. Линейное изменение ускорения определяется посредством кривой **ACCELERATION**. При низкой скорости линейное изменение будет более плоским, а при более высокой скорости ведущего устройства линейное изменение для ведомого частотного преобразователя будет иметь более крутой вид. Кривая ускорения задается в оборотах (1000 = 1,000 rev), если параметр P553 задан в качестве уставки положения. Если для уставки положения P553 используется формат INC, то для описания кривой ускорения также используется инкремент.

Если пусковое устройство устанавливается перед положением ведомого приводного механизма на расстоянии, сохраненном в **ACCELERATION**, то ведомое устройство точно синхронизируется с ведущим приводным механизмом по положению срабатывания.

ФБ включается посредством входа **ENABLE**. Запуск функции осуществляется либо через цифровой вход (P420[-xx]=64, *Start Fliegende Säge*) или с помощью сигнала **EXECUTE**. Частотный преобразователь ускоряется до скорости ведущей оси. При достижении синхронизации с ведущей осью выход **DONE** принимает значение 1.

Отключение функции редуктора производится с помощью входа **STOP** или цифровой функции P420[-xx] = 77, *Fliegende Säge anhalten*, частотный преобразователь снижает скорость до 0Гц и останавливается. Вход **HOME** используется для передачи преобразователю команды возвращения в абсолютное положение 0. По окончании выполнения команд **HOME** или **STOP** соответствующий назначенный выход остается активным. Повторное использование **EXECUTE** или цифрового входа позволяет снова запустить функцию редуктора. Цифровая функция входа (P420[-xx] = 63, *Gleichlauf ausschalten*) останавливает функцию привода и возвращает в абсолютное положение 0.

Если выполнение функции было прервано с помощью MC_Stop, **ABORT** принимает значение 1. В случает ошибки ERROR принимает значение 1, а код ошибки заносится в ERRORID. Все три выхода сбрасываются, когда ENABLE переключается на 0.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
ENABLE	Разблокировка	BOOL	**VALID**	Достижение заданной уставки частоты	BOOL
EXECUTE	Начало синхронизации	BOOL	**DONEHOME**	Окончен возврат в исходное положение	
STOP	Конец синхронизации	BOOL	**DONESTOP**	Выполнена команда остановки	
HOME	Откат в положение 0	BOOL	**ABORT**	Прерванная команда	BOOL
ACCELERATION	Путь разгона (1rev. = 1 000)	DINT	**ERROR**	Ошибка в ФБ	BOOL
			ERRORID	Код ошибки	INT

ERRORID	**Описание**
0	Отсутствие ошибки
1000ч	ЧП не разблокирован
1200ч	Контроль положения не активирован

3.3.2.3 FB_Gearing

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	

Функциональный блок FB_Gearing позволяет синхронизировать положение и скорость вращения частотного преобразователя относительно положения и скорости ведущего преобразователя. Ведомое устройство, использующее данную функцию, всегда следует за движениями ведущего преобразователя.

Выполняется абсолютная синхронизация, то есть положение ведущего и ведомого устройств всегда одинаковое.

Информация

Если при переходе в режим редуктора ведомое устройство находится в другом положении, чем ведущее, то ведомое устройство переводится в положение ведущего с максимальной частотой.

Если задано передаточное соотношение, то после повторного включения также будет принято новое положение.

Значение позиции, по которому буде выполняться синхронизация, а также скорость вращения, передаются через канал Broadcast. Вход **ENABLE** активирует функцию. Контроль положения и выходной каскад должны быть активированы. Разблокировать выходной каскад можно, например, с помощью функции MC_Power. Если **ENABLE** принимает значение 0, частотный преобразователь снижает скорость до 0Гц и останавливается. Преобразователь возвращается в режим контроля положения. При активации MC_Stop частотный преобразователь запускает режим редуктора, а выход **ABORT** принимает значение 1. В случает ошибки в ФБ ERROR принимает значение 1, а причина неисправности заносится в ERRORID. Значение 0 на **ENABLE** сбрасывает **ERROR, ERRORID** и **ABORT**.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
ENABLE	Синхронизация движения активирована	BOOL	**VALID**	Функция редуктора активирована	BOOL
RELATIVE	Относительный режим (от V2.1)	BOOL	**ABORT**	Прерванная команда	BOOL
			ERROR	Ошибка в ФБ	BOOL
			ERRORID	Код ошибки	INT

ERRORID	**Описание**
0	Отсутствие ошибки
1000h	ЧП не разблокирован
1200h	Контроль положения не активирован
1201h	Не задано верхнее положение High в уставках ПЛК
1202h	Не задано нижнее положение Low в уставках ПЛК

3.3.3 Motion Control

Библиотека Motion Control Lib создана на базе спецификации PLCopen „Function blocks for motion control“. Она содержит функциональные блоки для управления и перемещения частотного преобразователя, а также обеспечивает доступ к его параметрам. Для работы блоков управления движением следует выполнить ряд настроек в параметрах устройства.

Функциональный блок	Требуемые настройки
MC_MoveVelocity	• P350 = ПЛК активен • P351 = Получение главной уставки от ПЛК • P553 [-xx] = Уставка частоты • P600 = Контроль положения (режим позиционирования) выключен
MC_MoveAbsolute MC_MoveRelative MC_MoveAdditive MC_Home	• P350 = ПЛК активен • P351 = Получение главной уставки от ПЛК • P600 = Контроль положения (режим позиционирования) включен • Верхнее положение High должно быть задано в P553 [-xx] (PLC_Sollwerte) • Нижнее положение Low должно быть задано в P553 [-xx] (PLC_Sollwerte) • Уставка частоты должна быть задана в P553 [-xx] (PLC_Sollwerte)
MC_Power MC_Reset MC_Stop	• P350 = ПЛК активен • P351 = Получение команды управления от ПЛК

Информация

PLC_Sollwert от 1 до 5 и команда управления ПЛК также описываются переменными процессов. При использовании ФБ управления движением Motion Control не разрешается задавать соответствующие переменные процессов в таблице переменных, так как в этом случае данные Motion Control будут перезаписываться.

Информация **Определение фронта сигнала**

Чтобы последующие функциональные блоки могли распознавать фронт сигнала на входе нужно чтобы вызов функции был произведен на входе с различными состояниями.

Функциональный блок	**Описание**
MC_ReadParameter	Доступ для чтения параметров устройства
MC_WriteParameter	Доступ для записи параметров устройства
MC_MoveVelocity	Команда перемещения в режиме скорости вращения
MC_MoveAbsolute	Команда перемещения с абсолютным указанием положения
MC_MoveRelative	Команда перемещения с относительным указанием положения
MC_MoveAdditive	Команда перемещения с аддитивным указанием положения
MC_Home	Запуск возврата в исходное положение
MC_Power	Включение/выключение напряжения двигателя
MC_ReadStatus	Состояние устройства
MC_ReadActualPos	Считывание текущего положения
MC_Reset	Сброс ошибке прибора
MC_Stop	Остановка активной команды перемещения

3.3.3.1 MC_Control

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	

Данный ФБ служит для управления частотным преобразователем и позволяет использовать команды управления ЧП более детализировано, подобно MC_Power. Управление ЧП осуществляется посредством входов **ENABLE**, **DISABLEVOLTAGE** и **QUICKSTOP**, см. нижеследующую таблицу.

Блок входов			**Поведение частотного преобразователя**
ENABLE	**QUICKSTOP**	**DISABLEVOLTAGE**	
High	Low	Low	Включение частотного преобразователя.
Low	Low	Low	Частотный преобразователь останавливается до 0 Гц (P103) и отключает напряжение на двигателе.
X	X	High	Подача напряжения на частотный преобразователь сразу прекращается, двигатель отключается без торможения.
X	High	Low	Частотный преобразователь останавливается до 0 Гц (P103) и отключает напряжение на двигателе.

Вход **PARASET** позволяет настраивать активный набор параметров.

Если выход **STATUS** = 1, ЧП включен, на двигатель подается напряжение.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
ENABLE	Разблокировка	BOOL	**STATUS**	Подача тока на двигатель	BOOL
DISABLEVOLTAGE	Без напряжения	BOOL	**ERROR**	Ошибка в ФБ	BOOL
QUICKSTOP	Быстрый останов	BOOL	**ERRORID**	Код ошибки	INT
PARASET	Активный набор параметров Диапазон значений: 0 - 3	BYTE			

ERRORID	**Описание**
0	Отсутствие ошибки
1001ч	Активна функция остановки
1300h	Непредвиденное состояние ЧП

Пример на ST:

```
(* Разблокировка прибора с помощью Dig3*)
Control.Enable := _5_State_digital_input.2;
(* Наборы параметров определяются посредством Dig1 и Dig2. *)
Control.ParaSet := INT_TO_BYTE(_5_State_digital_input and 2#11);
Control;
(* Прибор разблокирован? *)
if Control.Status then
  (* Требуется переместить в другое положение? *)
  if SaveBit3 <> _5_State_digital_input.3 then
    SaveBit3 := _5_State_digital_input.3;
    if SaveBit3 then
      Move.Position := 500000;
    else
      Move.Position := 0;
    end_if;

    Move(Execute := False);
  end_if;
end_if;

(* Перейти в положение когда прибор разблокирован.  *)
Move(Execute := Control.Status);
```

3.3.3.2 MC_Control_MS

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие							X

Данный ФБ используется для управления пусковым устройством (ПУ).

Блок входов				**Поведение частотного преобразователя**
ENABLE_RIGHT	**ENABLE_LEFT**	**QUICKSTOP**	**DISABLEVOLTAGE**	
High	Low	Low	Low	Включение ПУ, вращение вправо
Low	High	Low	Low	Включение ПУ, вращение влево
High	High	Low	Low	Выключение ПУ
Low	Low	Low	Low	ПУ останавливается до 0 Гц (P103) и отключает напряжение на двигателе.
X	X	X	High	Подача напряжения на ПУ сразу прекращается, двигатель отключается без торможения.
X	X	High	Low	ПУ выполняет быстрый останов (P426)и отключает напряжение на двигателе.

(X = уровень на входе не важен)

Если выход **STATUS** = 1, ПУ включен, на двигатель подается напряжение.

Если **OPENBRAKE** принимает значение 1, тормоз открывается.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
ENABLE_RIGHT	Вправо пуск право	BOOL	**STATUS**	Подача тока на двигатель	BOOL
ENABLE_LEFT	Влево пуск влево	BOOL	**ERROR**	Ошибка в ФБ	BOOL
DISABLEVOLTAGE	Без напряжения	BOOL	**ERRORID**	Код ошибки	INT
QUICKSTOP	Быстрый останов	BOOL			
OPENBRAKE	Открыть тормоз	BOOL			

ERRORID	**Описание**
0	Отсутствие ошибки
1001ч	Активна функция остановки
1300h	Непредвиденное состояние ПУ

3.3.3.3 MC_Home

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие		X	X	X	X	X	

Дает команду частотному преобразователю начать перемещение в заданную точку, как только **EXECUTE** меняет значение с 0 на 1 (фронт). Перемещение ЧП выполняется согласно уставке частоты, заданной **VELOCITY**. При активации входа с сигналом контрольного положения (P420[-xx] = заданная точка), направление вращения меняется на обратное. При отрицательном фронте сигнала контрольного положения принимается значение, хранящееся в **POSITION**. После этого частотный преобразователь останавливается до 0Гц, сигнал **DONE** принимает значение 1. Во время выполнения перемещения в исходное положение **HOME** выход **BUSY** остается активным.

При необходимости прерывания данной процедуры (например, другим функциональным блоком MC), используется сигнал **COMMANDABORTED**.

В случае ошибки **ERROR** принимает значение 1. В этом случае значение **DONE** равно 0. **ERRORID** содержит соответствующий код ошибки.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
EXECUTE	Разблокировка	BOOL	**DONE**	Достижение заданной уставки положения	BOOL
POSITION	Уставка положения	DINT	**COMMAND-ABORTED**	Прерванная команда	BOOL
VELOCITY	Уставка частоты	INT	**ERROR**	Ошибка в ФБ	BOOL
			ERRORID	Код ошибки	INT
			BUSY	Активирован возврат в исходное положение	BOOL

ERRORID	**Описание**
0	Отсутствие ошибки
1000h	ЧП не разблокирован
1200h	Контроль положения не активирован
1201h	Не задано верхнее положение High в уставках ПЛК (P553)
1202h	Не задано нижнее положение Low в уставках ПЛК (P553)
1D00h	Абсолютный энкодер не поддерживается

3.3.3.4 MC_Home (SK 5xxP)

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X						

Дает команду частотному преобразователю начать перемещение в заданную точку, как только **EXECUTE** меняет значение с 0 на 1 (фронт). Перемещение ЧП выполняется согласно уставке частоты, заданной **VELOCITY**. При активации входа с сигналом контрольного положения (P420[-xx] = заданная точка), направление вращения меняется на обратное. При отрицательном фронте сигнала контрольного положения принимается значение, хранящееся в **POSITION**. После этого частотный преобразователь останавливается до 0Гц, сигнал **DONE** принимает значение 1. Во время выполнения перемещения в исходное положение **HOME** выход **BUSY** остается активным.

При необходимости прерывания данной процедуры (например, другим функциональным блоком MC), используется сигнал **COMMANDABORTED**.

В случае ошибки **ERROR** принимает значение 1. В этом случае значение **DONE** равно 0. **ERRORID** содержит соответствующий код ошибки.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
EXECUTE	Разблокировка	BOOL	**DONE**	Достижение заданной уставки положения	BOOL
POSITION	Уставка положения	DINT	**COMMAND-ABORTED**	Прерванная команда	BOOL
VELOCITY	Уставка частоты	INT	**ERROR**	Ошибка в ФБ	BOOL
MODE	см. ниже	BYTE	**ERRORID**	Код ошибки	INT
			BUSY	Активирован возврат в исходное положение	BOOL

ERRORID	**Описание**
0	Отсутствие ошибки
1000h	ЧП не разблокирован
1200h	Контроль положения не активирован
1201h	Не задано верхнее положение High в уставках ПЛК (P553)
1202h	Не задано нижнее положение Low в уставках ПЛК (P553)
1D00h	Абсолютный энкодер не поддерживается
1D01h	Диапазон значений на вход „Mode“ выше или ниже допустимых (P623)

Mode

Значение	Описание
1..14	Метод контрольных точек см. P623
15	При достижении датчика контрольной точки производится реверсирование привода. После прохождения датчика контрольной точки (отрицательный фронт) он принимается за заданную точку. Поэтому заданная точка, как правило, находится со стороны датчика, на которой был начат контрольный проход. **Примечание:** В случае „перебега“ датчика контрольной точки (слишком узкий датчик, слишком высокая скорость), после прохождения датчика контрольной точки (отрицательный фронт) он также принимается за заданную точку. В этом случае контрольная точка находится не со стороны датчика, на которой был начат контрольный проход. (P623 = [15] Метод Nord 1)
16	Как и 15, но в случае «перебега» датчика контрольной точки заданная точка не принимается. Только после завершения реверсирования отрицательный фронт позволяет установить заданную точку. В этом случае контрольная точка точно находится со стороны датчика, на которой был начат контрольный проход. (P623 = [16] Метод Nord 2)
17	В случае „перебега“ датчика контрольной точки во время контрольного прохода (положительный фронт → отрицательный фронт) привод определяет среднее значение обоих положений и устанавливает их как заданную точку. Производится реверсирование и привод останавливается в определенной таким образом заданной точке. (P623 = [17] Метод Nord 3)
18..34	Метод контрольных точек см. P623

3.3.3.5 MC_MoveAbsolute

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	

Записывает уставку положения и скорости для частотного преобразователя, если **EXECUTE** меняется с 0 на 1 (фронт). Уставка частоты **VELOCITY** передается после пересчета, описываемого в MC_MoveVelocity.

POSITION:

MODE = False:

Уставка положения определяется по значению, переданному в **POSITION**.

MODE = True:

Значение, переданное в **POSITION,** соответствует индексу из параметра P613 увеличенному на 1. Положение, заданное в этом индексе параметра, соответствует уставке положения.

Пример:

Mode = True; Position = 12

ФБ переводит в положение, указанное в текущем наборе параметров P613[-13].

При достижении преобразователем уставки положения, **DONE** принимает значение 1. **DONE** удаляется при сбросе **EXECUTE**. Когда **EXECUTE** удаляется перед достижением требуемого положения, **DONE** для цикла принимает значение 1. Сигнал **BUSY** остается активным во время перемещения к уставке положения. При необходимости прерывания данной процедуры (например, другим функциональным блоком MC), используется сигнал **COMMANDABORTED**. В случает ошибки ERROR принимает значение 1, а код ошибки заносится в ERRORID. В этом случае значение **DONE** равно 0. Отрицательный фронт **EXECUTE** возвращает 0 на все выходы.

VAR_INPUT			VAR_OUTPUT		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
EXECUTE	Разблокировка	BOOL	**DONE**	Достижение заданной уставки положения	BOOL
POSITION	Уставка положения	DINT	**BUSY**	Уставка частоты не достигнута	BOOL
VELOCITY	Уставка частоты	INT	**COMMAND-ABORTED**	Прерванная команда	BOOL
MODE	Режим источника уставки положения	BOOL	**ERROR**	Ошибка в ФБ	BOOL
			ERRORID	Код ошибки	INT

ERRORID	Описание
0	Отсутствие ошибки
0x1000	ЧП не разблокирован
0x1200	Контроль положения не активирован
0x1201	Не задано верхнее положение High в уставках ПЛК (P553)
0x1202	Не задано нижнее положение Low в уставках ПЛК (P553)

Пример на ST:

```
  (* Устройство разблокируется если DIG1 = TRUE *)
  Power(Enable := _5_State_digital_input.0);
  IF Power.Status THEN
    (* Устройство разблокировано и перемещается в положение 20000 с 50% макс. частоты.
       Датчик и контроль положения для двигателя должны быть активными для выполнения данного
действия. *)
    MoveAbs(Execute := _5_State_digital_input.1, Velocity := 16#2000, Position := 20000);
  END_IF
```

3.3.3.6 MC_MoveAdditive

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	

Соответствует MC_MoveAbsolute по всем пунктам до входа **DISTANCE**. Уставка положения определяется путем добавления текущей уставки положения к значению, переданому **DISTANCE**.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
EXECUTE	Разблокировка	BOOL	**DONE**	Достижение заданной уставки положения	BOOL
DISTANCE	Уставка положения	DINT	**COMMAND-ABORTED**	Прерванная команда	BOOL
VELOCITY	Уставка частоты	INT	**ERROR**	Ошибка в ФБ	BOOL
MODE	Режим источника уставки положения	BOOL	**ERRORID**	Код ошибки	INT
			BUSY	Уставка положения не достигнута	BOOL

ERRORID	**Описание**
0	Отсутствие ошибки
1000h	ЧП не разблокирован
1200h	Контроль положения не активирован
1201h	Не задано верхнее положение High в уставках ПЛК (P553)
1202h	Не задано нижнее положение Low в уставках ПЛК (P553)

3.3.3.7 MC_MoveRelative

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	

Соответствует MC_MoveAbsolute по всем пунктам до входа **DISTANCE**. Уставка положения определяется путем добавления текущего фактического положения к значению, переданому **DISTANCE**.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
EXECUTE	Разблокировка	BOOL	**DONE**	Достижение заданной уставки положения	BOOL
DISTANCE	Уставка положения	DINT	**COMMAND-ABORTED**	Прерванная команда	BOOL
VELOCITY	Уставка частоты	INT	**ERROR**	Ошибка в ФБ	BOOL
MODE	Режим источника уставки положения	BOOL	**ERRORID**	Код ошибки	INT
			BUSY	Уставка положения не достигнута	BOOL

ERRORID	**Описание**
0	Отсутствие ошибки
1000h	ЧП не разблокирован
1200h	Контроль положения не активирован
1201h	Не задано верхнее положение High в уставках ПЛК (P553)
1202h	Не задано нижнее положение Low в уставках ПЛК (P553)

3.3.3.8 MC_MoveVelocity

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	

Применяет уставку частоты для частотного преобразователя, если **EXECUTE** меняется с 0 на 1 (фронт). При достижении преобразователем уставки частоты, **INVELOCITY** принимает значение 1. Выход **BUSY** остается активным, пока ЧП ускоряется до уставки частоты. Если **EXECUTE** уже принял значение 0, то **INVELOCITY** принимает значение 1 только на один цикл. При необходимости прерывания данной процедуры (например, другим функциональным блоком MC), используется сигнал **COMMANDABORTED**.

Отрицательный фронт **EXECUTE** возвращает 0 на все выходы.

VELOCITY пересчитывается по следующей формуле:

VELOCITY = (уставка частоты (Гц) × 0x4000) / P105

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
EXECUTE	Разблокировка	BOOL	**INVELOCITY**	Достижение заданной уставки частоты	BOOL
VELOCITY	Уставка частоты	INT	**BUSY**	Уставка частоты еще не достигнута	BOOL
			COMMAND-ABORTED	Прерванная команда	BOOL
			ERROR	Ошибка в ФБ	BOOL
			ERRORID	Код ошибки	INT

ERRORID	**Описание**
0	Отсутствие ошибки
1000ч	ЧП не разблокирован
1100h	ЧП не находится в режиме частоты вращения (активирован контроль положения)
1101h	Не настроен параметр уставки частоты (P553)

Пример на AWL:

```
CAL Power
CAL Move

LD TRUE
ST Power.Enable

(*Установить 20 Гц (макс. 50 Гц) *)
LD DINT#20
MUL 16#4000
DIV 50

DINT_TO_INT
ST Move.Velocity

LD Power.Status
ST Move.Execute
```

Пример на ST:

```
(* Прибор готов к работе если установлен DIG1 *)
Power(Enable := _5_State_digital_input.0);
IF Power.Status THEN
  (* Прибор разблокируется с 50% от макс. частоты, если установлен DIG2 *)
  MoveVelocity(Execute := _5_State_digital_input.1, Velocity := 16#2000);
END_IF
```

3.3.3.9 MC_Power

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

Данная функция предназначена для включения и выключения выходного каскада. Если вход **ENABLE** принимает значение 1, происходит разблокировка выходного каскада. Обязательным условием для этого является нахождение прибора в состоянии „Switch-on Block“ или „Ready for Switch-on“. Если прибор находится в состоянии „Fault“ или „Fault reaction active“, то сначала необходимо устранить неисправность и обработать соответствующее сообщение. Только после этого может быть выполнена разблокировка с помощью данного блока. Если прибор находится в состоянии „Not Ready for Switch-on“, то включение также не выполняется. Во всех вышеперечисленных случаях ФБ переходит в состояние ошибки, поэтому **ENABLE** следует сначала присвоить 0, чтобы обработать ошибку.

Если вход **ENABLE** принимает значение 0, происходит выключение прибора. Если двигатель при этом работает, то посредством линейного изменения, установленного с помощью параметра P103, двигатель сначала переводится на 0 Гц.

При включенном выходном каскаде прибора выходу **STATUS** соответствует значение 1, при выключенном - 0.

Значения **ERROR** и **ERRORID** сбрасываются, когда **ENABLE** переключается на 0.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
ENABLE	Разблокировка	BOOL	**STATUS**	Подача тока на двигатель	BOOL
			ERROR	Ошибка в ФБ	BOOL
			ERRORID	Код ошибки	INT

ERRORID	**Описание**
0	Отсутствие ошибки
1001ч	Активна функция остановки
1300h	Устройство не приведено в состояние „Ready for Switch-on“ или „Switch-on Block“

Пример на AWL:

```
CAL Power
CAL Move

LD TRUE
ST Power.Enable

(*Установить 20 Гц (макс. 50 Гц) *)
LD DINT#20
MUL 16#4000
DIV 50

DINT_TO_INT
ST Move.Velocity

LD Power.Status
ST Move.Execute
```

Пример на ST:

```
(* Активировать блок питания Power Block *)
Power(Enable := TRUE);
IF Power.Status THEN
  (* Устройство готово к включению *)
END_IF
```

3.3.3.10 MC_ReadActualPos

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	

Непрерывно сообщает текущее фактическое положение частотного преобразователя, если **ENABLE** соответствует значение 1. При получении действующего фактического положения на выходе, **VALID** переводится в действующее значение. В случает ошибки **ERROR** принимает значение 1, а **VALID** значение 0.

Пересчет положения: 1 оборот вала двигателя = 1000

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
ENABLE	Разблокировка	BOOL	**VALID**	Действительный выход	BOOL
			ERROR	Ошибка в ФБ	BOOL
			POSITION	Текущее факт.положение ЧП	DINT

Пример на ST:

```
ReadActualPos(Enable := TRUE);
IF ReadActualPos.Valid THEN
  Pos := ReadActualPos.Position;
END_IF
```

3.3.3.11 MC_ReadParameter

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

Выполняет циклическое считывание параметра с устройства, пока **ENABLE** равен 1. Считанный параметр заносится в Value и действует, если **DONE** равно 1. Во время процедуры чтения выход **BUSY** имеет значение 1. Пока **ENABLE** равен 1, параметр считывается циклически непрерывно. Номер параметра и индекс могут быть изменены в любое время при активном **ENABLE**. Однако при этом трудно определить, когда считано новое значение,так как сигнал **DONE** все это время равен 1. В этом случае рекомендуется установить для сигнала **ENABLE** значение 0, так как сигнал **DONE** в этом случае сбрасывается. Индекс параметра определяется по индекс в документации минус 1. Например, P700 индекс 3 („Reason for switch-on block“) соответствует индексу параметра 2. В случае ошибки **ERROR** принимает значение 1. **DONE** в этом случае равен 0, а **ERRORID** содержит код ошибки. Если сигнал **ENABLE** принимает значение 0, то все сигналы и **ERRORID** удаляются.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
ENABLE	Разблокировка	BOOL	**DONE**	Действительное значение Value	BOOL
PARAMETERNUMBER	Номер параметра	INT	**ERROR**	Ошибка процедуры чтения	BOOL
PARAMETERINDEX	Индекс параметра	INT	**BUSY**	Процедура не завершена	BOOL
			ERRORID	Код ошибки	INT
			VALUE	Считанный параметр	DINT

ERRORID	**Описание**
0	недопустимый номер параметра
3	ошибка индекса параметра
4	нет массива
201	Недействительный элемент в последнем полученном задании
202	Невозможно отобразить внутренний код отклика

Пример на ST:

```
(* Блок движения FB_ReadParameter *)
ReadParam(Enable := TRUE,Parameternumber := 102, ParameterIndex := 0);
IF ReadParam.Done THEN
  Value := ReadParam.Value;
  ReadParam(Enable := FALSE);
END_IF
```

3.3.3.12 MC_ReadStatus

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

Считывает состояние устройства. Определение состояния осуществляется на основании спецификации PLCopen „Function blocks for motion control“. Если сигналу **ENABLE** соответствует 1, происходит считывание состояния.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
ENABLE	Разблокировка	BOOL	**VALID**	Действительный выход	BOOL
			ERROR	Ошибка в ФБ	BOOL
			ERRORSTOP	Ошибка устройства	BOOL
			DISABLED	Выходной каскад устройства выключен.	BOOL
			STOPPING	Активирована команда остановки.	BOOL
			DISCRETEMOTION	Активирован один из трех ФБ позиционирования.	BOOL
			CONTINUOUSMOTION	Активировано MC_Velocity	BOOL
			HOMING	Активировано MC_Home	BOOL
			STANDSTILL	Активные команды перемещения на устройстве отсутствуют Установлено число оборотов 0 об/мин, выходной каскад включен.	BOOL

Пример на ST:

```
ReadStatus(Enable := TRUE);
IF ReadStatus.Valid THEN
  fError := ReadStatus.ErrorStop;
  fDisable := ReadStatus.Disabled;
  fStopping := ReadStatus.Stopping;
  fInMotion := ReadStatus.DiscreteMotion;
  fInVelocity := ReadStatus.ContinuousMotion;
  fInHome := ReadStatus.Homing;
  fStandStill := ReadStatus.StandStill;
end_if
```

3.3.3.13 MC_Reset

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

Сброс ошибки на устройстве (обработка неисправности), при возрастающем фронте сигнала **EXECUTE**. В случает ошибки **ERROR** принимает значение 1, а причина неисправности заносится в **ERRORID**. При отрицательном фронте **EXECUTE** выполняется сброс всех ошибок.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
EXECUTE	Запуск	BOOL	**DONE**	Сброс ошибок устройства	BOOL
			ERROR	Ошибка в ФБ	BOOL
			ERRORID	Код ошибки	INT
			BUSY	Процесс сброса еще активен	BOOL

ERRORID	**Описание**
0	Отсутствие ошибки
1001ч	Активна функция остановки
1700h	Сброс ошибки не может быть выполнен, причина не устранена.

Пример на ST:

```
Reset(Execute := TRUE);
IF Reset.Done THEN
 (* Сброс ошибки выполнен *)
 Reset(Execute := FALSE);
ELSIF Reset.Error THEN
 (* Сброс ошибки не может быть выполнен, причина не устранена *)
 Reset(Execute := FALSE);
END_IF
```

3.3.3.14 MC_Stop

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

При повышении значения фронта (с 0 на 1) устройство переводится в состояние **STANDINGSTILL**. Все активные функции движения прерываются. Выполняется торможение на 0 Гц, выходной каскад отключается. Выполнение всех остальных ФБ перемещения блокируется, пока команда остановки активна (**EXECUTE** = 1). Выход **BUSY** активируется при растущем фронте на **EXECUTE** и остается активным до получения падающего фронта на **EXECUTE**.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
EXECUTE	Запуск	BOOL	**DONE**	Команда выполнена	BOOL
			BUSY	Команда активирована	BOOL

3.3.3.15 MC_WriteParameter_16 / MC_WriteParameter_32

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

Записывает параметр 16/32 бита на устройство, когда **EXECUTE** меняется с 0 на 1 (фронт). Когда **DONE** принимает значение 1 - параметр записан. Во время процедуры чтения выход **BUSY** имеет значение 1. В случает ошибки ERROR принимает значение 1, а код ошибки заносится в ERRORID. Сигналы **DONE**, **ERROR**, **ERRORID** сохраняют свои значения до тех пор, пока **EXECUTE** снова не изменит значение на 0. При изменении значения **EXECUTE** на 0 процесс записи не прерывается. Только сигнал **DONE** сохраняет значение на 1 цикл ПЛК.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
EXECUTE	Разблокировка	BOOL	**DONE**	Действительное значение Value	BOOL
PARAMETERNUMBER	Номер параметра	INT	**BUSY**	Активирована процедура записи	BOOL
PARAMETERINDEX	Индекс параметра	INT	**ERROR**	Ошибка процедуры чтения	BOOL
VALUE	Записываемое значение	INT	**ERRORID**	Код ошибки	INT
RAMONLY	Хранить значение в RAM (версия V2.1)	BOOL			

ERRORID	**Описание**
0	недопустимый номер параметра
1	Изменение значения параметра невозможно
2	выход за пределы нижней или верхней границы диапазона значений
3	ошибка индекса параметра
4	нет массива
5	недопустимый тип данных
6	только сброс (разрешена запись только значения 0)
7	Изменение элемента описания невозможно
201	Недействительный элемент в последнем полученном задании
202	Невозможно отобразить внутренний код отклика

Пример на ST:

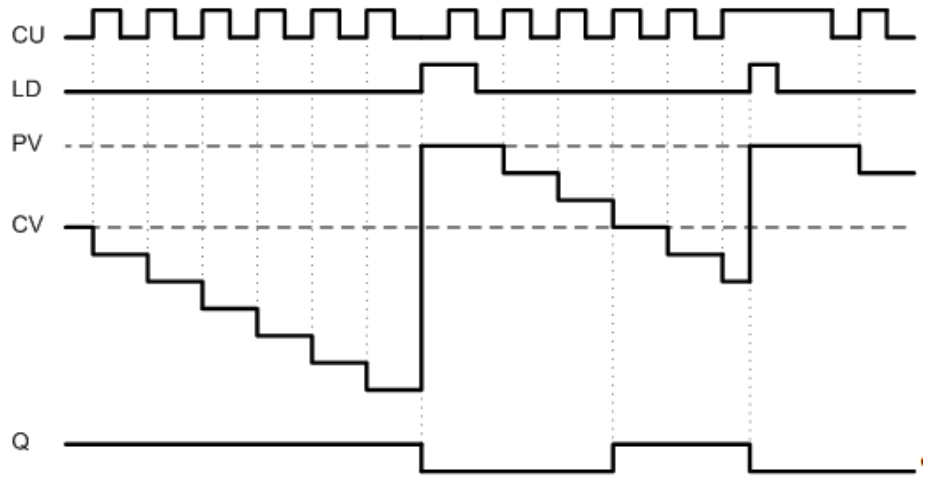
```
WriteParam16(Execute := TRUE, ParameterNumber := 102, ParameterIndex := 0, Value := 300);
  IF WriteParam16.Done THEN
    WriteParam16(Execute := FALSE);
  END_IF;
```

3.3.4 Standard

3.3.4.1 Вычитающий счетчик CTD

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

При растущем фронте на **CD** счетчик функционального блока **CV** уменьшается на единицу, пока значение CV больше -32768. Если **CV** меньше или равно 0, выход **Q** сохраняет значение TRUE. С помощью **LD** счетчику **CV** может быть присвоено значение, хранящееся в **PV**.



VAR_INPUT			VAR_OUTPUT		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
CD	Вход счетчика	**BOOL**	**Q**	TRUE, если CV <= 0	**BOOL**
LD	Загрузка начального значения	**BOOL**	**CV**	Текущее состояние счетчика	**INT**
PV	Начальное значение	**INT**			

Пример на AWL:

```
LD VarBOOL1
ST CTDInst.CD
LD VarBOOL2
ST CTDInst.LD
LD VarINT1
ST CTDInst.PV
CAL CTDInst
LD CTDInst.Q
ST VarBOOL3
LD CTDInst.CV
ST VarINT2
```

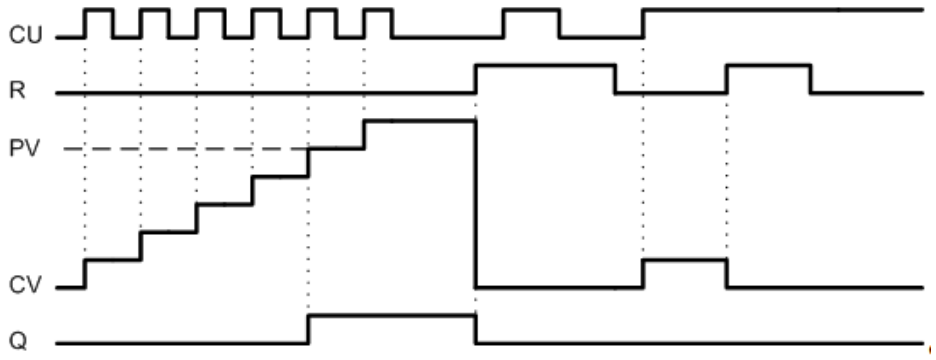
Пример на ST:

```
CTDInst(CD := VarBOOL1, LD := VarBOOL2, PV := VarINT1);
VarBOOL3 := CTDInst.Q;
VarINT2 := CTDInst.CV;
```

3.3.4.2 Суммирующий счетчик CTU

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

При растущем фронте на **CU** счетчик функционального блока **CV** увеличивается на единицу, пока **CV** не достигнет значения 32767. Если **CV** больше или равно PV, выход **Q** сохраняет значение TRUE. С помощью **R** может быть выполнен сброс счетчика **CV** на ноль.



VAR_INPUT			VAR_OUTPUT		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
CU	Вход счетчика	**BOOL**	**Q**	TRUE, если CV >= 0	**BOOL**
R	Сброс счетчика	**BOOL**	**CV**	Текущее состояние счетчика	**INT**
PV	Начальное значение	**INT**			

Пример на AWL:

```
LD VarBOOL1
ST CTUInst.CU
LD VarBOOL2
ST CTUInst.R
LD VarINT1
ST CTUInst.PV
CAL CTUInst(CU := VarBOOL1, R := VarBOOL2, PV := VarINT1)
LD CTUInst.Q
ST VarBOOL3
LD CTUInst.CV
ST VarINT2
```

Пример на ST:

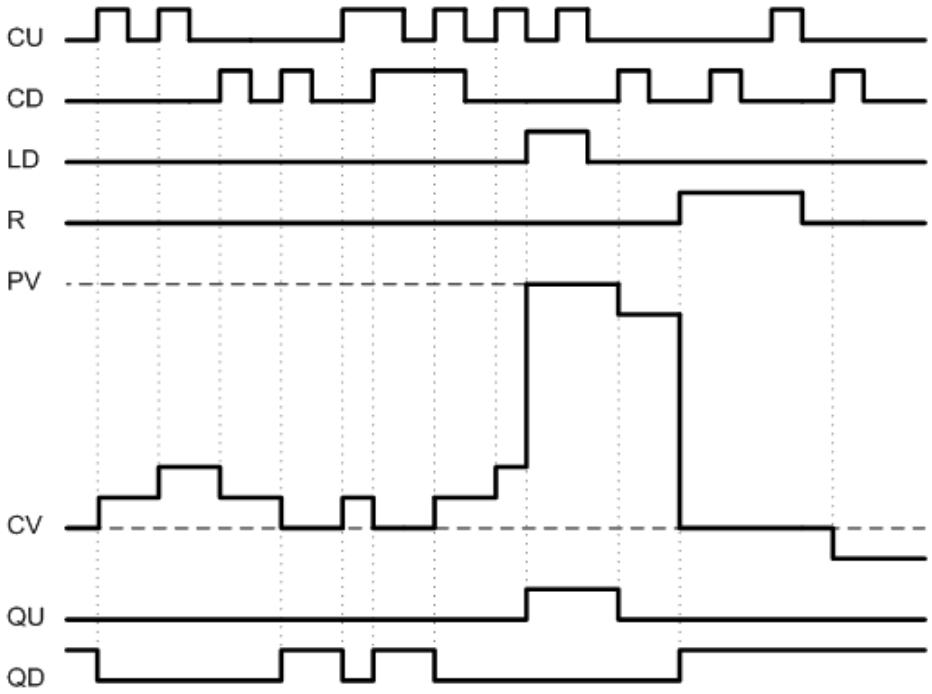
```
CTUInst(CU := VarBOOL1, R := VarBOOL2, PV := VarINT1);
VarBOOL3 := CTUInst.Q;
VarINT2 := CTUInst.CV;
```

3.3.4.3 Суммирующий и вычитающий счетчик CTUD

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

При растущем фронте на **CU** счетчик CV увеличивается на единицу, пока значение CV меньше 32767. При растущем фронте на **CD** счетчик **CV** уменьшается на единицу, пока значение CV больше -32768. С помощью **R** может быть выполнен сброс счетчика **CV** на ноль. С помощью **LD** хранящееся в **PV** значение копируется в **CV**.

R имеет преимущество перед **LD**, **CU** и **CV**. **PV** можно изменить в любое время, **QU** всегда относится к текущему установленному значению.



VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
CU	Суммирующий счетчик	BOOL	**QU**	TRUE, если CV >= PV	BOOL
CD	Вычитающий счетчик	BOOL	**QD**	TRUE, если CV <= 0	BOOL
R	Сброс счетчика	BOOL	**CV**	Текущее состояние счетчика	INT
LD	Загрузка начального значения	BOOL			
PV	Начальное значение	INT			

Пример на AWL:

```
LD VarBOOL1
ST CTUDInst.CU
LD VarBOOL3
ST CTUDInst.R
LD VarBool4
ST CTUDInst.LD
LD VarINT1
ST CTUInst.PV
CAL CTUDInst
LD CTUDInst.QU
ST VarBOOL5
LD CTUDInst.QD
ST VarBOOL5
LD CTUInst.CV
ST VarINT2
```

Пример на ST:

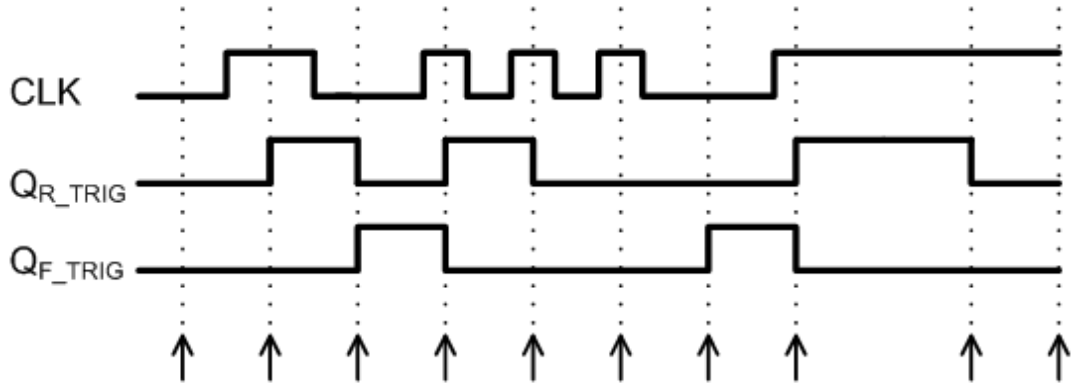
```
CTUDInst(CU:=VarBOOL1, R:=VarBOOL3, LD:=VarBOOL4, PV:=VarINT1);
VarBOOL5 := CTUDInst.QU;
VarBOOL5 := CTUDInst.QD;
VarINT2 := CTUDInst.CV;
```

3.3.4.4 R_TRIG и F_TRIG

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

Обе функции служат для распознавания фронта. При обнаружении фронта на **CLK**, **Q** принимает значение TRUE до следующего вызова функции, а после этого снова значение FALSE. Только после получения нового фронта **Q** может принять значение TRUE на цикл.

- R_TRIG = растущий фронт
- F_TRIG = падающий фронт



Вызов функции

VAR_INPUT			VAR_OUTPUT		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
CLK	Установка	**BOOL**	**Q**	Выход	**BOOL**

Пример на AWL:

```
LD VarBOOL1
ST RTRIGInst.CLK
CAL RTRIGInst
LD RTRIGInst.Q
ST VarBOOL2
```

Пример на ST:

```
RTRIGInst(CLK:= VarBOOL1);
VarBOOL2 := RTRIGInst.Q;
```

Информация

Выходной сигнал функции изменяется только при вызове функции. По этой причине целесообразно непрерывно вызывать определение фронта с помощью цикла ПЛК.

3.3.4.5 RS Flip Flop

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

Бистабильная функция, **S** устанавливает значение выхода **Q1**, а **R1** снова сбрасывает его. Если **R1** и **S** одновременно принимают значение TRUE, то приоритет имеет **R1**.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
S	Установка	BOOL	**Q1**	Выход	BOOL
R1	Сброс	BOOL			

Пример на AWL:

```
LD VarBOOL1
ST RSInst.S
LD VarBOOL2
ST RSInst.R1
CAL RSInst
LD RSInst.Q1
ST VarBOOL3
```

Пример на ST:

```
RSInst(S:= VarBOOL1, R1:=VarBOOL2);
VarBOOL3 := RSInst.Q1;
```

3.3.4.6 SR Flip Flop

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

Бистабильная функция, **S1** устанавливает значение выхода **Q1**, а **R** снова сбрасывает его. Если **R1** и **S** одновременно принимают значение TRUE, то приоритет имеет **S1**.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
S1	Установка	BOOL	**Q1**	Выход	BOOL
R	Сброс	BOOL			

Пример на AWL:

```
LD VarBOOL1
ST SRInst.S1
LD VarBOOL2
ST SRInst.R
CAL RSInst
LD SRInst.Q1
ST VarBOOL3
```

Пример на ST:

```
SRInst(S1:= VarBOOL1, R:=VarBOOL2);
VarBOOL3 := SRInst.Q1;
```

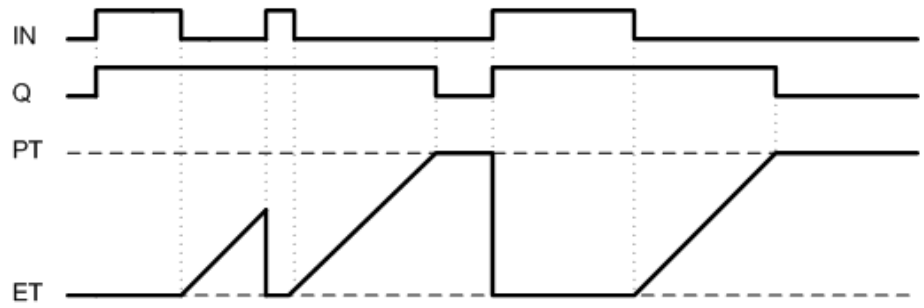
3.3.4.7 Задержка выключения TOF

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

Если **IN** = TRUE, то **Q** принимает значение TRUE. Если **IN** = FALSE, таймер увеличивается. Пока таймер работает (**ET** < **PT**), **Q** имеет значение TRUE. Если (**ET** = **PT**) - таймер останавливается, **Q** принимает значение FALSE. При новом растущем фронте в **IN**, таймер **ET** снова обнуляется.

Для упрощения ввода здесь могут использоваться литералы, например:

- LD TIME#50s20ms = 50,020 секунд
- LD TIME#1d30m = 1 день и 30 минут



VAR_INPUT			VAR_OUTPUT		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
IN	Таймер активирован	BOOL	**Q**	TRUE ß (ET<PT)	BOOL
PT	Количество времени	DINT	**ET**	Текущее состояние таймера	DINT

Пример на AWL:

```
LD VarBOOL1
ST TOFInst.IN
LD DINT#5000
ST TOFInst.PT
CAL TOFInst
LD TOFInst.Q
ST VarBOOL2
```

Пример на ST:

```
TOFInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 := TOFInst.Q;
```

Информация **Таймер ET**

Время ET идет независимо от цикла ПЛК. Запуск таймера с IN и назначение выхода Q производятся только при вызове функции „CAL“. Вызов функции осуществляется в цикле ПЛК, но в длинных программах для ПЛК он может быть больше 5 мс, что может привести к возникновению фазовой флуктуации.

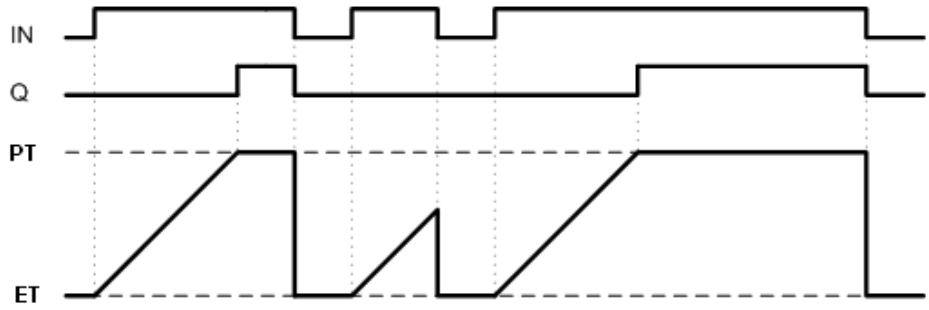
3.3.4.8 Задержка включения TON

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

Если **IN** = TRUE, таймер увеличивается. Если **ET** = PT, **Q** принимает значение TRUE, а таймер останавливается. **Q** сохраняет значение TRUE пока **IN** также имеет значение TRUE. При новом растущем фронте на **IN**, таймер запускается снова с нуля. **PT** может изменяться во время работы таймера. Продолжительность указывается в **PT** в миллисекундах. Допускается задержка по времени в диапазоне от 5мс до 24,8 дней. Минимальная задержка по времени составляет 5 мс, так как опорное время ПЛК также составляет 5 мс.

Для упрощения ввода здесь могут использоваться литералы, например:

- LD TIME#50s20ms = 50,020 секунд
- LD TIME#1d30m = 1 день и 30 минут



VAR_INPUT			VAR_OUTPUT		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
IN	Таймер активирован	BOOL	**Q**	TRUE ß (IN=TRUE & ET=PT)	BOOL
PT	Количество времени	DINT	**ET**	Текущее состояние таймера	DINT

Пример на AWL:

```
LD VarBOOL1
ST TONInst.IN
LD DINT#5000
ST TONInst.PT
CAL TONInst
LD TONInst.Q
ST VarBOOL2
```

Пример на ST:

```
TONInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 := TONInst.Q;
```

Информация **Таймер ET**

Время ET идет независимо от цикла ПЛК. Запуск таймера с IN и назначение выхода Q производятся только при вызове функции „CAL“. Вызов функции осуществляется в цикле ПЛК, но в длинных программах для ПЛК он может быть больше 5 мс, что может привести к возникновению фазовой флуктуации.

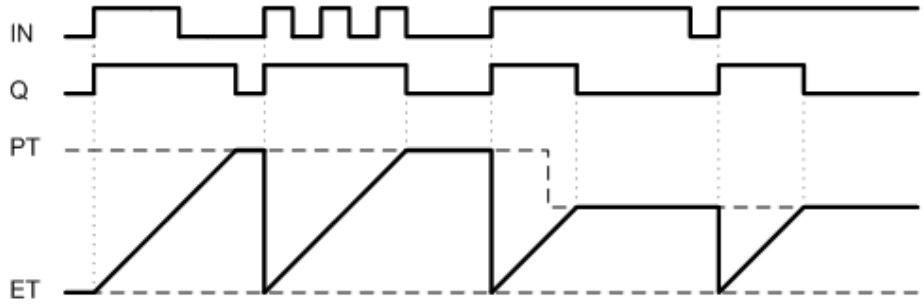
3.3.4.9 Синхроимпульс TP

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

При положительном фронте на **IN** таймер запускается со значения 0. Таймер отсчитывает время до достижения значения, указанного в **PT** и останавливается. Эта процедура не может быть прервана. Во время приращения PT может изменяться. Выход **Q** равен TRUE, пока таймер **ET** меньше **PT**. Если **ET = PT** и на **IN** зафиксирован растущий фронт, таймер снова запускается с нуля.

Для упрощения ввода здесь могут использоваться литералы, например:

- LD TIME#50s20ms = 50,020 секунд
- LD TIME#1d30m = 1 день и 30 минут



VAR_INPUT			VAR_OUTPUT		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
IN	Таймер активирован	BOOL	**Q**	TRUE ß (ET<PT)	BOOL
PT	Количество времени	DINT	**ET**	Текущее состояние таймера	DINT

Пример на AWL:

```
LD VarBOOL1
ST TPInst.IN
LD DINT#5000
ST TPInst.PT
CAL TPInst
LD TPInst.Q
ST VarBOOL2
```

Пример на ST:

```
TPInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 := TPInst.Q;
```

Информация **Таймер ET**

Время ET идет независимо от цикла ПЛК. Запуск таймера с IN и назначение выхода Q производятся только при вызове функции „CAL“. Вызов функции осуществляется в цикле ПЛК, но в длинных программах для ПЛК он может быть больше 5 мс, что может привести к возникновению фазовой флуктуации.

3.3.5 Доступ к областям памяти частотного преобразователя

При необходимости промежуточного хранения большого объема данных, их передачи или получения с других устройств следует использовать функциональные блоки FB_WriteTrace und FB_ReadTrace.

3.3.5.1 FB_ReadTrace

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X		

Данные ФБ позволяют напрямую считывать различные области памяти ЧП.

Когда ФБ определяет положительный фронт на входе **ENABLE**, происходит передача всех параметров, связанных с данным входом. **STARTINDEX** и **MEMORY** отмечают участки памяти, которые должны быть считаны. В случае успешного выполнения процедуры чтения выход **VALID** принимает значение 1 а в **VALUE** заносится считанное значение.

Если ФБ вызывается многократно, а на входе **ENABLE** сохраняется значение 1, то при каждом вызове считываемый адрес памяти увеличивается на 1, содержимое новой ячейки памяти сразу копируется на выход **VALUE**.

Текущий индекс памяти для следующего доступа может быть считан на выходе **ACTINDEX**. При достижении конца памяти выход **READY** принимает значение 1, а процедура чтения завершается.

Для считывания доступны форматы данных INT или DINT. Для значений в формате INT на выходе **VALUE** обрабатывается только нижняя часть Low. Для назначения используется вход **SIZE**: 0 соответствует формату INT, а 1 - формату DINT.

Распределение областей памяти осуществляется с помощью входа MEMORY:

MEMORY = 1 à P613[0-251] соответствует значениям 504 INT или 252 DINT

MEMORY = 0 à P900[0-247] до P906[0-111] соответствует значениям 3200 INT или 1600 DINT

Выполнение данного ФБ не может быть прервано другими блоками.

При негативном фронте на ENABLE все выходы принимают значение 0, а выполнение функции ФБ завершается.

VAR_INPUT			VAR_OUTPUT		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
ENABLE	Выполнение	BOOL	**VALID**	Процедура чтения выполнена успешно	BOOL
SIZE	Формат сохранения	BOOL	**READY**	Считан весь объем памяти	BOOL
MEMORY	Выбор области памяти	BYTE	**ERROR**	Ошибка ФБ	BOOL
STARTINDEX	Указывает на описываемую ячейку памяти	INT	**ERRORID**	Код ошибки	INT
			ACTINDEX	Текущий индекс памяти, из которого будет выполнено чтение в следующем цикле	INT
			VALUE	Считанное значение	DINT

ERRORID	**Описание**
0	Отсутствие ошибки
1A00h	Превышен диапазон значений STARTINDEX
1A01h	Превышен диапазон значений MEMORY

3.3.5.2 FB_WriteTrace

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X		

С помощью данного ФБ осуществляется промежуточное хранение отдельных значений или большого объема данных на частотном преобразователе. Хранение не является длительным, то есть при перезапуске ЧП все данные теряются.

Когда ФБ определяет положительный фронт на входе **ENABLE**, происходит передача всех параметров, связанных с данным входом. Значение, сохраненное в VALUE, записывается в ячейки памяти, отмеченные с помощью **STARTINDEX** и **MEMORY**. В случае успешного выполнения процедуры записи выход VALID принимает значение 1.

Если ФБ вызывается многократно, а на входе ENABLE сохраняется значение 1, то при каждом вызове ФБ происходит считывание и сохранение входа VALUE, а адрес памяти увеличивается на 1. Текущий индекс памяти для следующего доступа может быть считан на выходе **ACTINDEX**. При достижении конца памяти выход FULL принимает значение 1, а процедура сохранения завершается. Однако если вход **OVERWRITE** равен 1, то индекс памяти снова будет равен **STARTINDEX**, а ранее сохраненные данные будут перезаписаны.

Для сохранения доступны форматы данных INT или DINT. Для значений в формате INT на входе **VALUE** обрабатывается только нижняя часть Low. Для назначения используется вход **SIZE**: 0 соответствует формату INT, а 1 - формату DINT.

Распределение областей памяти осуществляется с помощью входа MEMORY:

MEMORY = 1 à P613[0-251] соответствует значениям 504 INT или 252 DINT

MEMORY = 0 à P900[0-247] до P906[0-111] соответствует значениям 3200 INT или 1600 DINT

Выполнение данного ФБ не может быть прервано другими блоками.

При негативном фронте на **ENABLE** все выходы принимают значение 0, а выполнение функции ФБ завершается.

VAR_INPUT			VAR_OUTPUT		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
ENABLE	Выполнение	BOOL	**VALID**	Процедура записи выполнена успешно	BOOL
SIZE	Формат сохранения	BOOL	**FULL**	Память заполнена	BOOL
OVERWRITE	Перезапись памяти	BOOL	**ERROR**	Ошибка ФБ	BOOL
MEMORY	Выбор области памяти	BYTE	**ERRORID**	Код ошибки	INT
STARTINDEX	Указывает на описываемую ячейку памяти	INT	**ACTINDEX**	Текущий индекс памяти, в который будет выполнено сохранение в следующем цикле	DINT
VALUE	Сохраняемое значение	DINT			

ERRORID	**Описание**
0	Отсутствие ошибки
1A00h	Превышен диапазон значений STARTINDEX
1A01h	Превышен диапазон значений MEMORY

Информация

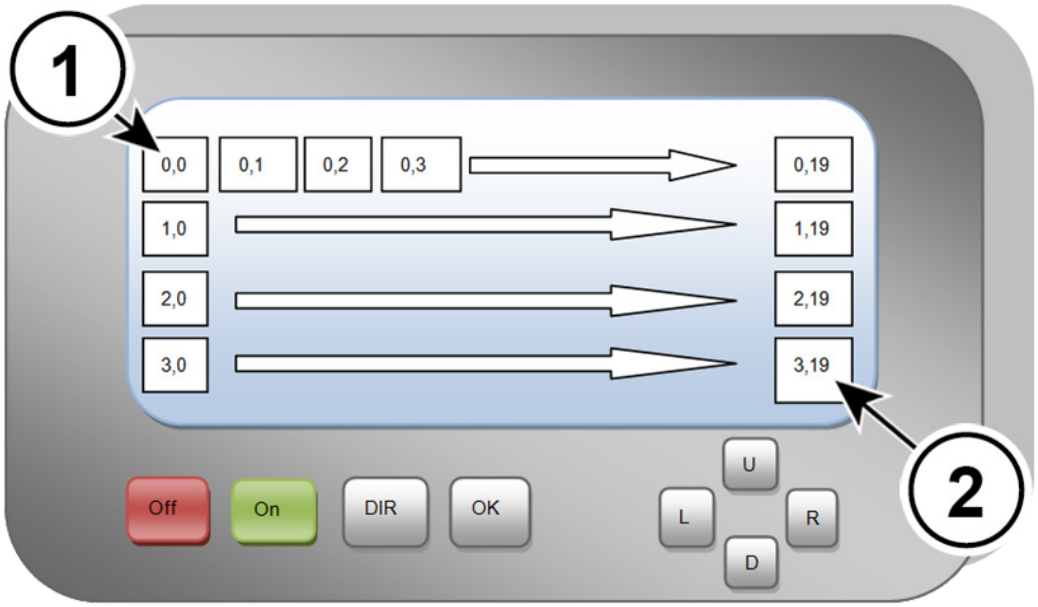
Внимание! Область памяти настройки MEMORY = 0 также используется функцией Scope. При использовании функции Scope сохраненные значения перезаписываются!

3.3.6 Модуль визуализации ParameterBox

Модуль ParameterBox позволяет использовать все содержимое дисплея для представления отдельной информации. Для этого ParameterBox следует перевести в режим визуализации. Эта опция доступна для модулей ParameterBox версии от V4.3 (параметр P1308) и реализуется следующим образом:

- В пункте меню „Display“ установить значение „PLC Display“ для параметра P1003
- Выбрать отображение рабочих значений с помощью кнопок со стрелками вправо и влево
- Индикация ПЛК активируется в P-Box и остается активной непрерывно

Режим визуализации P-Box описывается двумя ФБ для содержимого дисплея, подробно представленными далее. Предварительно следует активировать диалоговое окно конфигурирования ПЛК (кнопка ), пункт “Allow ParameterBox function modules“. Параметр „Parameterbox_key_state“ позволяет дополнительно запрашивать состояние клавиатуры модуля. За счет этого может выполняться ввод в программу ПЛК. Нижеследующее изображение описывает вид дисплея и положение задействуемых кнопок для модуля ParameterBox.



1	Первый символ	(0,0 → строка = 0, поле = 0)
2	Последний символ	(3,19 → строка = 3, поле = 19)

3.3.6.1 Обзор визуализации

Функциональный блок	Описание
FB_STRINGToPBox	Копирует строку в P-Box
FB_DINTToPBox	Копирует значение DINT в P-Box

3.3.6.2 FB_DINTToPBOX

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

Данный функциональный блок выполняет конвертацию значения в формате DINT в строку кодировки ASCII и копирует его в ParameterBox. Вывод может осуществляться в десятичном, двоичном или шестнадцатеричном формате, выбор формата осуществляется при помощи **MODE**. **ROW** и **COLUMN** используются для установки начального положения строки на дисплее модуля P-Box. Параметр **LENGTH** определяет длину строки в символах. Если **MODE** определен десятичный формат, то параметр **POINT** устанавливает запятую в отображаемом числе. В параметре **POINT** указывается сколько знаков стоит справа от запятой. Если настройка параметра равна 0, функция **POINT** отключена. Если число должно содержать больше символов, чем позволяет длина, а запятая не установлена, то превышение отображается символом „#“. Если число содержит запятую, то, при необходимости, все цифры после запятой могут быть отброшены. В шестнадцатеричном и двоичном форматах, заданных MODE, всегда будет отображаться младший разряд, если установленная длина слишком мала. Пока **ENABLE** равен 1, все изменения на входах сразу считываются. Если **VALID** равно 1, значит строка передана корректно. В случае ошибки **ERROR** принимает значение 1. В этом случае значение **VALID** равно 0. **ERRORID** содержит соответствующий код ошибки. При отрицательном фронте на **ENABLE** производится сброс значений **VALID**, **ERROR** и **ERRORID**.

Примеры:

Настройка	Отображаемое число	Индикация P-Box
Длина = 5 Точка = 0	12345	12345
Длина = 5 Точка = 0	-12345	#####
Длина = 10 Точка = 3	123456789	123456,789
Длина = 8 Точка = 3	123456789	123456,7

VAR_INPUT			VAR_OUTPUT		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
ENABLE	Передача строки	BOOL	**VALID**	Передать строку	BOOL
MODE	Формат представления 0 = десятичный 1 = двоичный 2 = шестнадцатеричный Диапазон значений = от 0 до 2	BYTE	**ERROR**	Ошибка в ФБ	BOOL
ROW	Строка дисплея Диапазон значений = от 0 до 3	BYTE	**ERRORID**	Код ошибки	INT
COLUMN	Поле дисплея Диапазон значений = от 0 до 19	BYTE			
POINT	Положение запятой Диапазон значений = от 0 до 10 0 = функция выключена	BYTE			
LENGTH	Длина вывода Диапазон значений = от 1 до 11	BYTE			
VALUE	Выводимое число	DINT			

ERRORID	**Описание**
0	Отсутствие ошибки
1500h	Строка перезаписывает область памяти массива P-Box
1501h	Превышен диапазон значений на входе LINE
1502h	Превышен диапазон значений на входе ROW
1504h	Превышен диапазон значений на входе POINT
1505h	Превышен диапазон значений на входе LENGTH
1506h	Превышен диапазон значений на входе MODE

Пример на ST:

```
(* Инициализация *)
if FirstTime then
  StringToPBox.ROW := 1;
  StringToPBox.Column := 16;
  FirstTime := False;
end_if;

(* Запрос текущего положения *)
ActPos(Enable := TRUE);
if ActPos.Valid then
  (* Отображение положения в PBox (PBox P1003 = Индикация ПЛК) *)
  DintToPBox.Value := ActPos.Position;
  DintToPBox.Column := 9;
  DintToPBox.LENGTH := 10;
  DintToPBox(Enable := True);
end_if;

(* Включение или выключение устройства через DIG1 *)
Power(Enable := _5_State_digital_input.0);
if OldState <> Power.Status then
  OldState := Power.Status;
  (* Устройство включено? *)
  if Power.Status then
    StringToPBox(Enable := False, Text := TextOn);
  else
    StringToPBox(Enable := False, Text := TextOff);
  end_if;

  StringToPBox(Enable := TRUE);
else
  StringToPBox;
end_if;
```

3.3.6.3 FB_STRINGToPBOX

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

Данный функциональный блок выполняет копирование строки (последовательности символов) в массив памяти P-Box. **ROW** и **COLUMN** используются для установки начального положения строки на дисплее модуля P-Box. Параметр **TEXT** передает требуемую строку в функциональный блок, имя строки может быть взято из таблицы переменных. Пока **ENABLE** равен 1, все изменения на входах сразу считываются. При сигнале на входе **CLEAR** все содержимое дисплея перезаписывается пробелами перед записью выбранной строки. Если **VALID** равно 1, значит строка передана корректно. В случае ошибки **ERROR** принимает значение 1. В этом случае значение **VALID** равно 0. **ERRORID** содержит соответствующий код ошибки. При отрицательном фронте на **ENABLE** производится сброс значений **VALID**, **ERROR** и **ERRORID**.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
ENABLE	Передача строки	BOOL	**VALID**	Передать строку	BOOL
CLEAR	Очистить дисплей	BOOL	**ERROR**	Ошибка в ФБ	BOOL
ROW	Строка дисплея Диапазон значений = от 0 до 3	BYTE	**ERRORID**	Код ошибки	INT
COLUMN	Поле дисплея Диапазон значений = от 0 до 19	BYTE			
TEXT	отображаемый текст	STRING			

ERRORID	**Описание**
0	Отсутствие ошибки
1500h	Строка перезаписывает область памяти массива P-Box
1501h	Превышен диапазон значений на входе ROW
1502h	Превышен диапазон значений на входе COLUMN
1503h	Выбранный номер строки не существует
1506h	В конфигурации ПЛК не активирована опция „Allow ParameterBox function modules“.

Пример на ST:

```
(* Инициализация *)
if FirstTime then
  StringToPBox.ROW := 1;
  StringToPBox.Column := 16;
  FirstTime := False;
end_if;

(* Запрос текущего положения *)
ActPos(Enable := TRUE);
if ActPos.Valid then
  (* Отображение положения в PBox (PBox P1003 = Индикация ПЛК) *)
  DintToPBox.Value := ActPos.Position;
  DintToPBox.Column := 9;
  DintToPBox.LENGTH := 10;
  DintToPBox(Enable := True);
end_if;

(* Включение или выключение устройства через DIG1 *)
Power(Enable := _5_State_digital_input.0);
if OldState <> Power.Status then
  OldState := Power.Status;
  (* Устройство включено? *)
  if Power.Status then
    StringToPBox(Enable := False, Text := TextOn);
  else
    StringToPBox(Enable := False, Text := TextOff);
  end_if;

  StringToPBox(Enable := TRUE);
else
  StringToPBox;
end_if;
```

3.3.7 FB_Capture (Получение быстрых результатов)

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X		

Время цикла ПЛК составляет 5 мс, если обработка внешних результатов производится очень быстро, такая длина цикла слишком велика. ФБ Capture позволяет считывать определенные физические величины по фронтам сигналов на входах ЧП. Контроль входов производится циклами продолжительностью 1 мс. Сохраненные таким образом значения потом могут считываться ПЛК.

При положительном фронте на **EXECUTE** производится считывание на всех входах и включение функции Capture. Контролируемый вход ЧП выбирается через вход **INPUT**. С помощью **EDGE** выбирается тип фронта и характер поведения функционального блока.

EDGE = 0 При первом положительном фронте выбранное значение сохраняется в **OUTPUT1**, а **DONE1** принимает значение 1. При следующем положительном фронте значение сохраняется в **OUTPUT2**, **DONE2** принимает значение 1. После этого функциональный блок отключается.

EDGE = 1 Поведение аналогично **EDGE** = 0, однако операция производится по отрицательному фронту.

EDGE = 2 При первом положительном фронте выбранное значение сохраняется в **OUTPUT1**, а **DONE1** принимает значение 1. При следующем отрицательном фронте выполняется сохранение в **OUTPUT2**, **DONE2** принимает значение 1. После этого функциональный блок отключается.

EDGE = 3 Поведение аналогично **EDGE** = 2, но операция выполняется сначала при отрицательном, а затем при положительном фронте.

Если вход **CONTINUOUS** имеет значение 1, функция **EDGE** используется только с настройками 0 и 1. Функциональный блок непрерывно работает и сохраняет последнее событие всегда в **OUTPUT1**. Начиная с первого события **DONE1** остается активным. **DONE2** и **OUTPUT2** не используются.

Выход **BUSY** остается активным, пока не наступят оба события Capture (**DONE1** и **DONE2**).

Работа функционального блока может быть прекращена в любое время посредством отрицательного фронта на **EXECUTE**. При этом все выходы сохраняют свои значения. Положительный фронт на **EXECUTE** сначала удаляет всю информацию на всех выходах, а затем запускает функциональный блок.

VAR_INPUT			VAR_OUTPUT		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
EXECUTE	Выполнение	BOOL	**DONE1**	Действительное значение на OUTPUT1	BOOL
CONTINUOUS	Однократное выполнение или непрерывный режим.	BOOL	**DONE2**	Действительное значение на OUTPUT2	BOOL
INPUT	**SK54xE** Контролируемый вход 0 = вход 1 - - - - 7 = вход 8 **SK52xE, SK53xE, SK2xxE, SK2xx-EFDS** Контролируемый вход 0 = вход 1 - - - - 3 = вход 4	BYTE	**BUSY**	Функциональный блок ожидает событие Capture	BOOL
EDGE	Инициирующий фронт	BYTE	**ERROR**	Ошибка ФБ	BOOL
SOURCE	Сохраняемая величина 0 = положение в оборотах 1 = факт. частота 2 = момент	BYTE	**ERRORID**	Код ошибки	INT
			OUTPUT1	Значение для 1го события Capture	DINT
			OUTPUT2	Значение для 2го события Capture	DINT

ERRORID	Описание
0	Отсутствие ошибки
1900h	Превышен диапазон значений INPUT.
1901h	Превышен диапазон значений EDGE.
1902h	Превышен диапазон значений SOURCE.
1903h	Активно более двух экземпляров объекта

Пример на ST:

```
Power(ENABLE := TRUE);
IF Power.STATUS THEN
    Move(EXECUTE := TRUE, POSITION := Pos, VELOCITY := 16#2000);
    (* Capture ожидает сигнала High на DIG1. При его обнаружении функциональный блок сохраняет в
памяти текущее положение. Значение опрашивается с помощью свойства "OUTPUT1". *)
    Capture(EXECUTE := TRUE, INPUT := 0);

    IF Capture.DONE1 THEN
      Pos := Capture.OUTPUT1;
      Move(EXECUTE := FALSE);
    END_IF;
END_IF;
```

Информация

Этот функциональный блок допускает несколько экземпляров объекта в программе ПЛК. Но одновременно могут быть активными только два из них!

3.3.8 FB_DinCounter

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	ab V2.3	ab V3.1	ab V2.1	X	ab V1.1	

Данный ФБ служит для подсчета импульсов через цифровые входы. Учитывается каждый фронт (Low – High и High – Low). Минимальная ширина импульса составляет 250мкс.

Для активации ФБ используется ENABLE. При положительном фронте происходит передача входов PV, UD, DIN и MODE, а все выходы сбрасываются.

UD определяет направление счета

- 0 = больший подсчет
- 1 = меньший подсчет

Значение счетчика может заноситься в PV. В зависимости от настройки входа MODE он действует по-разному.

MODE

- 0 = переполнение, счетчик работает как непрерывный счетчик. Переполнение может быть в положительном и отрицательном направлении. При запуске функции назначается CV = PV. В этом режиме BUSY всегда равен 1, а Q всегда 0.
- 1 = без переполнения
  - Счетчик прямого действия à CV запускается при 0, BUSY = 1, выполняется до CV=>PV. BUSY принимает значение 0, а Q равен 1. Процедура счета останавливается.
  - счетчик обратного действия à CV запускается с PV и выполняется до CV<=0. Все это время BUSY = 1 и принимает значение 0, когда достигается конец подсчета. Q, наоборот, принимает значение 1.
  - Новый запуск счетчика производится при появлении нового фронта на входе ENABLE

DIN определяет вход измерений. Количество входов зависит от соответствующего ЧП (до 4 штук).

- Вход 1 = 0
- Вход 2 = 1
- Вход 3 = 2
- Вход 4 = 3

VAR_INPUT			VAR_OUTPUT		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
ENABLE	Разблокировка	BOOL	**Q**	Подсчет окончен	BOOL
UD	Направление подсчета 0 = больший подсчет 1 = меньший подсчет	BOOL	**BUSY**	Счетчик работает	BOOL
PV	Значение счетчика	INT	**ERROR**	Ошибка ФБ	BOOL
MODE	Режим	BYTE	**ERRORID**	Код ошибки	INT
DIN	Измерительный вход	BYTE	**CV**	Значение счетчика	INT
			CF	Частота счета (разрешение 0,1)	INT

ERRORID	**Описание**
0	Отсутствие ошибки
0x1E00	Цифровой вход уже используется другим счетчиком
0x1E01	Цифровой вход не существует
0x1E02	Превышен диапазон значений MODE

3.3.9 FB_FunctionCurve

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	

Функциональный блок обеспечивает управление характеристиками. На него могут передаваться определенные точки, по которым он эмулирует функцию. Поведение выхода определяется соответствующей характеристикой. По отдельным опорным точкам выполняется линейная интерполяция. Опорные точки определяются значениями X и Y. Значения X при этом всегда имеют тип **INT**, а значения Y могут иметь тип **INT** или **DINT**, в зависимости от величины наибольшей опорной точки. При использовании типа **DINT** он занимает больше памяти. Опорные точки указываются в окне переменных в разделе "Init Value“. При получении на входе **ENABLE** значения TRUE, выполняется расчет соответствующего выходного значения **OUTVALUE** на основании входного значения **INVALUE**. Значение TRUE на **VALID** сигнализирует о том, что получено действительное выходное значение **OUTVALUE**. Пока **VALID** имеет значение FALSE, выходу **OUTVALUE** соответствует значение 0. Если входное значение **INVALUE** выходит за пределы верхнего или нижнего конца характеристики, на выходе сохраняется первое или последнее выходное значение характеристики, пока **INVALUE** не вернется снова в диапазон характеристики. При выходе за верхний или нижний пределы характеристики соответствующий выход **MINLIMIT** или **MAXLIMIT** принимает значение TRUE. **ERROR** принимает значение TRUE, если абсцисса (значение X) характеристики не увеличивается, либо если таблица не инициализирована. Соответствующая ошибка также указывается посредством **ERRORID**, а выходное значение равно 0. Ошибка сбрасывается когда **ENABLE** = FALSE.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
ENABLE	Выполнение	BOOL	**VALID**	Действительное выходное значение	BOOL
INVALUE	Входное значение (x)	INT	**ERROR**	Ошибка в ФБ	BOOL
			ERRORID	Код ошибки	INT
			MAXLIMIT	Достигнут макс.предел	BOOL
			MINLIMIT	Достигнут мин.предел	BOOL
			OUTVALUE	Выходное значение (y)	DINT

ERRORID	**Описание**
0	Отсутствие ошибки
1400h	Абсцисса (значение X) характеристики не возрастает
1401h	Характеристика не инициализирована

3.3.10 FB_PIDT1

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

P-I-DT1 представляет собой свободно параметрируемый дискретный регулятор. Если отдельные составляющие, P, I или DT1 - составляющие, не используются, то данный параметр описывается значением 0. Составляющая T1 работает только вместе с составляющей D. То есть регулятор PT1 не доступен для параметрирования. Внутренние ограничения памяти ограничивают параметры регулирования следующими диапазонами.

Допустимые диапазоны значений для параметров регулирования.			
Параметр	**Диапазон значений**	**Пересчет**	**полученный диапазон значений**
P (Kp)	0 – 32767	1/100	0,00 – 32,767
I (Ki)	0 – 10240	1/100	0,00 – 10,240
D (Kd)	0 – 32767	1/1000	0,000 – 3,2767
T1 (ms)	0 – 32767	1/1000	0,000 – 3,2767
Max	-32768 – 32767		
Min	-32768 – 32767		

Регулятор запускается когда **ENABLE** принимает значение TRUE. Передача параметров регулирования производится только при растущем фронте **ENABLE**. Пока **ENABLE** равен TRUE, изменения параметров регулирования не имеют действия. Если **ENABLE** принимает значение FALSE, на выходе сохраняется последнее значение.

Выходной бит **VALID** используется, пока выходное значение Q изменяется в пределах диапазона минимального и максимального значений, а вход **ENABLE** равен TRUE.

При появлении ошибки срабатывает **ERROR**. Тогда бит **VALID** принимает значение FALSE, а причина ошибки указывается посредством **ERRORID** (см. таблицу внизу).

Если бит **RESET** принимает TRUE, содержимое интегратора и дифференциатора меняется на 0. Если вход **ENABLE** принимает значение FALSE, то выход **OUTPUT** принимает значение 0. Если вход **ENABLE** принимает значение TRUE, работает только P-составляющая выхода **OUTPUT**.

Если значение на выходе **OUTPUT** больше максимального или меньше минимального выходного значения, используется соответствующий бит **MAXLIMIT** или **MINLIMIT**, а бит **VALID** принимает значение FALSE.

Информация

Если программа не может быть обработана в рамках одного цикла, регулятор рассчитывает выходное значение второй раз со старым результатом считывания. За счет этого достигается постоянная скорость считывания. По этой причине необходимо, чтобы команда CAL выполнялась для регулятора PIDT1 в каждом цикле ПЛК и только в конце программы ПЛК!

RU

VAR_INPUT			VAR_OUTPUT		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
ENABLE	Выполнение	BOOL	**VALID**	Действительное выходное значение	BOOL
RESET	Восстановить выходное значение	BOOL	**ERROR**	Ошибка в ФБ	BOOL
P	P-составляющая (Kp)	INT	**ERRORID**	Код ошибки	INT
I	I-составляющая (Ki)	INT	**MAXLIMIT**	Достигнут макс.предел	BOOL
D	D-составляющая (Kd)	INT	**MINLIMIT**	Достигнут мин.предел	BOOL
T1	T1-составляющая в мс	INT	**OUTPUT**	Выходное значение	INT
MAX	Макс.выходное значение	INT			
MIN	Мин.выходное значение	INT			
SETPOINT	Уставка	INT			
VALUE	Фактическое значение	INT			

ERRORID	Описание
0	Отсутствие ошибки
1600h	P-составляющая выходит за пределы диапазона значений
1601h	I-составляющая выходит за пределы диапазона значений
1602h	D-составляющая выходит за пределы диапазона значений
1603h	T1-составляющая выходит за пределы диапазона значений

3.3.11 FB_ResetPostion

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	ab V2.3	ab V3.1	ab V2.1	X	ab V1.2	

При фронте на входе **EXECUTE**, текущее положение принимает заданное значение. Для абсолютных энкодеров текущее положение может сбрасываться только на значение 0. Значение положения не используется.

VAR_INPUT			**VAR_OUTPUT**		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
EXECUTE	Выполнение	BOOL			
Position	Положение	DINT			

3.3.12 FB_Weigh

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	от V2.3	от V3.1	от V2.1	X	от V1.2	

Функциональный блок служит для определения среднего момента вращения при перемещении с постоянным количеством оборотов. На основании данного значения могут определяться другие физические величины, например, перемещаемый вес.

ФБ запускается при положительном фронте на **EXECUTE**. Фронт запускает передачу всех входов в ФБ. ЧП движется со скоростью вращения, заданной параметром **SPEED**. Измерение начинается при достижении времени, установленного параметром **STARTTIME**. Продолжительность измерения определяется **MEASURETIME**. ЧП останавливается после окончания времени измерения. Если вход **REVERSE** = 1, то процедура измерения запускается заново, но с инвертированным числом оборотов. В противном случае измерение прекращается, выход **DONE** принимает значение 1, а результат измерений заносится в VALUE.

BUSY активен на всем протяжении процедуры измерений.

Пересчет результатов измерений в **VALUE** выполняется по формуле 1 = 0,01% от номинального момента вращения двигателя.

Вызов другого ФБ управления движением останавливает выполнение функции измерения, а выход **ABORT** принимает значение 1.

При новом положительном фронте на **EXECUTE** все остальные выходы ФБ сбрасываются.

VAR_INPUT			VAR_OUTPUT		
Вход	**Описание**	**Тип**	**Выход**	**Описание**	**Тип**
EXECUTE	Выполнение	BOOL	**DONE**	Окончание измерений	BOOL
REVERSE	Изменение направления вращения	BOOL	**BUSY**	Выполнение измерений	BOOL
STARTTIME	Время до начала измерений в мс	INT	**ABORT**	Прерывание измерений	BOOL
MEASURETIME	Время измерений в мс.	INT	**ERROR**	Ошибка ФБ	BOOL
SPEED	Скорость измерений в % (нормированная по максимальной частоте, 16#4000 соотв. 100%)	INT	**ERRORID**	Код ошибки	INT
			VALUE	Результат измерений	INT

ERRORID	Описание
0	Отсутствие ошибки
0x1000	Частотный преобразователь не включен
0x1101	Уставка частоты не задана в качестве уставки параметра (P553)
0x1C00	Превышен диапазон значений STARTTIME
0x1C01	Превышен диапазон значений MEASURETIME
0x1C02	Погрешность измеренных значений относительно друг друга больше 1/8

Пример на ST:

```
(* Разблокировать прибор *)
Power(Enable := TRUE);
(* Прибор разблокирован? *)
if Power.Status then
  (* Задать время запуска 2000 мс *)
  Weigh.STARTTIME := 2000;
  (* Задать время измерений 1000 мс *)
  Weigh.MEASURETIME := 1000;
  (* Задать скорость 25% от максимальной *)
  Weigh.SPEED := 16#1000;
end_if;

Weigh(EXECUTE := Power.Status);
(* Взвешивание окончено? *)
if Weigh.done then
  Value := Weigh.Value;
end_if;
```

Информация

Этот ФБ допускает только один экземпляр объекта в программе ПЛК!

3.4 Операторы

3.4.1 Арифметические операторы

Информация

Некоторые из нижеследующих операторов могут содержать дополнительные команды. Они указываются в скобках после оператора. При этом следует обратить внимание на то, что после открытой скобки должен стоять пробел. Закрывающая скобка устанавливается в отдельную программную строку.

```
LD Var1
ADD( Var2
SUB Var3
)
```

3.4.1.1 ABS

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных			X	X

Создает абсолютное значение из Akku.

Пример на AWL:

```
LD -10 (* Загружает значение -10 *)
ABS (* Akku = 10 *)
ST Value1 (* Сохраняет значение 10 в Value1 *)
```

Пример на ST:

```
Value1 := ABS(-10); (* Результат равен 10 *)
```

3.4.1.2 ADD и ADD(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Складывает переменные и константы с учетом знака. Первое значение находится в Akku, второе загружается по команде ADD, либо находится внутри скобок. Команда ADD может применяться к нескольким переменным или константам. При сложении со скобками Akku прибавляется к результату выражения в скобках. Допускается использование до 6 уровней скобок. Слагаемые величины должны относится к одному типу переменных.

Пример на AWL:

```
LD 10
ADD 204 (* Сложение двух констант *)
ST Value
LD 170 (* Сложение одной константы и 2 переменных. *)
ADD Var1, Var2 (* 170dez + Var1 + Var2 *)
ST Value
LD Var1
ADD( Var2
SUB Var3 (* Var1 + ( Var2 - Var3 ) *)
)
ST Value
```

Пример на ST:

```
Результат := 10 + 30; (* Результат равен 40 *)
Результат := 10 + Var1 + Var2;
```

3.4.1.3 DIV и DIV(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Делит значение в Akku на операнд. При делении на ноль в Akku заносится максимальный возможный результат, например при делении на значения типа INT это будет значение 0x7FFF, а если делитель отрицательный, то значение равно 0x8000. При делении со скобками Akku делится на результат выражения в скобках. Допускается использование до 6 уровней скобок. Делимое и делитель должны относится к одному типу переменных.

Пример на AWL:

```
LD 10
DIV 3 (* Деление двух констант *)
ST iValue (* Результат равен 9 *)
LD 170 (* Деление одной константы и 2 переменных. *)
DIV Var1, Var2 (* (170dez : Var1) : Var2 *)
ST Value
LD Var1 (* Деление Var1 на содержимое скобок*)
DIV( Var2
SUB Var3
) (* Var1 : ( Var2 - Var3 ) *)
ST Value
```

Пример на ST:

```
Результат := 30 / 10; (* Результат равен 3 *)
Результат := 30 / Var1 / Var2;
```

3.4.1.4 LIMIT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Команда ограничивает хранящееся в Akku значение полученными минимальным и максимальным значениями. Значения. Если значение больше максимального в Akku заносится максимальное, если меньше минимального - минимальное. Если значение не выходит за границы диапазонов, оно не изменяется.

Пример на AWL:

```
LD 10 (* Загружает значение 10 в регистр Akku *)
LIMIT 20, 30 (* Значение сравнивается с границами 20 и 30. *)
(* Значение в Akku меньше, перезаписывается значение 20 в Akku *)
ST iValue (* Сохраняет значение 20 в Value1 *)
```

Пример на ST:

```
Результат := Limit(10, 20, 30); (* Результат равен 20 *)
```

3.4.1.5 MAX

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Определяет максимальную из двух переменных или констант. При этом текущее содержимое накопительного регистра сравнивается со значением отправленным команде MAX. После выполнения команды в регистре сохраняется максимальное из двух значений. Обе величины должны относится к одному типу переменных.

Пример на AWL:

```
LD 100 (* Загрузка значения 100 в Akku *)
MAX 200 (* Сравнение со значением 200 *)
ST iValue (* Сохранение 200 в Value2 (т.к. это значение больше) *)
```

Пример на ST:

```
Результат := Max(100, 200); (* Результат равен 200 *)
```

3.4.1.6 MIN

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Определяет минимальную из двух переменных или констант. При этом текущее содержимое накопительного регистра сравнивается со значением отправленным команде MIN. После выполнения команды в регистре сохраняется минимальное из двух значений. Обе величины должны относится к одному типу переменных.

Пример на AWL:

```
LD 100 (* Загрузка значения 100 в Akku *)
MIN 200 (* Сравнение со значением 200 *)
ST Value2 (* Сохранение 100 в Value2 (т.к. это значение меньше) *)
```

Пример на ST:

```
Результат := Min(100, 200); (* Сохранение 100 в Value2 (т.к. это значение меньше) *)
```

3.4.1.7 MOD и MOD(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Выполняется деление Akku на одну или несколько переменных или констант, остаток от деления сохраняется в качестве результата в Akku. При использовании скобок модуля содержимое Akku делится на результат выражения в скобках и берется по модулю. Допускается использование до 6 уровней скобок.

Пример на AWL:

```
LD 25 (* Загрузка делимого *)
MOD 20 (* Деление 25/20 по модулю = 5 *)
ST Var1 (* Сохранение результата 5 в Var1 *)
LD 25 (* Загрузка делимого *)
MOD( Var1 (* Результат = 25/(Var1 + 10) по модулю в Akku *)
ADD 10
)
ST Var3 (* Сохранение результата 10 в Var3 *)
```

Пример на ST:

```
Результат := 25 MOD 20; (* Сохранение результата 5 в Var1 *)
Результат := 25 MOD (Var1 + 10); (* Результат = 25/(Var1 + 10) по модулю в Akku *)
```

3.4.1.8 MUL и MUL(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Умножает содержимое Akku на одну или несколько переменных или констант. При умножении со скобками Akku умножается на результат выражения в скобках. Допускается использование до 6 уровней скобок. Обе величины должны относится к одному типу переменных.

Пример на AWL:

```
LD 25 (* Загрузка множителя *)
MUL Var1, Var2 (* 25 * Var1 * Var2 *)
ST Var2 (* Сохранение результата *)
LD 25 (* Загрузка множителя *)
MUL( Var1 (* Результат = 25*(Var1 + Var2) *)
ADD Var2
ST Var3 (* Сохранение результата в Var3 *)
)
```

Пример на ST:

```
Результат := 25 * Var1 * Var2;
Результат := 25 * (Var1 + Var2);
```

3.4.1.9 MUX

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

С помощью одного индекса, расположенного в Akku перед командой, может осуществляться выбор различных констант или переменных. Первому значению соответствует Индекс 0. Выбранное значение загружается в Akku. Количество значений ограничивается только размером программной памяти.

Пример на AWL:

```
LD 1 (* Выбор нужного элемента *)
MUX 10,20,30,40,Value1 (* Команда MUX для 4 констант и одной переменной *)
ST Value (* Сохраняемый Результат = 20 *)
```

Пример на ST:

```
Результат := Mux(1, 10, 20, 30, 40, Value1) (* Сохраняемый Результат = 20 *)
```

3.4.1.10 SUB и SUB(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Вычитает одну или несколько переменных или констант из значения в Akku. При вычитании со скобками из Akku вычитается результат выражения в скобках. Допускается использование до 6 уровней скобок. Вычитаемые и уменьшаемые величины должны относится к одному типу переменных.

Пример на AWL:

```
LD 10
SUB Var1 (* Результат = 10 - Var1 *)
Результат ST
LD 20
SUB Var1, Var2, 30 (* Результат = 20 - Var1 - Var12 - 30 *)
Результат ST
LD 20
SUB( 6 (* Вычитание содержимого скобок из 20 *)
AND 2
) (* Результат = 20 - (6 AND 2) *)
Результат ST (* Результат = 18 *)
```

Пример на ST:

```
Результат := 10 - Value1;
```

3.4.2 Расширенные математические операторы

Информация

Описанные далее операторы требуют большого объема вычислений. Их использование может привести к существенному увеличению времени выполнения программы для ПЛК.

3.4.2.1 COS, ACOS, SIN, ASIN, TAN, ATAN

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X		

	BOOL	BYTE	INT	DINT
Тип данных				X

Вычисление соответствующей математической функции. Рассчитываемая величина должна находится в Akku в дуговых минутах. Пересчет выполняется по формуле 1 = 1000.

Пересчет: Угол в дуговом измерении = (угол в градусах * PI / 180)*1000 например, угол 90° пересчитывается следующим образом: 90° * 3.14 / 180) *1000 = 1571

$$AE = \sin\left(\frac{AE}{1000}\right) \cdot 1000 \qquad AE = \cos\left(\frac{AE}{1000}\right) \cdot 1000 \qquad AE = \tan\left(\frac{AE}{1000}\right) \cdot 1000$$

Пример на AWL:

```
LD 1234
SIN
Результат ST (* Результат = 943 *)
```

Пример на ST:

```
Результат := COS(1234); (* Результат = 330 *)
Результат := ACOS(330); (* Результат = 1234 *)
Результат := SIN(1234); (* Результат = 943 *)
Результат := ASIN(943); (* Результат = 1231 *)
Результат := TAN(999); (* Результат = 1553 *)
Результат := ATAN(1553); (* Результат = 998 *)
```

3.4.2.2 EXP

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X		

	BOOL	BYTE	INT	DINT
Тип данных				X

Создает из Akku экспоненциальную функцию числа Эйлера (2,718). После запятой может быть указано 3 знака, то есть число 1,002 вводится как 1002.

$$AE = e^{\left(\frac{AE}{1000}\right)} \cdot 1000$$

Пример на AWL:

```
LD 1000
EXP
Результат ST (* Результат = 2718 *)
```

Пример на ST:

```
Результат := EXP(1000); (* Результат = 2718 *)
```

3.4.2.3 LN

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X		

	BOOL	BYTE	INT	DINT
Тип данных				X

Логарифм по основанию e (2,718). После запятой может быть указано 3 знака, то есть число 1,000 вводится как 1000.

$$AE = \ln\left(\frac{AE}{1000}\right) \cdot 1000$$

Пример на AWL:

```
LD 1234
LN
Результат ST
```

Пример на ST:

```
Результат := LN(1234); (* Результат = 210 *)
```

3.4.2.4 LOG

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X		

	BOOL	BYTE	INT	DINT
Тип данных				X

Создает из Akku логарифм по основанию 10. После запятой может быть указано 3 знака, то есть число 1,000 вводится как 1000.

$$AE = log_{10}\left(\frac{AE}{1000}\right) \cdot 1000$$

Пример на AWL:

```
LD 1234
LOG
Результат ST (* Результат = 91 *)
```

Пример на ST:

```
Результат := LOG(1234); (* Результат = 91 *)
```

3.4.2.5 SQRT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X		

	BOOL	BYTE	INT	DINT
Тип данных				X

Вычисляет квадратный корень из Akku. После запятой может быть указано 3 знака, то есть число 1,000 вводится как 1000.

$$AE = \sqrt{\left(\frac{AE}{1000}\right)} \cdot 1000$$

Пример на AWL:

```
LD 1234
SQRT
Результат ST (* Результат = 1110 *)
```

Пример на ST:

```
Результат := SQRT(1234); (* Результат = 1110 *)
```

3.4.3 Операции над битами

3.4.3.1 AND и AND(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных	X	X	X	X

Побитовая связь И значения AE/Akku с одной или двумя переменными или константами. Побитовая связь "И" (…) содержимого AE/Akku и содержимого AE/Akku, предварительно полученного в скобках. Допускается использование до 6 уровней скобок. Все величины должны относится к одному типу переменных.

Пример на AWL:

```
LD 170
AND 204 (* AND связь между двумя константами *)
(* Akku = 136 (см. пример под таблицей) *)

LD 170 (* Связь между одной константой и 2 переменными.*)
AND Var1, Var2 (* Akku = 170dez AND Var1 AND Var2 *)

LD Var1
AND ( Var2 (* AE/Akku = Var1 AND ( Var2 OR Var3 ) *)
OR Var3
)
```

Пример на ST:

```
Результат := 170 AND 204; (* Результат = 136dez *)
```

Var2	Var1	Результат
0	0	0
0	1	0
1	0	0
1	1	1

Пример: 170dez (1010 1010bin) AND 204dez (1100 1100bin) = (1000 1000bin) 136dez

3.4.3.2 ANDN и ANDN(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных	X	X	X	X

Побитовая связь И значения AE/Akku с инвертированным операндом. Побитовая связь "И" (…) с AE/Akku и инвертированным результатом в скобках. Допускается использование до 6 уровней скобок. Связываемые величины должны относится к одному типу переменных.

Пример на AWL:

```
LD 2#0000_1111
ANDN 2#0011_1010 (* ANDN связь между двумя константами *)
(* Akku = 2#1111_0101 *)

LD 170 (* Связь между одной константой и 2 переменными *)
ANDN Var1, Var2 (* Akku = 170d ANDN Var1 ANDN Var2 *)

LD Var1
ANDN ( Var2 (* AE/Akku = Var1 ANDN ( Var2 OR Var3 ) *)
OR Var3
)
```

Var2	Var1	Результат
0	0	1
0	1	1
1	0	1
1	1	0

Пример: 170dez (1010 1010bin) AND 204dez (1100 1100bin) = (1000 1000bin) 136dez

3.4.3.3 NOT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных	X	X	X	X

Побитовая инверсия Akku.

Пример на AWL:

```
LD BYTE#10 (* Загрузка десятеричного значения в AKKU в формате Byte *)
NOT (* Значение переводится на уровень битов (0000 1010), *)
(* Выполняется побитовая инверсия (1111 0101) и обратная конвертация в десятичное значение*)
(* результат преобразования = 245дес.*)
ST Var3 (* Сохранение результата в Var3 *)
```

Пример на ST:

```
Результат := not BYTE#10; (* Результат = 245дес. *)
```

3.4.3.4 OR и OR(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных	X	X	X	X

Побитовая связь "ИЛИ" AE/Akku с одной или двумя переменными или константами. Побитовая связь "ИЛИ" (…) содержимого AE/Akku и содержимого AE/Akku, предварительно полученного в скобках. Допускается использование до 6 уровней скобок. Все величины должны относится к одному типу переменных.

Пример на AWL:

```
LD 170
OR 204 (* OR связь между двумя константами *)

LD 170 (* Связь между одной константой и 2 переменными *)
OR Var1, Var2 (* Akku = 170d OR Var1OR Var2 *)

LD Var1
OR ( Var2 (* AE/Akku = Var1 OR ( Var2 AND Var3 ) *)
AND Var3
)
```

Пример на ST:

```
Результат := 170 or 204; (* Результат = 238 *)
```

Var2	Var1	Результат
0	0	0
0	1	1
1	0	1
1	1	1

3.4.3.5 ORN и ORN(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных	X	X	X	X

Побитовая связь ИЛИ значения AE/Akku с инвертированным операндом. Побитовая связь "ИЛИ" (…) с AE/Akku и инвертированным результатом в скобках. Допускается использование до 6 уровней скобок. Обе величины должны относится к одному типу переменных.

Пример на AWL:

```
LD 2#0000_1111
ORN 2#0011_1010 (* ORN связь между двумя константами *)
(* Akku = 2#1100_0000 *)

LD 170 (* Связь между одной константой и 2 переменными *)
ORN Var1, Var2 (* Akku = 170d ORN Var1 ORN Var2 *)

LD Var1
ORN ( Var2 (* AE/Akku = Var1 ORN ( Var2 OR Var3 ) *)
OR Var3
)
```

Пример на ST:

```
Результат := 2#0000_1111 ORN 2#0011_1010; (* Результат = 2#1100_0000 *)
```

Var2	Var1	Результат
0	0	1
0	1	0
1	0	0
1	1	0

3.4.3.6 ROL

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Побитовый циклический сдвиг содержимого Akku влево. При этом содержимое Akku сдвигается влево на n раз, левый бит перемещается на место правого.

Пример на AWL:

```
LD 175     (* Загружает значение 1010_1111 *)
ROL 2      (* Содержимое Akku 2 раза прокручивается влево *)
ST Value1  (* Сохраняет значение 1011_1110 *)
```

Пример на ST:

```
Результат := ROL(BYTE#175, 2); (* Результат = 2#1011_1110 *)
Результат := ROL(INT#175, 2); (* Результат = 16#C02B *)
```

3.4.3.7 ROR

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Побитовый циклический сдвиг содержимого Akku вправо. При этом содержимое Akku сдвигается вправо на n раз, правый бит перемещается на место левого.

Пример на AWL:

```
LD 175     (* Загружает значение 1010_1111 *)
ROR 2      (* Содержимое Akku 2 раза сдвигается вправо *)
ST Value1  (* Сохраняет значение 1110_1011 *)
```

Пример на ST:

```
Результат := ROR(BYTE#175, 2); (* Результат = 2#1110_1011 *)
```

3.4.3.8 S и R

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных	X			

Назначение и сброс булевой переменной, если предыдущий результат связи (AE) имел значение TRUE.

Пример на AWL:

```
LD TRUE    (* Загружает в AE значение TRUE *)
S Var1     (* VAR1 присваивается TRUE*)
R Var1     (* VAR1 присваивается FALSE*)
```

Пример на ST:

```
Результат := TRUE;
Результат := FALSE;
```

3.4.3.9 SHL

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Побитовый сдвиг влево Akku. При этом содержимое Akku сдвигается влево на n раз, выдвигаемые биты теряются.

Пример на AWL:

```
LD 175     (* Загружает значение 1010_1111 *)
SHL 2      (* Содержимое Akku 2 раза сдвигается влево *)
ST Value1  (* Сохраняет значение 1011_1100 *)
```

Пример на ST:

```
Результат := SHL(BYTE#175, 2); (* Результат = 2#1011_1100 *)
Результат := SHL(INT#175, 2); (* Результат = 16#2BC *)
```

3.4.3.10 SHR

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Побитовый сдвиг Akku вправо. При этом содержимое Akku сдвигается вправо на n раз, выдвигаемые биты теряются.

Пример на AWL:

```
LD 175     (* Загружает значение 1010_1111 *)
SHR 2      (* Содержимое Akku 2 раза сдвигается вправо *)
ST Value1  (* Сохраняет значение 0010_1011 *)
```

Пример на ST:

```
Результат := SHR(BYTE#175, 2); (* Результат = 2#0010_1011 *)
```

3.4.3.11 XOR и XOR(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных	X			

Побитовая связь “exclusive OR“ между AE/Akku и с одной или двумя переменными или константами. Первое значение находится в AE/Akku, второе загружается по команде, либо находится внутри скобок. Связываемые величины должны относится к одному типу переменных.

Пример на AWL:

```
LD 2#0000_1111
XOR 2#0011_1010 (* XOR связь между двумя константами *)
                  (* Akku = 2#0011_0101 *)

LD 170          (* Связь между одной константой и 2 переменными *)
XOR Var1, Var2  (* Akku = 170d XOR Var1 XOR Var2 *)

LD Var1
XOR ( Var2      (* AE/Akku = Var1 XOR ( Var2 OR Var3 ) *)
OR Var3
)
```

Пример на ST:

```
Результат := 2#0000_1111 XOR 2#0011_1010; (* результат = 2#0011_0101 *)
```

Var2	Var1	Результат
0	0	0
0	1	1
1	0	1
1	1	0

3.4.3.12 XORN и XORN(

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных	X			

Побитовая связь "исключающее ИЛИ" значения AE/Akku с инвертированным операндом. Побитовая связь "исключающее ИЛИ" (…) с AE/Akku и инвертированным результатом в скобках. Допускается использование до 6 уровней скобок. Связываемые величины должны относится к одному типу переменных.

Пример на AWL:

```
LD 2#0000_1111
XORN 2#0011_1010 (* XORN связь между двумя константами *)
                 (* Akku = 2#1100_1010 *)

LD 170           (* Связь между одной константой и 2 переменными *)
XORN Var1, Var2  (* Akku = 170d XORN Var1 XORN Var2 *)

LD Var1
XORN ( Var2      (* AE/Akku = Var1 XORN ( Var2 OR Var3 ) *)
OR Var3
)
```

Пример на ST:

```
Результат := 2#0000_1111 XORN 2#0011_1010; (* Результат = 2#1100_1010 *)
```

Var2	Var1	Результат
0	0	1
0	1	0
1	0	0
1	1	1

3.4.4 Операторы загрузки и сохранения

3.4.4.1 LD

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных	X	X	X	X

Загружает константы или переменные в текущее значение AE или в накопительный регистр Akku.

Пример на AWL:

```
LD 10 (* Загружает 10 типа BYTE *)
LD -1000 (* Загружает -1000 типа INT *)
LD Value1 (* Загружает переменную Value1 *)
```

3.4.4.2 LDN

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных	X			

Загружает инвертированную булеву переменную в AE.

Пример на AWL:

```
LDN Value1 (* Value1 = TRUE à AE = FALSE *)
ST Value2 (* Сохраняет в Value2 = FALSE *)
```

3.4.4.3 ST

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных	X	X	X	X

Сохраняет содержимое AE/Akku в переменную. Сохраняемая переменная должна соответствовать предварительно загруженному и обработанному типу данных.

Пример на AWL:

```
LD 100 (* Загружает значение 1010_1111 *)
ST Value1 (* Содержимое Akku 100 сохраняется в Value1 *)
```

3.4.4.4 STN

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных	X			

Сохраняет содержимое AE в переменную и инвертирует ее. Сохраняемая переменная должна соответствовать предварительно загруженному и обработанному типу данных.

Пример на AWL:

```
LD Value1 (* Value1 = TRUE à AE = TRUE *)
STN Value2 (* Сохраняет в Value2 = FALSE *)
```

3.4.5 Операторы сравнения

3.4.5.1 EQ

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Сравнивает содержание Akku с переменной или константой. Если значения равны, AE присваивается TRUE.

Пример на AWL:

```
LD Value1 (* Value1 = 5 *)
EQ 10 (* AE = 5 равно 10 ? *)
JMPC NextStep (* AE = FALSE - программа не выполняет переход*)
ADD 1
NextStep:
ST Value1
```

Пример на ST:

```
(* Value = 10? *)
if Value = 10 then
  Value2 := 5;
  end_if;
```

3.4.5.2 GE

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Сравнивает содержание Akku с переменной или константой. Если значение в Akku больше или равно переменной или константе, AE присваивается значение TRUE.

Пример на AWL:

```
LD Value1 (* Value1 = 5 *)
GE 10 (* 5 больше или равно 10?*) *)
JMPC NextStep (* AE = FALSE - программа не выполняет переход*)
ADD 1

NextStep:
ST Value1
```

Пример на ST:

```
(* 5 больше или равно 10?*) *)
if Value >= 10 then
  Value := Value - 1
  end_if;
```

3.4.5.3 GT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Сравнивает содержание Akku с переменной или константой. Если значение в Akku больше переменной или константы, AE присваивается значение TRUE.

Пример на AWL:

```
LD Value1(* Value1 = 12 *)
GT 8 (* 12 больше 8? *) *)
JMPC NextStep (* AE = TRUE - программа выполняет переход*)
ADD 1
NextStep:
ST Value1
```

Пример на ST:

```
(* 12 больше 8? *) *)
if Value > 8 then
  Value := 0;
end_if;
```

3.4.5.4 LE

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Сравнивает содержание Akku с переменной или константой. Если значение в Akku меньше или равно переменной или константе, AE присваивается значение TRUE.

Пример на AWL:

```
LD Value1 (* Value1 = 5 *)
LE 10 (* 5 меньше или равно 10?*) *)
JMPC NextStep:
ST Value1
```

Пример на ST:

```
(* Значение меньше или равно 10?*)
if Value <= 10 then
  Value := 11;
  end_if;
```

3.4.5.5 LT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Сравнивает содержание Akku с переменной или константой. Если значение в Akku меньше переменной или константе, AE присваивается значение TRUE.

Пример на AWL:

```
LD Value1 (* Value1 = 12 *)
LT 8 (* 12 меньше 8 ? *)
JMPC NextStep (* AE = FALSE à программа не выполняет переход*)
ADD 1
NextStep:
ST Value1
```

Пример на ST:

```
(* Значение меньше 0? *)
if Value < 0 then
  Value := 0;
  end_if;
```

3.4.5.6 NE

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X	X	X

Сравнивает содержание Akku с переменной или константой. Если значение в Akku не равно переменной или константе, AE присваивается значение TRUE.

Пример на AWL:

```
LD Value1 (* Value1 = 5 *)
NE 10 (* 5 равно 10 ?
JMPC NextStep (* AE = TRUE à программа выполняет переход*)
ADD 1
NextStep:
ST Value1
```

Пример на ST:

```
if Value <> 5 then
Value := 5;
end_if;
```

3.5 Параметры процессов

Все аналоговые и цифровые входы и выходы, а также уставки и фактические значения шин, могут считываться и обрабатываться ПЛК, либо назначаться им (для выходных значений). Доступ к отдельным значениям осуществляется с помощью описанных далее параметров процессов. Для выходных значений выход (например, цифровой выход или уставка ПЛК) должен быть запрограммирован так, чтобы в качестве источника события выступал ПЛК. Все данные процессов при каждом новом выполнении цикла сначала считываются ПЛК с устройства, и только в конце программы для ПЛК записываются на устройство! В нижеследующей таблице представлены все параметры, к которым может напрямую обращаться функция ПЛК. Доступ к другим параметрам осуществляется через функциональные блоки MC_ReadParameter или MC_WriteParameter.

3.5.1 Входы и выходы

Здесь объединены все параметры процессов, описывающие интерфейсы входов и выходов прибора.

Имя	Функция	Нормирование	Тип	Доступ	Устройства
_0_Set_digital_output	Установка цифровых выходов	Бит 0: Mfr1 Бит 1: Mfr2 Бит 2: DOUT 1 Бит 3: DOUT 2 Бит 4: DOUT 1 CU5-MLT Бит 5: DOUT 2 CU5-MLT Бит 6: DOUT 3 CU5-MLT Бит 7: DOUT 4 CU5-MLT Бит 8: цифр. функция AOUT Бит 9: свободно Бит 10: BusIO Bit0 Бит 11: BusIO Bit1 Бит 12: BusIO Bit2 Бит 13: BusIO Bit3 Бит 14: BusIO Bit4 Бит 15: BusIO Bit5	UINT	R/W	SK 5xxP
_0_Set_digital_output	Установка цифровых выходов	Бит 0: Mfr1 Бит 1: Mfr2 Бит 2: DOUT1 Бит 3: DOUT2 Бит 4: цифр. функция AOUT Бит 5: DOUT3 (Din7) Бит 6: Слово состояния Бит 10 Бит 7: Слово состояния Бит 13 Бит 8: BusIO Bit0	UINT	R/W	SK 54xE

Имя	Функция	Нормирование	Тип	Доступ	Устройства
		Бит 9: BusIO Bit1 Бит 10: BusIO Bit2 Бит 11: BusIO Bit3 Бит 12: BusIO Bit4 Бит 13: BusIO Bit5 Бит 14: BusIO Bit6 Бит 15: BusIO Bit7			
_0_Set_digital_output	Установка цифровых выходов	Бит 0: Mfr1 Бит 1: Mfr2 Бит 2: DOUT1 Бит 3: DOUT2 Бит 4: цифр. функция AOUT Бит 5: свободно Бит 6: Слово состояния Бит 10 Бит 7: Слово состояния Бит 13 Бит 8: BusIO Bit0 Бит 9: BusIO Bit1 Бит 10: BusIO Bit2 Бит 11: BusIO Bit3 Бит 12: BusIO Bit4 Бит 13: BusIO Bit5 Бит 14: BusIO Bit6 Бит 15: BusIO Bit7	UINT	R/W	SK 52xE SK 53xE
_0_Set_digital_output	Установка цифровых выходов	Бит 0: DOUT1 Бит 1: BusIO Bit0 Бит 2: BusIO Bit1 Бит 3: BusIO Bit2 Бит 4: BusIO Bit3 Бит 5: BusIO Bit4 Бит 6: BusIO Bit5 Бит 7: BusIO Bit6 Бит 8: BusIO Bit7 Бит 9: Bus PZD Bit 10 Бит 10: Bus PZD Бит 13 Бит 11: DOUT2	UINT	R/W	SK 2xxE SK 2xxE-FDS
_0_Set_digital_output	Установка цифровых выходов	Бит 0: DOUT1 Бит 1: DOUT2 Бит 2: BusIO Bit0 Бит 3: BusIO Bit1 Бит 4: BusIO Bit2 Бит 5: BusIO Bit3 Бит 6: BusIO Bit4 Бит 7: BusIO Bit5 Бит 8: BusIO Bit6 Бит 9: BusIO Bit7 Бит 10: Bus PZD Bit 10 Бит 11: Bus PZD Бит 13	UINT	R/W	SK 180E SK 190E

Имя	Функция	Нормирование	Тип	Доступ	Устройства
_0_Set_digital_output	Установка цифровых выходов	Бит 0: DOUT1 Бит 1: DOUT2 Бит 2: DOUT_BRAKE Бит 3: DOUT_BUS1 Бит 4: DOUT_BUS2	UINT	R/W	SK 155E-FDS SK 175E-FDS
_1_Set_analog_output	Установка аналоговых выходов ЧП	10,0В = 100	BYTE	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE
_2_Set_external_ analog_out1	Установка аналог. выхода 1 IOE	10,0В = 100	BYTE	R/W	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_3_Set_external_ analog_out2	Установка аналог. выхода 2 IOE	10,0В = 100	BYTE	R/W	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_4_State_digital_output	Состояние цифровых выходов	Бит 0: Mfr1 Бит 1: Mfr2 Бит 2: DOUT 1 Бит 3: DOUT 2 Бит 4: DOUT 1 CU5-MLT Бит 5: DOUT 2 CU5-MLT Бит 6: DOUT 3 CU5-MLT Бит 7: DOUT 4 CU5-MLT Бит 8: цифр. функция AOUT Бит 9: свободно Бит 10: DOUT1 IOE1 Бит 11: DOUT2 IOE1 Бит 12: DOUT1 IOE2 Бит 13: DOUT2 IOE2 Бит 14: свободно Бит 15: свободно	INT	R	SK 5xxP
_4_State_digital_output	Состояние цифровых выходов	Бит 0: Mfr1 Бит 1: Mfr2 Бит 2: DOUT1 Бит 3: DOUT2 Бит 4: цифр. функция AOUT Бит 5: DOUT3 (Din7)	INT	R	SK 54xE

Имя	Функция	Нормирование	Тип	Доступ	Устройства
		Бит 6: Слово состояния Бит 8 Бит 7: Слово состояния Бит 9 Бит 8: BusIO Bit0 Бит 9: BusIO Bit1 Бит 10: BusIO Bit2 Бит 11: BusIO Bit3 Бит 12: BusIO Bit4 Бит 13: BusIO Bit5 Бит 14: BusIO Bit6 Бит 15: BusIO Bit7			
_4_State_digital_output	Состояние цифровых выходов	P711	BYTE	R	SK 52xE SK 53xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_4_State_digital_output	Состояние цифровых выходов	Бит 0: DOUT1 Бит 1: DOUT2 Бит 2: DOUT_BRAKE Бит 3: DOUT_BUS1 Бит 4: DOUT_BUS2	BYTE	R	SK 155E-FDS SK 175E-FDS
_5_State_Digital_input	Состояние цифровых входов	Бит 0: DIN1 Бит 1: DIN2 Бит 2: DIN3 Бит 3: DIN4 Бит 4: DIN5 Бит 5: DIN6 Бит 6: DIN1 CU5-MLT Бит 7: DIN2 CU5-MLT Бит 8: DIN3 CU5-MLT Bit 9 DIN4 CU5-MLT Бит 10 свободно Бит 11 свободно Бит 12: Цифровая функция AIN1 Бит 8: Цифровая функция AIN2	INT	R	SK 5xxP
_5_State_Digital_input	Состояние цифровых входов	Бит 0: DIN1 Бит 1: DIN2 Бит 2: DIN3 Бит 3: DIN4 Бит 4: DIN5 Бит 5: DIN6 Бит 6: DIN7 Бит 7: Цифровая функция AIN1 Бит 8: Цифровая функция AIN2	INT	R	SK 54xE

Имя	Функция	Нормирование	Тип	Доступ	Устройства
_5_State_Digital_input	Состояние цифровых входов	Бит 0: DIN1 Бит 1: DIN2 Бит 2: DIN3 Бит 3: DIN4 Бит 4: DIN5 Бит 5: DIN6 Бит 6: DIN7	INT	R	SK 52xE SK 53xE
_5_State_Digital_input	Состояние цифровых входов	Бит 0: DIN1 Бит 1: DIN2 Бит 2: DIN3 Бит 3: DIN4 Бит 4: свободно Бит 5: Позистор Бит 6: свободно Бит 7: свободно Бит 8: DIN1 IOE 1 Бит 9: DIN2 IOE 1 Бит 10: DIN3 IOE 1 Бит 11: DIN4 IOE 1 Бит 12: DIN1 IOE 2 Бит 13: DIN2 IOE 2 Бит 14: DIN3 IOE 2 Бит 15: DIN4 IOE 2	INT	R	SK 2xxE
_5_State_Digital_input	Состояние цифровых входов	Бит 0: DIN1 Бит 1: DIN2 Бит 2: DIN3 Бит 3: AIN1 Бит 4: AIN2 Бит 5: Позистор Бит 6: свободно Бит 7: свободно Бит 8: DIN1 IOE 1 Бит 9: DIN2 IOE 1 Бит 10: DIN3 IOE 1 Бит 11: DIN4 IOE 1 Бит 12: DIN1 IOE 2 Бит 13: DIN2 IOE 2 Бит 14: DIN3 IOE 2 Бит 15: DIN4 IOE 2	INT	R	SK 180E SK 190E
_5_State_Digital_input	Состояние цифровых входов	Бит 0: DIN1 Бит 1: DIN2 Бит 2: DIN3 Бит 3: TF (позистор) Бит 4: DIN-BUS1 (ASiI1) Бит 5: DIN-BUS2 (ASiI2) Бит 6: DIN-BUS3 (ASiI3) Бит 7: DIN-BUS4 (ASiI4) Бит 8: BDDI1 (ASIO3) Бит 9: BDDI2 (ASIO4) Бит 10: STO	INT	R	SK 155E-FDS SK 175E-FDS

Имя	Функция	Нормирование	Тип	Доступ	Устройства
_5_State_Digital_input	Состояние цифровых входов	Бит 0: DIN1 Бит 1: DIN2 Бит 2: DIN3 Бит 3: DIN4 Бит 4: DIN5 Бит 5: DIN6/AIN1 Бит 6: DIN7/AIN2 Бит 7: Позистор Бит 8: DIN1 IOE 1 Бит 9: DIN2 IOE 1 Бит 10: DIN3 IOE 1 Бит 11: DIN4 IOE 1 Бит 12: DIN1 IOE 2 Бит 13: DIN2 IOE 2 Бит 14: DIN3 IOE 2 Бит 15: DIN4 IOE 2	INT	R	SK 2xxE-FDS
_6_Delay_digital_inputs	Состояние цифровых входов после P475	Бит 0: DIN1 Бит 1: DIN2 Бит 2: DIN3 Бит 3: DIN4 Бит 4: DIN5 Бит 5: DIN6 Бит 6: DIN7 Бит 7: Цифровая функция AIN1 Бит 8: Цифровая функция AIN2	INT	R	SK 5xxP SK 54xE
_6_Delay_digital_inputs	Состояние цифровых входов после P475	Бит 0: DIN1 Бит 1: DIN2 Бит 2: DIN3 Бит 3: DIN4 Бит 4: DIN5 Бит 5: DIN6 Бит 6: DIN7	INT	R	SK 52xE SK 53xE
_6_Delay_digital_inputs	Состояние цифровых входов после P475	Бит 0: DIN1 Бит 1: DIN2 Бит 2: DIN3 Бит 3: AIN1 Бит 4: AIN2 Бит 5: Позистор Бит 6: свободн. Бит 7: свободн. Бит 8: DIN1 IOE 1 Бит 9: DIN2 IOE 1 Бит 10: DIN3 IOE 1 Бит 11: DIN4 IOE 1 Бит 12: DIN1 IOE 2 Бит 13: DIN2 IOE 2 Бит 14: DIN3 IOE 2 Бит 15: DIN4 IOE 2	INT	R	SK 2xxE SK 180E SK 190E

Имя	Функция	Нормирование	Тип	Доступ	Устройства
_6_Delay_digital_inputs	Состояние цифровых входов после P475	Бит 0: DIN1 Бит 1: DIN2 Бит 2: DIN3 Бит 3: DIN4 Бит 4: DIN5 Бит 5: DIN6/AIN1 Бит 6: DIN7/AIN2 Бит 7: Позистор Бит 8: DIN1 IOE 1 Бит 9: DIN2 IOE 1 Бит 10: DIN3 IOE 1 Бит 11: DIN4 IOE 1 Бит 12: DIN1 IOE 2 Бит 13: DIN2 IOE 2 Бит 14: DIN3 IOE 2 Бит 15: DIN4 IOE 2	INT	R	SK 2xxE-FDS
_7_Analog_input1	Значение аналогового входа 1 (AIN1)	10,00В = 1000	INT	R	все
_8_Analog_input2	Значение аналогового входа 2 (AIN2)	10,00В = 1000	INT	R	все
_9_Analog_input3	Значение аналоговой функции DIN2	10,00В = 1000	INT	R	SK 5xxP SK 54xE SK 155E-FDS SK 175E-FDS
_10_Analog_input4	Значение аналоговой функции DIN3	10,00В = 1000	INT	R	SK 5xxP SK 54xE SK 155E-FDS SK 175E-FDS
_11_External_analog_input1	Значение аналогового входа 1 (1.IOE)	10,00В = 1000	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_12_External_analog_input2	Значение аналогового входа 2 (1.IOE)	10,00В = 1000	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_13_External_analog_input3	Значение аналогового входа 1 (2.IOE)	10,00В = 1000	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_14_External_analog_	Значение аналогового	10,00В = 1000	INT	R	SK 5xxP

Имя	Функция	Нормирование	Тип	Доступ	Устройства
input4	входа 2 (2.IOE)				SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_15_State_analog_ output	Состояние аналогового выхода	10,0В = 100	BYTE	R	SK 5xxP SK 54xE
_16_State_ext_analog_ out1	Состояние аналогового выхода (1 IOE)	10,00В = 1000	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_17_State_ext_analog_ out2	Состояние аналогового выхода (2 IOE)	10,00В = 1000	INT	R	SK 5xxP SK 54xE SK 2xxE SK 180E SK 190E
_18_Dip_switch_state	Состояние DIP переключателя	Бит 0: DIP1 Бит 1: DIP2 Бит 2: DIP3 Бит 3: DIP4 Бит 4: DIP_I1 Бит 5: DIP_I2 Бит 6: DIP_I3 Бит 7: DIP_I4	INT	R	SK 155E-FDS SK 175E-FDS

3.5.2 Уставки и фактические значения ПЛК

Представленные здесь параметры процессов формируют интерфейс между ПЛК и устройством. Функция уставок ПЛК определяется параметром (P553).

Информация

Параметр PLC_control_word перезаписывает функциональный блок MC_Power. Уставки ПЛК перезаписывают функциональные блоки MC_Move…. и MC_Home.

Имя	Функция	Нормирование	Тип	Доступ	Устройство
_20_PLC_control_word	Команда управления ПЛК	Соответствует профилю USS	INT	R/W	все
_21_PLC_set_val1	Заданное значение ПЛК 1	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_22_PLC_set_val2	Заданное значение ПЛК 2	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_23_PLC_set_val3	Заданное значение ПЛК 3	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_24_PLC_set_val4	Заданное значение ПЛК 4	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS
_25_PLC_set_val5	Заданное значение ПЛК 5	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS
_26_PLC_additional_	Доп. команда	Соответствует	INT	R/W	SK 5xxP

Имя	Функция	Нормирование	Тип	Доступ	Устройство
control_word1	управления ПЛК 1	профилю USS			SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_27_PLC_additional_control_word2	Доп. команда управления ПЛК 2	Соответствует профилю USS	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_28_PLC_status_word	Слово состояния ПЛК	Соответствует профилю USS	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_29_PLC_act_val1	Факт. значение ПЛК 1	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_30_PLC_act_val2	Факт. значение ПЛК 2	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_31_PLC_act_val3	Факт. значение ПЛК 3	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_32_PLC_act_val4	Факт. значение ПЛК 4	100% = 4000h	INT	R/W	SK 5xxP SK 54xE

Имя	Функция	Нормирование	Тип	Доступ	Устройство
					SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS
_33_PLC_act_val5	Факт. значение ПЛК 5	100% = 4000h	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS
_34_PLC_Busmaster_Control_word	Команда управления ведущей функции (функция ведущей шины) через ПЛК	Соответствует профилю USS	INT	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_35_PLC_32Bit_set_val1	Уставка ПЛК 32бит - P553[1] = Low Part значения 32бит - P553[2] = High Part значения 32бит	_	LONG	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_36_PLC_32Bit_act_val1	Факт. значение ПЛК 32бит - Факт. значение ПЛК 1 = Low Part значения 32бит - Факт. значение ПЛК 2 = High Part значения 32бит	_	LONG	R/W	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_37_PLC_status_bits	Виртуальные выходы состояния ПЛК	Бит 0: PLC-DOUT1 Бит 1: PLC-DOUT2	INT	R/W	SK 155E-FDS SK 175E-FDS
_38_PLC_control_bits	Виртуальные управляющие выходы ПЛК	Бит 0: PLC-DIN1 Бит 1: PLC-DIN2 Бит 2: PLC-DIN3 Бит 3: PLC-DIN4 Бит 4: PLC-DIN5 Бит 5: PLC-DIN6 Бит 6: PLC-DIN7 Бит 7: PLC-DIN8	INT	R/W	SK 155E-FDS SK 175E-FDS

3.5.3 Уставки и действительные значения шины

Эти параметры процессов отражают все уставки и фактические значения, передаваемые на устройство по различным системам шин.

Имя	Функция	Нормирование	Тип	Доступ	Устройства
_40_Inverter_status	Слово состояния ЧП	Соответствует профилю USS	INT	R	все
_41_Inverter_act_val1	Тек.знач. 1 ПЛК	100% = 4000h	INT	R	все
_42_Inverter_act_val2	Тек.знач. 2 ПЛК	100% = 4000h	INT	R	все
_43_Inverter_act_val3	Тек.знач. 3 ПЛК	100% = 4000h	INT	R	все
_44_Inverter_act_val4	Тек.знач. 4 ПЛК	100% = 4000h	INT	R	SK 5xxP SK 54xE
_45_Inverter_act_val5	Тек.знач. 5 ПЛК	100% = 4000h	INT	R	SK 5xxP SK 54xE
_46_Inverter_lead_val1	Широковещательная функция ведущего устройства: ведущее значение 1	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_47_Inverter_lead_val2	Широковещательная функция ведущего устройства: ведущее значение 2	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_48_Inverter_lead_val3	Широковещательная функция ведущего устройства: ведущее значение 3	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_49_Inverter_lead_val4	Широковещательная функция ведущего устройства: ведущее значение 4	100% = 4000h	INT	R	SK 5xxP SK 54xE
_50_Inverter_lead_val5	Широковещательная функция ведущего устройства: ведущее значение 5	100% = 4000h	INT	R	SK 5xxP SK 54xE

Имя	Функция	Нормирование	Тип	Доступ	Устройства
_51_Inverter_control_word	Результирующее управляющее слово шины	Соответствует профилю USS	ÎNT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_52_Inverter_set_val1	Результирующая главная уставка 1 шины	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_53_Inverter_set_val2	Результирующая главная уставка 2 шины	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_54_Inverter_set_val3	Результирующая главная уставка 3 шины	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_55_Inverter_set_val4	Результирующая главная уставка 4 шины	100% = 4000h	INT	R	SK 5xxP SK 54xE
_56_Inverter_set_val5	Результирующая главная уставка 5 шины	100% = 4000h	INT	R	SK 5xxP SK 54xE
_57_Broadcast_set_val1	Широковещательный режим, ведомое устройство: Вспом. уставка 1	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_58_Broadcast_set_val2	Широковещательный режим, ведомое устройство: Вспом. уставка 2	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE

Имя	Функция	Нормирование	Тип	Доступ	Устройства
					SK 180E SK 190E
_59_Broadcast_set_val 3	Широковещательный режим, ведомое устройство: Вспом. уставка 3	100% = 4000h	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_60_Broadcast_set_val 4	Широковещательный режим, ведомое устройство: Вспом. уставка 4	100% = 4000h	INT	R	SK 5xxP SK 54xE
_61_Broadcast_set_val 5	Широковещательный режим, ведомое устройство: Вспом. уставка 5	100% = 4000h	INT	R	SK 5xxP SK 54xE
_62_Inverter_32Bit_set _val1	Результирующая главная уставка 1 шины 32 бит	- Low Part в P546[1] - High Part в P546[2]	LONG	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_63_Inverter_32Bit_act _val1	Тек.знач. 1 ЧП 32 бит	- Low Part в P543[1] - High Part в P543[2]	LONG	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_64_Inverter_32Bit_lea d_val1	Ведущее значение 1 32 бит	- Low Part в P502[1] - High Part в P502[2]	LONG	R	SK 5xxP SK 54xE SK 2xxE SK 180E SK 190E
_65_Broadcast_32Bit_s et_val1	Вспом. уставка 1 ведомого устройства 32 бит, широковещ. режим	- Low Part в P543[1] - High Part в P543[2]	LONG	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_66_BusIO_input_bits	Входящие данные шины ввода-вывода	- Бит 0 – 7 = Bus I/O In бит 0 – 7 - Бит 8 = Метка 1 - Бит 9 = Метка 2 - Бит 10 = Бит 8 из упр. слова шины	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS

Имя	Функция	Нормирование	Тип	Доступ	Устройства
		- Бит 11 = Бит 9 из упр. слова шины			SK 180E SK 190E
_67_BusIO_output_bits	Выходящие данные шины ввода-вывода	Бит0 = Шина / AS-i цифр.выход1 Бит1 = Шина / AS-i цифр.выход2 Бит2 = Шина / AS-i цифр.выход3 Бит3 = Шина / AS-i цифр.выход4 Бит4 = Шина / 1.IOE цифр. выход1 Бит5 = Шина / 1.IOE цифр. выход2 Бит6 = Шина / 2.IOE цифр. выход1 Бит7 = Шина / 2.IOE цифр. выход2 Бит8 = Бит 10 шина Команда управления Бит9 = Бит 11 шина Команда управления	INT	R	SK 5xxP SK 54xE
_67_BusIO_output_bits	Выходящие данные шины ввода-вывода	Бит0 = Шина / AS-i цифр.выход1 Бит1 = Шина / AS-i цифр.выход2 Бит2 = Шина / AS-i цифр.выход3 Бит3 = Шина / AS-i цифр.выход4 Бит4 = AS-i исполн. механизм 1 Бит5 = AS-i исполн. механизм 2 Бит6 = Метка 1 Бит7 = Метка 2 Бит8 = Бит 10 шина Команда управления Бит9 = Бит 11 шина Команда управления	INT	R	SK 53xE SK 52xE
_67_BusIO_output_bits	Выходящие данные шины ввода-вывода	Бит0 = Шина / AS-i цифр.выход1 Бит1 = Шина / AS-i цифр.выход2 Бит2 = Шина / AS-i цифр.выход3 Бит3 = Шина / AS-i цифр.выход4 Бит4 = Шина / IOE цифр. выход1 Бит5 = Шина / IOE цифр. выход2 Бит6 = Шина / 2ой IOE	INT	R	SK 2xxE

Имя	Функция	Нормирование	Тип	Доступ	Устройства
		цифр. выход1 Бит7 = Шина / 2ой IOE цифр. выход2 Бит8 = Бит 10 шина Команда управления Бит9 = Бит 11 шина Команда управления			
_67_BusIO_output_bits	Выходящие данные шины ввода-вывода	Бит0 = Шина / AS-i цифр.выход1 Бит1 = Шина / AS-i цифр.выход2 Бит2 = Шина / AS-i цифр.выход3 Бит3 = Шина / AS-i цифр.выход4 Бит4 = Шина / AS-i цифр.выход5 Бит5 = Шина / AS-i цифр.выход6 Бит6 = Шина / 2ой IOE цифр. выход1 Бит7 = Шина / 2ой IOE цифр. выход2 Бит8 = Бит 10 шина Команда управления Бит9 = Бит 11 шина Команда управления	INT	R	SK 2xxE-FDS

3.5.4 ControlBox и ParameterBox

С помощью представленных здесь параметров процессов может осуществляться доступ к блокам управления. Это позволяет реализовывать простые задачи интерфейса HMI.

Информация

Чтобы „key_states“ отображались в ПЛК, ControlBox и ParameterBox должны находиться в режиме индикации-ПЛК-. В ином случае отображается только значение „0“.

Имя	Функция	Нормирование	Тип	Доступ	Устройства
_70_Set_controlbox_ show_val	Отображаемое значение для ControlBox	Отображаемое значение = бит 29 – бит 0 Положение запятой = бит 31 – бит 30	DINT	R/W	все
_71_Controlbox_key_ state	Состояние клавиатуры ControlBox	Бит 0: ON Бит 1: OFF Бит 2: DIR Бит 3: UP Бит 4: DOWN Бит 5: Enter	BYTE	R	все
_72_Parameterbox_ key_state	Состояние клавиатуры ParameterBox	Бит 0: ON Бит 1: OFF Бит 2: DIR Бит 3: UP Бит 4: DOWN Бит 5: Enter Бит 6: Right Бит 7: Left	BYTE	R	все

3.5.5 Информационный параметр

Здесь приводятся важнейшие фактические значения прибора.

Имя	Функция	Нормирование	Тип	Доступ	Устройство
_80_Current_fault	текущий номер неисправности	Ошибка 10.0 = 100	BYTE	R	все
_81_Current_warning	текущее предупреждение	Предупреждение 10.0 = 100	BYTE	R	все
_82_Current_reason_FI_blocked	текущая причина для состояния "Einschaltsperre"	Проблема 10.0 = 100	BYTE	R	все
_83_Input_voltage	текущее сетевое напряжение	100В = 100	INT	R	все
_84_Current_frequenz	текущая частота	10Гц = 100	INT	R	все
_85_Current_set_point_frequency1	текущая уставка частоты, полученная из источника уставки	10Гц = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_86_Current_set_point_frequency2	текущая уставка частоты преобразователь	10Гц = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_87_Current_set_point_frequency3	текущая уставка частоты по линейному изменению	10Гц = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_88_Current_Speed	текущее рассчитанное значение частоты вращения	100об/мин = 100	INT	R	все
_89_Actual_current	текущее значение выходного тока	10,0А = 100	INT	R	все
_90_Actual_torque_current	действительный ток крутящего момента	10,0А = 100	INT	R	все
_91_Current_voltage	текущее значение напряжения	100В = 100	ÎNT	R	все

Имя	Функция	Нормирование	Тип	Доступ	Устройство
_92_Dc_link_voltage	текущее значение напряжения в промежуточной цепи	100В = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_93_Actual_field_current	действительный ток намагничивания	10,0A = 100	INT	R	все
_94_Voltage_d	текущее значение составляющей напряжения ось d	100В = 100	INT	R	все
_95_Voltage_q	текущее значение составляющей напряжения ось q	100В = 100	INT	R	все
_96_Current_cos_phi	текущее значение Cos(phi)	0.80 = 80	BYTE	R	все
_97_Torque	текущее значение крутящего момента	100% = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_98_Field	текущее поле	100% = 100	BYTE	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_99_Apparent_power	текущее значение потребляемой мощности	1,00кВт = 100	INT	R	все
_100_Mechanical_power	текущая механическая мощность	1,00кВт = 100	INT	R	все
_101_Speed_encoder	текущее измеренное значение частоты вращения	100об/мин = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE
_102_Usage_rate_motor	текущий коэффициент нагрузки двигателя (мгновенное значение)	100% = 100	INT	R	все

Имя	Функция	Нормирование	Тип	Доступ	Устройство
_103_Usage_rate_motor_I2t	текущий коэффициент нагрузки двигателя I2t	100% = 100	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_104_Usage_rate_brake_resistor	текущий коэффициент нагрузки тормозного резистора	100% = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_105_Head_sink_temp	текущая температура радиатора	100°C = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_106_Inside_temp	текущая температура внутри устройства	100°C = 100	INT	R	SK 5xxP SK 54xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_107_Motor_temp	текущая температура двигателя	100°C = 100	INT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 2xxE-FDS SK 180E SK 190E
_108_Actual_net_frequency	текущая частота сети	10Гц = 100	INT	R	SK 155E-FDS SK 175E-FDS
_109_Mains_phase_sequence	текущая последовательность фаз сети	0=CW, 1=CCW	BYTE	R	SK 155E-FDS SK 175E-FDS
_141_Pos_Sensor_Inc	Положение инкрементного датчика	0.001 оборота	DINT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E

Имя	Функция	Нормирование	Тип	Доступ	Устройство
_142_Pos_Sensor_Abs	Положение абсолютного энкодера	0.001 оборота	DINT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_143_Pos_Sensor_Uni	Положение универсального энкодера	0.001 оборота	DINT	R	SK 5xxP SK 54xE
_144_Pos_Sensor_HTL	Положение HTL-датчика	0.001 оборота	DINT	R	SK 5xxP SK 54xE
_145_Actual_pos	Текущее положение	0.001 оборота	DINT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_146_Actual_ref_pos	Текущая уставка положения	0.001 оборота	DINT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E
_147_Actual_pos_diff	Расхождение положений между уставкой и факт.знач.	0.001 оборота	DINT	R	SK 5xxP SK 54xE SK 53xE SK 52xE SK 2xxE SK 180E SK 190E

3.5.6 Ошибки ПЛК

Метки ошибок пользователя User Error Flags могут описывать ошибки устройства из программы ПЛК от E23.0 до E24.7.

Имя	Функция	Нормирование	Тип	Доступ	Устройство
_110_ErrorFlags	Ошибка пользователя на устройстве	Бит 0: E 23.0 бит 1: E 23.1 бит 2: E 23.2 бит 3: E 23.3 бит 4: E 23.4 бит 5: E 23.5 бит 6: E 23.6 бит 7: E 23.7	BYTE	R/W	все
_111_ErrorFlags_ext	Ошибка пользователя на устройстве	Бит 0: E 24.0 бит 1: E 24.1 бит 2: E 24.2 бит 3: E 24.3 бит 4: E 24.4 бит 5: E 24.5 бит 6: E 24.6 бит 7: E 24.7	BYTE	R/W	все

3.5.7 Параметры ПЛК

Эта группа данных процессов обеспечивает прямой доступ к параметрам ПЛК P355, P356 и P360.

Имя	Функция	Нормирование	Тип	Доступ	Устройство
_115_PLC_P355_1	ПЛК INT параметр P355 [-01]	-	INT	R	все
_116_PLC_P355_2	ПЛК INT параметр P355 [-02]	-	INT	R	все
_117_PLC_P355_3	ПЛК INT параметр P355 [-03]	-	INT	R	все
_118_PLC_P355_4	ПЛК INT параметр P355 [-04]	-	INT	R	все
_119_PLC_P355_5	ПЛК INT параметр P355 [-05]	-	INT	R	все
_120_PLC_P355_6	ПЛК INT параметр P355 [-06]	-	INT	R	все
_121_PLC_P355_7	ПЛК INT параметр P355 [-07]	-	INT	R	все
_122_PLC_P355_8	ПЛК INT параметр P355 [-08]	-	INT	R	все
_123_PLC_P355_9	ПЛК INT параметр P355 [-09]	-	INT	R	все
_124_PLC_P355_10	ПЛК INT параметр P355 [-10]	-	INT	R	все
_125_PLC_P356_1	ПЛК LONG параметр P355 [-01]	-	DINT	R	все
_126_PLC_P356_2	ПЛК LONG параметр P355 [-02]	-	DINT	R	все
_127_PLC_P356_3	ПЛК LONG параметр P355 [-03]	-	DINT	R	все
_128_PLC_P356_4	ПЛК LONG параметр P355 [-04]	-	DINT	R	все
_129_PLC_P356_5	ПЛК LONG параметр P355 [-05]	-	DINT	R	все
_130_PLC_P360_1	ПЛК параметр индикации P355 [-01]	-	DINT	R/W	все
_131_PLC_P360_2	ПЛК параметр индикации P355 [-02]	-	DINT	R/W	все
_132_PLC_P360_3	ПЛК параметр индикации P355 [-03]	-	DINT	R/W	все
_133_PLC_P360_4	ПЛК параметр индикации P355 [-04]	-	DINT	R/W	все
_134_PLC_P360_5	ПЛК параметр	-	DINT	R/W	все

Имя	Функция	Нормирование	Тип	Доступ	Устройство
	индикации P355 [-05]				
_135_PLC_Scope_Int_1	ПЛК Scope Отображаемое значение 1	-	INT	R/W	все
_136_PLC_Scope_Int_2	ПЛК Scope Отображаемое значение 2	-	INT	R/W	все
_137_PLC_Scope_Int_3	ПЛК Scope Отображаемое значение 3	-	INT	R/W	все
_138_PLC_Scope_Int_4	ПЛК Scope Отображаемое значение 4	-	INT	R/W	все
_139_PLC_Scope_ Bool_1	ПЛК Scope Отображаемое значение 5	-	INT	R/W	все
_140_PLC_Scope_ Bool_2	ПЛК Scope Отображаемое значение 6	-	INT	R/W	все

3.6 Языки

3.6.1 Список инструкций (AWL / IL)

3.6.1.1 Общие сведения

Тип данных

ПЛК поддерживает приведенные ниже типы данных.

Имя	Занимаемая память	Диапазон значений
BOOL	1 бит	от 0 до 1
BYTE	1 байт	от 0 до 255
INT	2 байта	от -32768 до 32767
DINT	4 байта	от -2 147 483 648 до 2 147 483 647
LABEL_ADDRESS	2 байта	Метка перехода

Литералы

Для лучшей наглядности константы всех типов данных могут быть заданны в различных формах представления. В нижеследующей таблице приводятся все возможные варианты.

Литерал	Пример	Число в десятичном выражении
Bool	FALSE	0
	TRUE	1
	BOOL#0	0
	BOOL#1	1
Двоичное (основа 2)	2#01011111	95
	2#0011_0011	51
	BYTE#2#00001111	15
	BYTE#2#0001_1111	31
Восьмеричное (основа 8)	8#0571	377
	8#05_71	377
	BYTE#8#10	8
	BYTE#8#111	73
	BYTE#8#1_11	73
Шестнадцатеричное (основа 16)	16#FFFF	-1
	16#0001_FFFF	131071
	INT#16#1000	4096
	DINT#16#0010_2030	1056816
Целочисленное (основа 10)	10	10
	-10	-10
	10_000	10000
	INT#12	12
	DINT#-100000	-100000
Время	TIME#10s50ms	10,050 секунд
	T#5s500ms	5,5 секунд
	TIME#5.2s	5,2 секунд
	TIME#5D10H15M	5дней+10часов+15минут
	T#1D2H30M20S	1день+2часа+30минут+20секунд

Комментарии

Для лучшего понимания содержания программы для ПЛК в последующем рекомендуется снабжать ее разделы комментариями. В пользовательской программе такие комментарии выделяются символами „(*“ в начале и символами „*)“ в конце, согласно нижеследующему примеру.

```
(* Комментарий к блоку программы *)
LD 100 (* Комментарий после команды *)
ADD 20
```

Метка перехода

Операторы JMP, JMPC или JMPCN позволяют перепрыгивать целые разделы программы. В качестве целевой точки таких переходов служит метка перехода. Метка может содержать любые буквы, кроме „ß“ и умлаутов, цифры от 0 до 9 и нижнее подчеркивание. Использование других символов не допускается. В конце метки ставится двоеточие. Она может стоять сама по себе. Также после нее, в той же строке, может находиться другая команда.

Возможные варианты представлены далее:

Пример:

```
Метка перехода:
LD 20

Das_Ist_eine_Sprungmarke:
ADD 10

MainLoop: LD 1000
```

Другой вариант: передача метки перехода в качестве переменной. Для этого переменная должна быть описана в таблице переменных с типом LABLE_ADDRESS, чтобы в эту переменную можно было загружать метки перехода. Этот способ позволяет легко создавать машины состояний, см. ниже.

Пример:

```
LD FirstTime
JMPC AfterFirstTime
(* Перед началом необходимо инициализировать адрес метки. *)
LD Address_1
ST Address_Var
LD TRUE
ST FirstTime
AfterFirstTime:
JMP Address_Var
Address_1:
LD Address_2
ST Address_Var
JMP Ende
Address_2:
LD Address_3
ST Address_Var
JMP Ende
Address_3:
LD Address_1
ST Address_Var
Окончание:
```

Вызов функции

Редактор поддерживает одну форму вызова функции. В представленных далее вариантах производится вызов функции CTD с помощью экземпляра объекта I_CTD. Результат сохраняется в переменных. Значение использованных функций описывается в руководстве далее.

Пример:

```
LD 10000
ST I_CTD.PV
LD LoadNewVar
ST I_CTD.LD
LD TRUE
ST I_CTD.CD
CAL I_CTD
LD I_CTD.Q
ST ResultVar
LD I_CTD.CV
ST CurrentCountVar
```

Побитовый доступ к переменным

Для доступа к одному биту из переменной или переменной процесса может быть использован упрощенный способ записи.

Команда	Значение
`LD Var1.0`	загружает бит 0 из Var1 в AE
`ST Var1.7`	сохраняет AE в бит 7 Var1
`EQ Var1.4`	сравнивает AE с бит4 из Var1

3.6.2 Структурированный текст (ST)

Структурированный текст состоит из ряда операторов, выполняемых по условию ("IF..THEN..ELSE) или в цикле (WHILE..DO), аналогично языкам высокого уровня.

Пример:

```
IF value < 7 THEN
  WHILE value < 8 DO
    value := value + 1;
  END_WHILE;
END_IF;
```

3.6.2.1 Общие сведения

Типы данных в ST

ПЛК поддерживает приведенные ниже типы данных.

Имя	Занимаемая память	Диапазон значений
BOOL	1 бит	от 0 до 1
BYTE	1 байт	от 0 до 255
INT	2 байт	от -32768 до 32767
DINT	4 байт	от -2 147 483 648 до 2 147 483 647

Информация

Для повышения эффективности программы для ПЛК рекомендуется указывать числа вместе с типом данных, например: VarInt := INT#-32768, VarDINT := DINT#-2147483648.

Операторы присваивания

В левой части оператора присваивания находится операнд (переменная, адрес), которому будет присвоено значение выражения из правой части при помощи оператора ":=".

Пример:

```
Var1 := Var2 * 10;
```

После выполнения данной строки переменная Var1 принимает значение Var2, умноженное на десять.

Вызов функциональных блоков в ST

Вызов функционального блока на языке ST производится путем указания имени экземпляра объекта ФБ и требуемого значения параметра в скобках. В следующем примере производится вызов таймера с назначением IN и PT для его параметров. Затем переменная результата Q назначается переменной A.

Обращение к переменной результата, также как на языке AWL, производится с помощью имени ФБ, соединительной точки и имени переменной.

Пример:

```
Timer(IN := TRUE, PT := 300);
A := Timer.Q;
```

Обработка выражений

Обработка выражения производится путем обработки операторов согласно определенным правилам связывания. Сначала выполняется обработка операторов с сильным связыванием, а затем оператора менее сильным связывание и т.д., пока не будут обработаны все операторы. Операторы с одинаковой силой связывания обрабатываются слева направо.

Далее представлена таблица операторов языка ST в порядке силы связывания.

Операция	Символ	Сила связывания
Заключение в скобки	(Выражение)	Сильное связывание
Вызов функции	Имя функции (список параметров)	
Инвертированное дополнение	NOT	
Умножение Деление Модуль AND	* / MOD AND	
Сложение Вычитание OR XOR	+ - OR XOR	
Сравнение Равенство Неравенство	<,>,<=,>= = <>	Слабое связывание

3.6.2.2 Операторы

Return

Оператор RETURN может использоваться для перехода к концу программы, например, при выполнении какого-либо условия.

IF

Оператор IF позволяет проверять выполнение условия и в зависимости от этого выполнять другие операторы.

Синтаксис:

```
IF <Булево_выражение1> THEN
  <IF_оператор>
ELSIF <Булево_выражение2> THEN
  <ELSIF_Оператор1>
ELSIF <Булево_выражение n> THEN
  <ELSIF_Оператор n-1>
ELSE
  <ELSE_Оператор>}
END_IF;
```

Часть в фигурных скобках {} является опциональной.

Если результатом <Булево_выражение1> является TRUE, то выполняется <IF_оператор>, а все последующие операторы пропускаются. В противном случае производится проверка по порядку все булевых выражений, начиная с <Булево_выражение2>, пока одно из них не даст результат TRUE. После этого проверяются только операторы после такого булево выражения и до следующего ELSE или ELSIF. Если ни одно булево выражение не принимает значение TRUE, то обрабатываться будут только <ELSE_операторы>.

Пример:

```
IF temp < 17 THEN
  Bool1 := TRUE;
ELSE
  Bool2 := FALSE;
END_IF;
```

CASE

Оператор CASE позволяет объединить несколько условных операторов для одной и той же условной переменной в одну логическую структуру.

Синтаксис:

```
CASE <Var1> OF
  <Значение 1>: <Оператор 1>
  <Значение 2>: <Оператор 2>
  <Значение3, Значение4, Значение5: <Оператор 3>
  <Значение6 .. Значение10 : <Оператор 4>
  ...
  <Значение n>: <Оакрвтор n>
ELSE <ELSE-Оператор>
END_CASE;
```

Оператор CASE обрабатывается по следующей схеме:

- Если переменная <Var1> принимает значение <Значение i>, то выполняется оператор <Оператор i>
- Если <Var 1> не принимает одно из заданных значений, выполняется <ELSE-оператор>.
- Если при нескольких значениях переменных выполняется один и тот же оператор, то можно записать несколько значений через запятую, объединив их в общее условие для оператора.
- Если данный оператор выполнятся для определенного диапазона значений переменной, то начальное и конечное значение следует указать друг за другом через двоеточие, объединив их в общее условие для оператора.

Пример:

```
CASE INT1 OF
  1, 5:
    BOOL1 := TRUE;
    BOOL3 := FALSE;
  2:
    BOOL2 := FALSE;
    BOOL3 := TRUE;
  10..20:
    BOOL1 := TRUE;
    BOOL3:= TRUE;
  ELSE
    BOOL1 := NOT BOOL1;
    BOOL2 := BOOL1 OR BOOL2;
END_CASE;
```

Цикл FOR

Цикл FOR позволяет запрограммировать повторяющиеся процедуры.

Синтаксис:

```
FOR <INT_Var> := <INIT_ЗНАЧ> TO <END_ЗНАЧ> {BY <размер шага>} DO
  <Операторы>
END_FOR;
```

Часть в фигурных скобках {} является опциональной. <Операторы> выполняются до тех пор, пока счетчик <INT_Var> не превысит конечное значение <END_ЗНАЧ>. Условие проверяется перед выполнением <Операторов>, поэтому <Операторы> никогда не будут выполнены, если начальное значение <INIT_ЗНАЧ> больше конечного <END_ЗНАЧ>. Всегда, когда <Операторы> выполняются, значение <INT_Var> увеличивается на <размер шага>. Размер шага может быть любым целым значением. Если он не задан, то по умолчанию принимает значение 1. При этом цикл нужно будет прервать, так как <INT_Var> будет только увеличиваться.

Пример:

```
FOR Счетчик :=1 TO 5 BY 1 DO
  Var1 := Var1 * 2;
  END_FOR;
```

Цикл REPEAT

Цикл REPEAT отличается от цикла WHILE тем, что условие выхода из цикла проверяется только после выполнения цикла. В результате этого цикл будет выполнен как минимум один раз, независимо от условия выхода.

Синтаксис:

```
REPEAT
  <Операторы>
UNTIL <Булево выражение>
END_REPEAT;
```

<Операторы выполняются до тех пор, пока <Булево выражение> не примет значение TRUE. Если <Булево выражение> принимает значение TRUE уже при первой проверке, то <Операторы> также выполняются ровно один раз. Если <Булево _выражение> никогда не принимает значение TRUE, то <Операторы> повторяться бесконечно, что приведет к ошибке периода выполнения.

Информация

Программист должен самостоятельно следить за тем, чтобы циклы не были бесконечными, когда он изменяет условие в разделе операторов цикла, например, увеличении или уменьшении счетчика.

Пример:

```
REPEAT
  Var1 := Var1 * 2;
  Счетчик := Счетчик - 1;
UNTIL
  Счетчик = 0
END_REPEAT
```

Цикл WHILE

Цикл WHILE может использоваться как и цикл FOR, с тем отличием, что условием его прерывания может быть любое булево выражение. То есть задается условие, при наступлении которого цикл выполняется.

Синтаксис:

```
WHILE <Булево выражение> DO
  <Операторы>
END_WHILE;
```

<Операторы выполняются до тех пор, пока <Булево выражение> не примет значение FALSE. Если <Булево выражение> принимает значение FALSE уже при первом выполнении, то <Операторы> также выполняются ровно один раз. Если <Булево _выражение> никогда не принимает значение FALSE, то <Операторы> повторяться бесконечно, что приведет к ошибке периода выполнения.

Информация

Программист должен самостоятельно следить за тем, чтобы циклы не были бесконечными, когда он изменяет условие в разделе операторов цикла, например, увеличении или уменьшении счетчика.

Пример:

```
WHILE Счетчик >0 DO
  Var1 := Var1 * 2;
  Счетчик := Счетчик - 1;
END_WHILE
```

Exit

Когда команда EXIT срабатывает при выполнении цикла FOR, WHILE или REPEAT, выполнение самого внутреннего цикла заканчивается, независимо от условия прерывания.

3.7 Переходы

3.7.1 JMP

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

Безусловный переход к метке перехода.

Пример на AWL:

```
JMP NextStep (* Безусловный переход к NextStep *)
ADD 1

NextStep:
ST Value1
```

3.7.2 JMPC

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

Условный переход (Jump Conditional) к метке перехода. Если AE = TRUE то команда JMPC выполняет переход к заданной метке.

Пример на AWL:

```
LD 10
JMPC NextStep (* AE = TRUE - программа выполняет переход*)
ADD 1

NextStep:
ST Value1
```

3.7.3 JMPCN

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

Условный переход (Jump Conditional) к метке перехода. JMPCN выполняет переход если регистр AE = FALSE. В противном случае программа продолжает выполнение последующих выражений.

Пример на AWL:

```
LD 10
JMPCN NextStep (* AE = TRUE à программа не выполняет переход*)
ADD 1

NextStep:
ST Value1
```

3.8 Конвертация типов

3.8.1 BOOL_TO_BYTE

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных	X			

Конвертирует тип данных AE из BOOL в BYTE. Если AE равна FALSE, Akku принимает значение 0. Если AE равна TRUE, Akku принимает значение 1.

Пример на AWL:

```
LD TRUE
BOOL_TO_BYTE (* AE = 1 *)
```

Пример на ST:

```
Результат := BOOL_TO_BYTE(TRUE); (* Результат = 1 *)
```

3.8.2 BYTE_TO_BOOL

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X		

Конвертирует тип данных из BYTE в BOOL. Пока BYTE не равно нулю, результатом конвертирования всегда будет TRUE.

Пример на AWL:

```
LD 10
BYTE_TO_BOOL (* AE = TRUE *)
```

Пример на ST:

```
Результат := BYTE_TO_BOOL(10); (* Результат = TRUE *)
```

3.8.3 BYTE_TO_INT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных		X		

Конвертирует тип данных из BYTE в INT. Переменная BYTE копируется в нижнюю часть переменной INT, а в верхнюю часть INT записывается 0.

Пример на AWL:

```
LD 10
BYTE_TO_INT (* Akku = 10 *)
```

Пример на ST:

```
Результат := BYTE_TO_INT(10); (* Результат = 10 *)
```

3.8.4 DINT_TO_INT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных				X

Конвертирует тип данных из DINT в INT. При этом верхняя часть значения переменной DINT не переносится.

Пример на AWL:

```
LD 200000
DINT_TO_INT     (* Akku = 3392 *)

LD DINT# -5000
DINT_TO_INT     (* Akku = -5000 *)

LD DINT# -50010
DINT_TO_INT     (* Akku = 15526 *)
```

Пример на ST:

```
Результат := DINT_TO_INT(200000); (* Результат = 3392 *)
Результат := DINT_TO_INT(-5000); (* Результат = -5000 *)
Результат := DINT_TO_INT(-50010); (* Результат = 15526 *)
```

3.8.5 INT_TO_BYTE

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных			X	

Конвертирует тип данных из INT в BYTE. При этом верхняя часть значения переменной INT не переносится. Знак теряется, так как тип BYTE не имеет знака.

Пример на AWL:

```
LD 16#5008
INT_TO_BYTE (* Akku = 8 *)
```

Пример на ST:

```
Результат := INT_TO_BYTE(16#5008); (* Результат = 8 *)
```

3.8.6 INT_TO_DINT

	SK 5xxP	SK 54xE	SK 53xE SK 52xE	SK 2xxE	SK 2xxE-FDS	SK 180E SK 190E	SK 155E-FDS SK 175E-FDS
Наличие	X	X	X	X	X	X	X

	BOOL	BYTE	INT	DINT
Тип данных			X	

Конвертирует тип данных из INT в DINT. Переменная INT копируется в нижнюю часть переменной DINT, а в верхнюю часть DINT записывается 0.

Пример на AWL:

```
LD 10
INT_TO_DINT (* Akku = 10 *)
```

Пример на ST:

```
Результат := INT_TO_DINT(10); (* Результат = 10 *)
```

3.9 Сообщения о неисправностях ПЛК

Чтобы не допустить повреждения, при возникновении ошибки устройство отключается. При появлении сообщении о неисправности ПЛК его работа останавливается и ПЛК переводится в состояние „PLC-Error“. В случае появления других сообщений о неисправностях работа ПЛК продолжается. После обработки ошибки работа ПЛК возобновляется автоматически.

При возникновении ошибки ПЛК PLC User Fault 23.X работа ПЛК продолжается!

Вывод на дисплее SimpleBox		**Неисправность** **Текстовое сообщение в Parameter Box**	**Причина** **Способ устранения**
Группа	**Описание в P700 [-01] / P701**		
E022	**22.0**	**Отсутствует программа для ПЛК**	ПЛК запущен, но на ЧП отсутствует программа для ПЛК. - Загрузить программу для ПЛК на устройство
	22.1	**Ошибка программы для ПЛК**	В результате проверки контрольных сумм программой ПЛК произошла ошибка. - Заново запустить устройство (Power ON) и повторить попытку - или заново загрузить программу для ПЛК
	22.2	**Неверный адрес перехода**	Ошибка программы, аналогично ошибке 22.1
	22.3	**Переполнен стэк**	При выполнении программы было открыто более 6 уровней скобок. - Проверить программу на наличие ошибок периода выполнения
	22.4	**Превышено макс. время цикла ПЛК**	Превышена заданная максимальная продолжительность цикла программы ПЛК. - Привести в соответствие продолжительность цикла или проверить программу
	22.5	**Неизвестный код команды**	Невозможно выполнить один из кодов команды, указанный в программе, так как такой код неизвестен. - Программная ошибка, аналогично ошибке 22.1, - Версия ПЛК не соответствует версии NORD CON
	22.6	**Доступ записи ПЛК**	В ходе выполнения программы для ПЛК ее содержание было изменено.
	22.9	**Суммарная ошибка ПЛК**	Невозможно точно установить причину ошибки. - Аналогично ошибке 22.1
E023	**23.0**	**PLC User Fault 1**	Ошибка может быть вызвана программой ПЛК, чтобы не подавать внешний сигнал о проблеме в ходе выполнения программы для ПЛК. Устранение выполняется посредством описания переменной процесса „ErrorFlags“.
	23.1	**PLC User Fault 2**	
	23.2	**PLC User Fault 3**	

4 Параметры

Параметры устройств, связанные с использованием функционала ПЛК, подробно описаны в руководстве соответствующего частотного преобразователя или пускового устройства двигателя.

5 Приложение

5.1 Указания по техническому обслуживанию и вводу в эксплуатацию

В случае затруднений, возникающих, например, при вводе в эксплуатацию, просим обращаться в нашу техническую службу:

☎ +49 4532 289-2125

Наша техническая служба работает круглосуточно 7 дней в неделю. Чтобы мы могли вам помочь, просим предоставить следующую информацию об устройстве и его оснащении:

- Маркировка модели
- Серийный номер
- Версия встроенного ПО.

5.2 Документы и программы

Документы и программы можно загрузить на нашем веб-сайте www.nord.com.

Применяемые и дополнительные документы

Документация	Содержание
BU 0155	Руководство к периферийному устройству плавного пуска NORDAC *LINK* **SK 180E / SK 190E**
BU 0180	Руководство к частотному преобразователю NORDAC *BASE* **SK 180E / SK 190E**
BU 0200	Руководство к частотному преобразователю NORDAC *FLEX* **SK 200E .. SK 235E**
BU 0250	Руководство к периферийному частотному преобразователю NORDAC *LINK* **SK 250E-FDS .. SK 280E-FDS**
BU 0500	Руководство к частотному преобразователю NORDAC *PRO* **SK 500E .. SK 535E**
BU 0505	Руководство к частотному преобразователю NORDAC *PRO* **SK 540E .. SK 545E**
BU 0600	Руководство к частотному преобразователю NORDAC *PRO* **SK 500P .. SK 550P**
BU 0000	Руководство пользователя программы NORDCON
BU 0040	Руководство по работе с модулями параметризации NORD

Программное обеспечение

Программное обеспечение	Описание
NORDCON	Программа для параметризации и диагностики

5.3 Обозначения

- **AE** Текущее значение
- **AIN** Аналоговый вход
- **AOUT** Аналоговый выход
- **AWL** Список инструкций (также IL)
- **COB-ID** Идентификатор объекта связи (Communication Objekt Identifier)
- **DI / DIN** Цифровой вход
- **DO / DOUT** Цифровой выход
- **E/A или I/O** Вход/выход
- **EEPROM** Постоянное запоминающее устройство
- **EMV** Электромагнитная совместимость
- **FB/ФБ** Функциональный блок
- **FU/ЧП** Частотный преобразователь
- **HSW** Главная уставка
- **IL** Список инструкций (см. также AWL)
- **ISD** Ток намагничивания (управление вектором тока)
- **LED** Светодиод
- **MC** Модуль управления движением Motion Control
- **NSW** Вспомогательная уставка
- **P** Параметры, зависящие от набора параметров, т. е. параметры, которые могут принимать разные функции или значения в зависимости от того, в каком из четырех наборов они используются.
- **P-BOX** ParameterBox
- **PDO** Объект данных процессов (Prozess Daten Objekt)
- **ПЛК** Программируемый логический контроллер
- **S** Защищенный параметр, т. е. параметр, значение которого становится доступными только после ввода пароля в параметре **P003**.
- **SW** Версия ПО (см. параметр **P707**)
- **STW** Команда управления
- **ZSW** Команда состояния

Предметный указатель

С

NORD DRIVESYSTEMS Group

Headquarters and Technology Centre
in Bargteheide, close to Hamburg

Innovative drive solutions
for more than 100 branches of industry

Mechanical products
parallel shaft, helical gear, bevel gear and worm gear units

Electrical products
IE2/IE3/IE4 motors

Electronic products
centralised and decentralised frequency inverters,
motor starters and field distribution systems

7 state-of-the-art production plants
for all drive components

Subsidiaries and sales partners
in 98 countries on 5 continents
provide local stocks, assembly, production,
technical support and customer service

More than 4,000 employees throughout the world
create customer oriented solutions

www.nord.com/locator

Headquarters:

Getriebebau NORD GmbH & Co. KG
Getriebebau-Nord-Straße 1
22941 Bargteheide, Germany
T: +49 (0) 4532 / 289-0
F: +49 (0) 4532 / 289-22 53
info@nord.com, www.nord.com

Member of the NORD DRIVESYSTEMS Group

NORD®
DRIVESYSTEMS

6075507 / 1919