

Приложение

Данный раздел книги содержит сведения о полезных дополнениях к языкам программирования LAD и FBD, обзор содержимого библиотек блоков STEP 7 (STEP 7 Block Libraries) и обзор функций всех элементов LAD и FBD.

- Вы можете с помощью программы LAD/FBD обеспечить блоки **защитой (block protection)**. Для этой цели используется ориентированный на источник (исходные файлы) редактор языка программирования STL.
- Вы можете воспользоваться дополнительной функцией STL, **косвенной адресацией (indirect addressing)**, чтобы передавать области данных в языках программирования LAD и FBD; в этом случае адреса таких областей данных не вычисляются до выполнения программы. Библиотеки «LAD_Book» и «FBD_Book» на прилагаемой дискете содержат «Пример фрейма сообщения» («Sample Message Frame»), показывающий способ установки и передачи областей данных.
- Стандартный пакет STEP 7 включает в себя **библиотеки блоков (block libraries)** с загружаемыми функциями и функциональными блоками, а также с заголовками блоков и описаниями интерфейсов для системных блоков (SFC и SFB).
- Завершает книгу **обзор всех функций** LAD и FBD.

На прилагаемой к книге дискете вы можете найти архивные библиотеки «LAD_Book» и «FBD_Book». Эти библиотеки вы можете разархивировать в SIMATIC-менеджере с помощью команды File → Retrieve (Файл → Разархивировать). Выберите архив (дискету) в открывшемся диалоговом окне. В следующем поле можно определить директорию назначения. Вообще, библиотеки располагаются в директории ...\\STEP7\\S7LIBS; но вы можете выбрать любую другую директорию, например, ...\\STEP7\\S7PROJ, которая обычно содержит проекты. В последнем поле «Retrieve - Options» («Разархивирование - Опции») вы можете отменить опцию «Restore full path» («Восстановить полный путь»).

Библиотеки «LAD_Book» и «FBD_Book» содержат по восемь программ, являющиеся примерами представлений LAD и FBD. Два больших примера показывают программирование функций, функциональных блоков и локальных экземпляров (пример конвейера) и обработку данных (пример фрейма сообщения). Требуется примерно 1,93 Мб дискового пространства.

Чтобы опробовать пример, установите проект, который соответствует вашей конфигурации аппаратных средств, и скопируйте программу, включая таблицу символов, из библиотеки в проект. Теперь вы можете протестировать программу в онлайн-режиме.

24 Дополнения к графическому программированию

Защита блока; косвенная адресация; пример фрейма сообщения

25 Библиотеки блоков

Организационные блоки; системные функциональные блоки; функциональные блоки IEC; блоки преобразования S5-S7; блоки преобразования TI-S7; блоки управления PID; коммуникационные блоки

26 Обзор функций LAD

Все функции LAD

27 Обзор функций FBD

Все функции FBD

Содержание главы 24

<u>24</u>	<u>Дополнения к графическому программированию</u>	4
<u>24.1</u>	<u>Защита блока</u>	4
<u>24.2</u>	<u>Косвенная адресация</u>	6
<u>24.2.1</u>	<u>Указатели: общие замечания</u>	6
<u>24.2.2</u>	<u>Указатель области</u>	6
<u>24.2.3</u>	<u>DB-указатель</u>	8
<u>24.2.4</u>	<u>ANY-указатель</u>	8
<u>24.2.5</u>	<u>«Переменный» ANY-указатель</u>	10
<u>24.3</u>	<u>Краткое описание «Примера фрейма сообщения»</u>	11

24 Дополнения к графическому программированию

24.1 Защита блока

Защиту блока представляет ключевое слово `KNOW_HOW_PROTECT`. Блок с этим атрибутом нельзя просмотреть, распечатать или изменить. Редактор только отображает заголовок блока и таблицу описаний с параметрами блока. При ориентированном на источник (исходный файл) вводе вы сами можете защитить любой блок с помощью `KNOW_HOW_PROTECT`. Это означает, что никто, включая вас, не сможет просмотреть скомпилированный блок (сохраните исходный файл в безопасном месте!).

Вы можете ввести защиту блока `KNOW_HOW_PROTECT` при помощи STL, и она должна быть ориентированной на источник. Чтобы реализовать это, действуйте следующим образом:

- 1) Создайте блок в LAD или FBD обычным способом. Позднее этот блок будет перезаписан в пользовательской программе *Blocks* (Блоки) блоком с ключевым словом. Если вы хотите сохранить (исходный) блок (настоятельно рекомендуется при вводе защиты блока), то перед вводом ключевого слова вы можете сохранить блок, к примеру, в (созданной пользователем) библиотеке. Таким же образом вы также можете сохранить всю пользовательскую программу.
- 2) Создайте контейнер исходных файлов. Если в *S7 program* (S7-программа) нет объекта *Source Files* (Исходные файлы) (на том же уровне, что и программа пользователя *Blocks*), вы должны создать его. Выберите *S7 program* и вставьте объект *Source Files* с помощью команды меню Insert → S7 Software → Source Directory (Вставка → Программное обеспечение S7 → Директория исходных файлов).
- 3) Из блока сгенерируйте исходный файл STL. Перейдите в редактор (через панель задач, например, или откройте любой блок в *Blocks* и затем закройте его) и выберите пункт меню File → Generate Source File (Файл → Генерирование исходного файла). В появившемся диалоговом окне установите ваш проект, выберите объект *Source Files* и назначьте имя исходному файлу в поле «Object Name» («Имя объекта»). Нажмите «ОК». В следующем диалоговом окне вам будут показаны все блоки в контейнере *Blocks*; отметьте блок(и), из которого вы хотите создать исходный файл. Нажмите «ОК».
- 4) Откройте исходный файл (например, двойным щелчком на символе исходного файла в SIMATIC-менеджере или с помощью редактора и команды File → Open, Файл → Открыть). Теперь вы можете увидеть исходный ASCII-файл вашего LAD/FBD-блока. Если вы предварительно отметили несколько блоков, то эти блоки будут скомпонованы по порядку в исходном файле.

Ввод для кодовых блоков осуществляется в следующем порядке:

- Ключевое слово для типа блока (FUNCTION, FUNCTION_BLOCK, ORGANIZATION_BLOCK) с указанием адреса. Затем может следовать заголовок блока (начинающийся с TITLE=...) и комментарий блока (начинающийся с //...).
- Атрибуты блока (в зависимости от того, заполнили вы поля в окне свойств (Properties) блока или нет; если да, то сколько).
- Описание (объявление) переменных (несколько разделов с ключевыми словами VAR_xxx, ..., END_VAR); оно зависит от того, объявлены локальные блочные переменные, и если так, то какие.
- Программа, начинающаяся с BEGIN и заканчивающаяся ключевым словом, определяющим конец блока (к примеру, END_FUNCTION_BLOCK).

Ввод для блоков данных производится в следующем порядке:

- Ключевое слово для типа блока (DATA_BLOCK) с указанием адреса.
 - Атрибуты блока (зависящие от того, заполнили вы поля в окне свойств (Properties) блока или нет; если да, то сколько).
 - Описание переменных (начинающееся со STRUCT и заканчивающееся END_STRUCT) или, в случае экземплярных блоков данных, адрес соответствующего функционального блока.
 - Инициализация переменных, начинающаяся с BEGIN и заканчивающаяся END_DATA_BLOCK.
- 5) В исходном файле вводится ключевое слово KNOW_HOW_PROTECT во всех случаях в своей строке после атрибутов блока и перед объявлением переменных. Если вы создали исходный файл из нескольких блоков, введите ключевые слова во все выбранные блоки. В заключении сохраните исходный файл.
 - 6) Откомпилируйте исходный файл, вызвав команду меню File → Compile (Файл → Компилировать). Компилятор создаст блок с определенными атрибутами (в используемом для создания языке STL в случае кодовых блоков; используемый для создания язык здесь значения не имеет, так как KNOW_HOW_PROTECT означает, что блок нельзя будет просмотреть или распечатать). Компилированный (новый) блок находится в пользовательской программе *Blocks* и заменяет (старый) блок с тем же номером.

24.2 Косвенная адресация

Язык программирования STL предоставляет вам метод доступа к операндам, адреса которых не вычисляются до исполнения программы. В меньшей степени это возможно в LAD или FBD: вы можете подождать до выполнения программы, чтобы определить, какие области данных требуется скопировать при помощи SFC 20 BLKMOV.

Впрочем, сначала немного полезной информации об указателях.

24.2.1 Указатели: общие замечания

Для косвенной адресации вам потребуется формат данных, содержащий адрес бита, а также адрес байта и, если применимо, область операнда. Этим форматом данных является *указатель (pointer)*. Указатель также используется для ссылки на операнд. Имеется три вида указателей:

- указатели областей (area pointers); их размер – 32 бита, и они содержат определенный операнд или его адрес;
- DB-указатели (DB pointers); их размер составляет 48 битов, и вдобавок к указателю области они также содержат номер блока данных;
- ANY-указатели (ANY pointers); они имеют размер 80 битов и в дополнение к DB-указателю содержат дополнительную информацию, например, тип данных операнда.

24.2.2 Указатель области

Указатель области содержит адрес операнда и, если применимо, область операнда. Без области операнда это внутренний указатель области (area-internal); если указатель содержит также область операнда, то он называется указателем пересечения области (area-crossing).

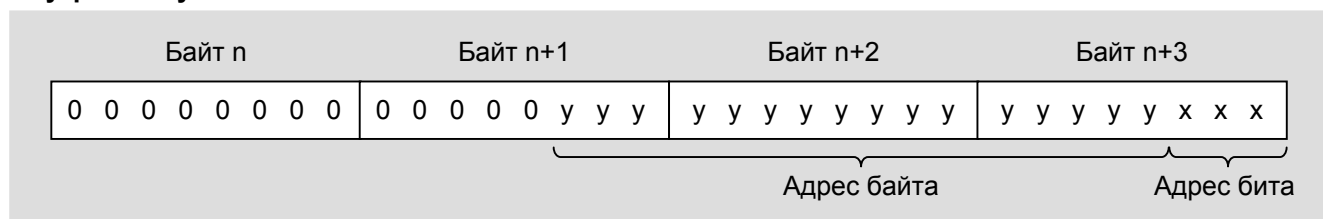
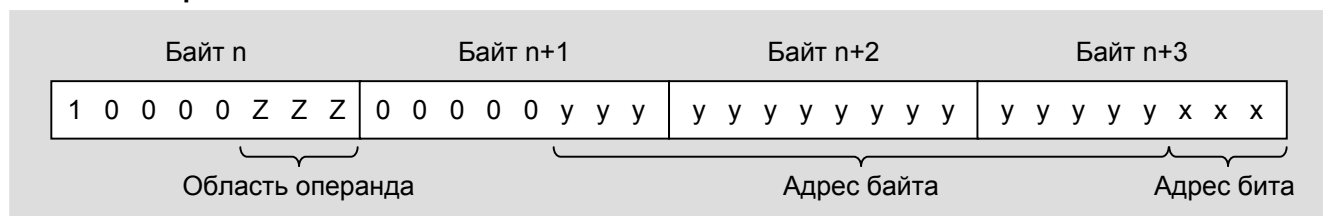
Форма записи представления константы следующая:

R#y.x для внутреннего указателя области, например, R#22.0;

R#Zy.x для указателя пересечения области, например, R#M22.0,

где x = адрес бита, y = адрес байта, а Z = область. Вы можете задать идентификатор (ID) операнда в качестве области. Два типа указателей различаются по значению 31-го бита.

На рисунке 24.1 показаны все типы указателей и их содержимое, как это реализовано в STEP 7.

Внутренний указатель области**Указатель пересечения области**

		ANY-указатель для типов данных	ANY-указатель для таймеров / счетчиков	ANY-указатель для блоков
ДВ-указатель Байт n Байт n+1 Байт n+2 Байт n+3 Байт n+4 Байт n+5	Номер	Байт n 16#10	Байт n 16#10	Байт n 16#10
	блока данных	Байт n+1 Тип	Байт n+1 Тип	Байт n+1 Тип
		Байт n+2 Количество	Байт n+2 Количество	Байт n+2 Количество
		Байт n+3 Количество	Байт n+3 Количество	Байт n+3 Количество
	Указатель	Байт n+4 Номер	Байт n+4 16#0000	Байт n+4 16#0000
	области	Байт n+5 блока данных	Байт n+5 16#0000	Байт n+5 16#0000
		Байт n+6 Указатель	Байт n+6 Тип	Байт n+6 16#0000
		Байт n+7 Указатель	Байт n+7 16#00	Байт n+7 16#0000
		Байт n+8 области	Байт n+8 Номер	Байт n+8 Номер
		Байт n+9 Номер	Байт n+9 Номер	Байт n+9 Номер

Область адресов:

- 0 0 0 Периферийные входы/выходы (P)
- 0 0 1 Входы (I)
- 0 1 0 Выходы (Q)
- 0 1 1 Меркеры (M)
- 1 0 0 Глобальные данные (DBX)
- 1 0 1 Экземплярные данные (DIX)
- 1 1 0 Временные локальные данные (L) ¹⁾
- 1 1 1 Временные локальные данные вызывающего блока (V) ²⁾

¹⁾ Не с адресацией пересечения области²⁾ Только с передачей параметров блока**Тип в ANY-указателе:****Простые типы данных**

- 01 BOOL
- 02 BYTE
- 03 CHAR
- 04 WORD
- 05 INT
- 06 DWORD
- 07 DINT
- 08 REAL
- 09 DATE
- 0A TOD
- 0B TIME
- 0C S5TIME

Сложные типы данных

- 0E DT
- 13 STRING

Параметрические типы

- 17 BLOCK_FB
- 18 BLOCK_FC
- 19 BLOCK_DB
- 1A BLOCK_SDB
- 1C COUNTER
- 1D TIMER

Нулевой указатель

- 00 NIL

Рисунок 24.1 Структура указателей в STEP 7

Указатель области имеет, в принципе, адрес бита, который всегда должен быть определен даже с числовыми операндами; в случае числовых операндов укажите 0 в качестве адреса бита. Пример: вы можете использовать указатель области P#M22.0, чтобы адресовать (меркерный) бит памяти M 22.0, но, кроме того, байт памяти MB 22, слово памяти MW 22 или двойное слово памяти MD 22.

24.2.3 DB-указатель

В дополнение к указателю области DB-указатель также содержит номер блока данных в виде положительного числа INT. Он определяет блок данных, если указатель области ссылается на глобальные данные или область экземплярных данных. Во всех других случаях первые два байта содержат ноль.

Форма записи указателя знакома вам по полной адресации операндов данных. Здесь также определяются блок данных и операнд данных, разделенные точкой: P#DataBlock.DataOperand (P#БлокДанных.ОперандДанных).

Пример: P#DB 10.DBX 20.5

Вы можете применить этот указатель к параметру блока параметрического типа POINTER для ссылки на операнд данных. Редактор использует этот тип указателя в своей работе для передачи фактических параметров.

24.2.4 ANY-указатель

Кроме DB-указателя ANY-указатель также содержит тип данных и коэффициент повторения. Это дает возможность указывать на область данных.

ANY-указатель доступен в двух вариантах: для переменных с типами данных и для переменных параметрических типов. Если вы ссылаетесь на переменную с типом данных, ANY-указатель содержит DB-указатель, тип и коэффициент повторения. Если ANY-указатель ссылается на переменную параметрического типа, он содержит вместо DB-указателя только номер в дополнение к типу. В случае функции таймера или счетчика тип повторяется в байте (n + 6); байт (n + 7) содержит B#16#00. В остальных случаях эти два байта содержат значение W#16#0000.

В первом байте ANY-указателя хранится синтаксический ID; в STEP 7 он всегда равен 10hex. Тип определяет тип данных переменной, для которой применяется ANY-указатель. Переменные простых типов, DT и STRING приобретают указанный тип и количество 1.

Если вы в параметре ANY применяете переменную типа ARRAY или STRUCT (а также UDT), то редактор генерирует ANY-указатель на поле или структуру. Этот ANY-указатель содержит ID для BYTE (02hex) в качестве типа и байтовый размер переменной в качестве количества. Тип данных отдельного поля или компонента структуры здесь не важен. Таким образом, ANY-указатель ссылается с использованием удвоенного числа байтов на поле WORD. Исключение: указатель на поле, состоящее из компонентов типа CHAR, создается также с типом CHAR (03hex).

Вы можете применить ANY-указатель в параметре блока параметрического типа ANY, если вы хотите получить ссылку на переменную или область операнда. Представление константы для типов данных следующее:

`P#[DataBlock.]Operand Type Quantity` (`P#[БлокДанных.]Операнд Тип Количество`)

Примеры:

- `P#DB 11.DBX 30.0 INT 12`
Область с 12 словами в DB 11, начиная с DBB 30;
- `P#M 16.0 BYTE 8`
Область с 8 байтами, начиная с MB 16;
- `P#E 18.0 WORD 1`
Входное слово IW 18;
- `P#E 1.0 BOOL 1`
Вход I 1.0.

В случае параметрических типов указатели записываются в следующем виде:
`L#Number Type Quantity` (`L#Номер Тип Количество`)

Примеры:

- `L#10TIMER 1`
Функция таймера T 10;
- `L#2 COUNTER 1`
Функция счетчика C 2.

Затем редактор применяет ANY-указатель, который соответствует по типу и количеству со спецификацией в представлении константы. Обратите внимание, что для типов данных адрес операнда в ANY-указателе должен быть также адресом бита.

Определение постоянного ANY-указателя имеет смысл, если вы хотите обратиться к области данных, для которой не объявлено никаких переменных. В принципе вы также можете применить переменные или операнды в параметре ANY. Например, представление «`P#1 1.0 BOOL 1`» идентично «`I 1.0`» или соответствующему символическому адресу.

Если вы не указываете значений по умолчанию при описании (объявлении) параметра ANY в функциональном блоке, то редактор назначает 10hex синтаксическому ID и 00hex остальным байтам. Тогда редактор представляет (пустой) ANY-указатель (в режиме просмотра данных) в следующем виде: `P#P0.0 VOID 0`.

24.2.5 «Переменный» ANY-указатель

При копировании с использованием SFC 20 вы указываете в параметрах источника и назначения либо абсолютно адресованную область (к примеру, R#DB127.DBX0.0 BYTE 32), либо переменную. В обоих случаях исходная область и область назначения во время программирования фиксированы (переменная индексация невозможна даже для элементов массива). Доступен следующий метод для модификации в ходе выполнения программы области данных, созданной в параметре типа ANY.

Во временных локальных данных создается переменная типа данных ANY и используется для инициализации параметра ANY. Редактор программ в таком случае не генерирует ANY-указатель (как он поступил бы для другой переменной), но использует переменную ANY во временных локальных данных в качестве ANY-указателя на область-источник и область назначения. Переменная ANY во временных локальных данных структурирована точно так же, как ANY-указатель; теперь вы можете изменять отдельные вводы информации (записей) в ходе выполнения программы.

Процедура работает не только в случае SFC 20 BLKMOV, но и с параметрами типа ANY в других блоках.

Библиотеки «LAD_Book» и «FBD_Book» на прилагаемой к книге дискете содержат программу «Message Frame Example» («Пример фрейма сообщения»), которая, в свою очередь, содержит пример «переменного» ANY-указателя.

24.3 Краткое описание «Примера фрейма сообщения»

В первую очередь в примере рассматривается управление данными. Он разбит на разделы следующим образом:

- Данные фрейма сообщения; показана обработка структур данных;
- Опрос времени суток; показано управление системными и стандартными блоками;
- Редактирование фрейма сообщения; демонстрируется использование SFC 20 BLKMOV с фиксированными адресами;
- Непрямое копирование области данных; показано использование функции «непрямого копирования» с помощью «переменных» ANY-указателей;
- Сохранение фрейма сообщения; продемонстрировано применение «непрямого копирования».

На рисунке 24.2 представлена программа и структура данных для этого примера.

Данные фрейма сообщения

Пример показывает вам, как можно определить часто встречающиеся структуры данных в качестве своего собственного типа данных и как использовать этот тип данных при описании переменных и параметров.

Мы конструируем хранилище данных для входящих и исходящих фреймов сообщений: почтовый ящик передачи (Send mailbox) со структурой фрейма сообщения, почтовый ящик приема (Receive mailbox) той же структуры и кольцевой буфер (приема) для обеспечения промежуточного хранения для входящих фреймов сообщений. Так как структура фрейма сообщения встречается часто, мы определим ее как фрейм определенного пользователем типа данных (UDT). Фрейм сообщения содержит заголовок фрейма, структуру которого также требуется назвать. Почтовый ящик передачи (Send) и почтовый ящик приема (Receive) будут блоками данных, содержащими переменную структуры *frame* (*фрейм*). Наконец, имеется кольцевой буфер, блок данных с массивом из восьми компонентов, которые имеют такую же структуру данных, как и *frame* (*фрейм*).

Опрос времени суток

Пример демонстрирует обработку системных и стандартных блоков (выявление ошибок, копирование из библиотеки, переименование).

Функция считывания времени суток должна в качестве своего значения вывести время суток из встроенного в CPU таймера реального времени. Для этой цели нам потребуется системная функция SFC 1 READ_CLK, которая считывает дату и время

суток из таймера реального времени в формате данных DATE_AND_TIME или DT. Так как мы хотим прочитать только время суток, то также понадобится IEC-функция FC 8 DT_TOD. Эта функция получает время суток в формате TIME_OF_DAY или TOD из формата данных DT.

Пример фрейма сообщения

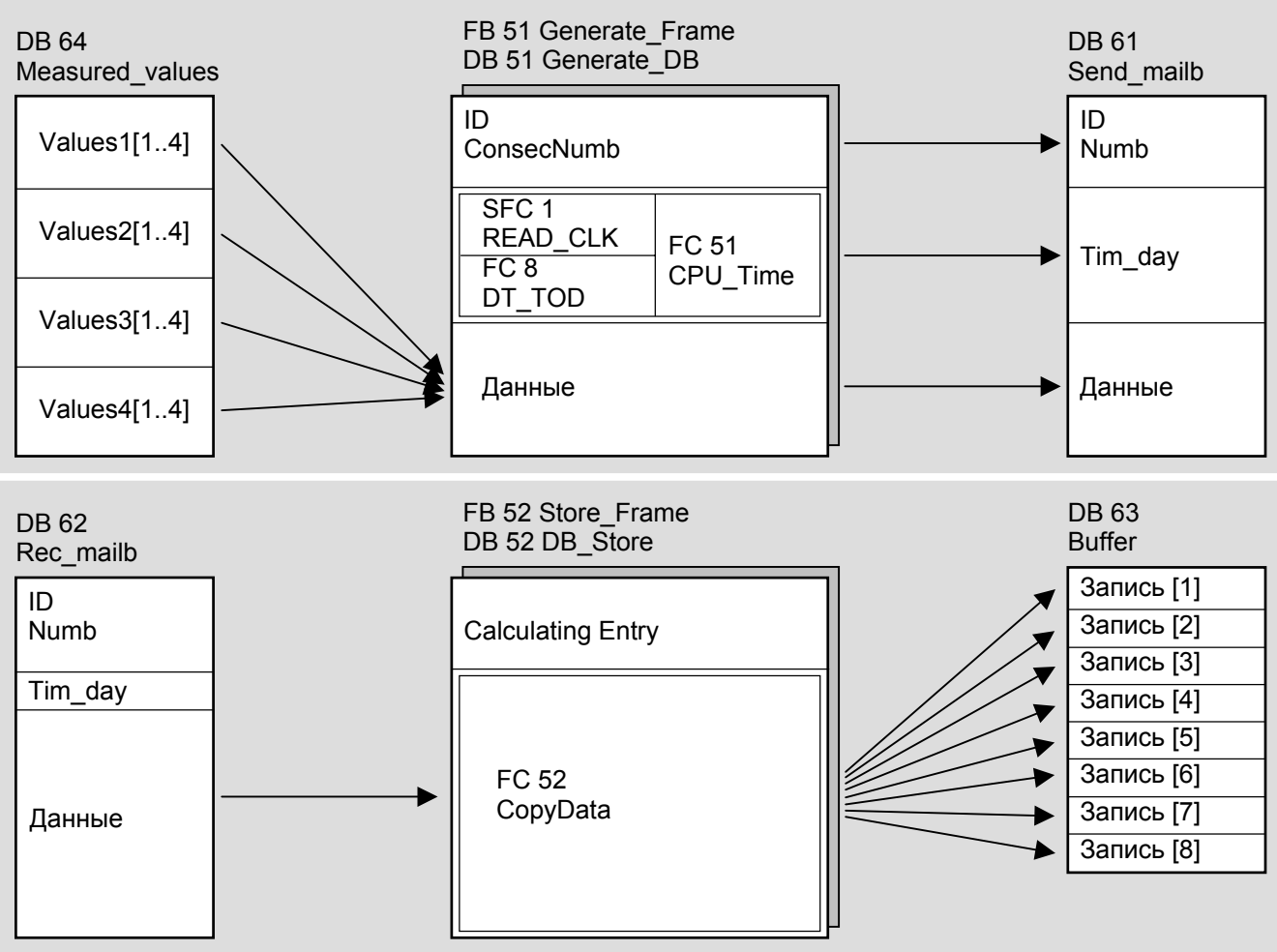


Рисунок 24.2 Структура данных для примера данных фрейма сообщения

Оценка ошибок

Системные функции сообщают об ошибке через параметр BR (бинарный результат) и значение функции RET_VAL. Ошибка возникла, если бинарный результат BR = «0»; значение функции в таком случае отрицательно (бит 15 установлен). Стандартные IEC-функции сигнализируют об ошибке только посредством бинарного результата. В примере показаны оба вида оценки ошибки. Если ошибка была выявлена, то выводится недействительное значение времени суток. Также устанавливается бинарный результат. После вызова опроса времени суток вы можете, таким образом, по бинарному результату определить, возникла ли ошибка.

Автономное программирование системных функций

Перед сохранением блока входа системная функция SFC 1 и стандартная функция FC 8 должны быть внесены в автономную пользовательскую программу. Обе функции включены в стандартный пакет STEP 7. Вы можете найти эти функции в предоставляемой библиотеке блоков. (Что касается системных функций, встроенных в CPU, библиотека содержит только описание их интерфейсов, а не фактическую программу системных функций. Функция может быть вызвана автономно через это описание интерфейса; описание интерфейса не передается в CPU. Загружаемые функции, такие как IEC-функции, доступны в библиотеке как исполняемые программы.)

Выберите стандартную библиотеку *Standard Library* с помощью команды File → Open → Library (Файл → Открыть → Библиотека) в SIMATIC-менеджере и откройте библиотеку *System Function Blocks* (Системные функциональные блоки). В *Blocks* (Блоки) вы можете найти все описания интерфейсов системных функций. Если окно вашего проекта все еще открыто, то можно отобразить два окна рядом, выбрав команду меню Window → Arrange → Vertically (Окно → Упорядочить → Вертикально), и перетащить с помощью мыши выбранные системные функции в вашу программу (отметьте мышью SFC, удерживая нажатой кнопку мыши, перетащите в *Blocks* или в ее открытое окно и отпустите). Таким же образом скопируйте стандартную функцию FC 8. Вы можете найти ее в библиотеке *IEC Function Blocks* (Функциональные блоки IEC). FC 8 является загружаемой функцией; вследствие этого она резервирует пользовательскую память в отличие от SFC 1.

Если стандартный функциональный блок вызывается из раздела «Libraries» («Библиотеки») каталога программных элементов (Program Element Catalog) при помощи редактора, то он автоматически копируется в *Blocks* и вносится в таблицу символов.

Переименование стандартных функций

Вы можете переименовать загружаемую стандартную функцию. Выберите стандартную функцию (например, FC 8) в окне проекта и щелкните (еще раз) на обозначении. Имя отобразится в рамке, и вы сможете указать новый адрес (например, FC 98). Если вы нажмете клавишу F1, пока отмечена стандартная функция (переименованная в FC 98), то вы тем не менее получите онлайн-помощь по исходной стандартной функции FC 8.

Если при копировании имеется идентично адресованный блок, то появляется диалоговое окно, где вы можете выбрать между перезаписью и переименованием.

Символический адрес

Присвоить имена системным и стандартным функциям вы можете в таблице символов с тем, чтобы также можно было обратиться к этим функциям символически. Вы можете произвольно назначить имена в рамках правил, применимых к именам блоков. В примере имя блока каждый раз выбирается как символическое имя (для лучшей идентификации).

Редактирование фрейма сообщения

Блок данных *Send_Mailb* должен быть заполнен данными фрейма сообщения. Мы используем функциональный блок, который имеет ID и последовательный номер, записанные в его экземплярном блоке данных. Сетевые данные окончательно записываются в глобальный блок данных; они копируются в почтовый ящик передачи (Send) с помощью системной функции BLKMOV. Мы используем функцию опроса времени суток для считывания времени из таймера реального времени CPU.

Первая сеть (network) в функциональном блоке FB *Generate_Frame* передает ID, сохраненный в экземпляре блока данных, в заголовок фрейма. Последовательный номер увеличивается на 1 и также пересылается в заголовок фрейма.

Вторая сеть содержит вызов функции *READ_CLK*, который считывает время суток из таймера реального времени и вводит его в заголовок фрейма в формате TIME_OF_DAY.

В последующих сетях вы увидите метод копирования выбранных переменных в ходе выполнения программы при помощи системной функции SFC 20 BLKMOV и без использования косвенной адресации. Таким образом, нет необходимости знать абсолютный адрес или структуру переменных. Принцип чрезвычайно прост: требуемая функция выбирается с использованием функций сравнения. В качестве критерия выбора допустимы номера с 1 по 14.

FB *Generate_Frame* программируется таким образом, что он вызывается для генерирования фрейма сообщения посредством фронта сигнала.

Косвенное копирование области данных

Пример показывает редактирование и использование «переменного» ANY-указателя с графическими программными элементами.

Функция *I_Copy* копирует область данных, требуемые адрес и размер которой вы можете установить через параметры блока. Отдельные параметры блока соответствуют отдельным элементам ANY-указателя (обратитесь к параграфу 24.2.4 «ANY-указатель»). Указанные в параметрах блока значения должны быть действительными; они не проверяются (SFC 20 BLKMOV сообщает об ошибке копирования в своем параметре значения функции, который пересылается в параметр значения функции *I_Copy*).

Существенными элементами являются две временных переменных *SoPointer* и *DesPointer* типа данных ANY. Они содержат ANY-указатели для системной функции SFC 20 BLKMOV. *SoPointer* указывает на исходную область данных, предназначенных для передачи, *DesPointer* ссылается на область назначения. На рисунке 24.3 показана структура переменной *SoPointer*; *DesPointer* имеет такую же структуру. Доступ к отдельным байтам, словам и двойным словам переменных ANY осуществляется через их абсолютные адреса.

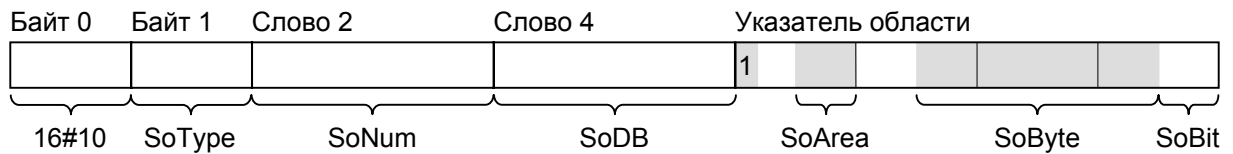


Рисунок 24.3 Структура переменной *SoPointer*

Сохранение фрейма сообщения

Пример демонстрирует использование функции *I_Copy* (копирование области данных с программируемым адресом).

Фрейм сообщения в блоке данных *Rec_Mailb* должен быть записан в следующую ячейку блока данных *Buffer*. Локальная блочная переменная *Entry* определяет местоположение в кольцевом буфере; значение этой ячейки используется для вычисления адреса в кольцевом буфере.

Переменная *Entry* имеет значение из области от 0 до 7. В первой сети компаратор определяет, *Entry* меньше 7 или нет. Если меньше, то *Entry* инкрементируется на 1 в следующей сети, иначе она устанавливается в 0. *Entry*, умноженная на 16, дает абсолютный байтовый адрес следующей записи данных в кольцевом буфере (структура данных *Message frame* состоит из 16 байтов).

Функция *I_Copy*, которая копирует фрейм сообщения из почтового ящика приема (Receive) (блок данных DB 62) в кольцевой буфер (блок данных DB 63), вызывается в сети 3 (Network 3).

Содержание главы 25

<u>25</u>	<u>Библиотеки блоков</u>	4
<u>25.1</u>	<u>Организационные блоки</u>	4
<u>25.2</u>	<u>Системные функциональные блоки</u>	6
<u>25.3</u>	<u>Функциональные блоки ИЕС</u>	11
<u>25.4</u>	<u>Блоки преобразования S5-S7</u>	13
<u>25.5</u>	<u>Блоки преобразования TI-S7</u>	16
<u>25.6</u>	<u>Блоки PID-управления</u>	17
<u>25.7</u>	<u>Коммуникационные блоки</u>	17

25 Библиотеки блоков

Базовое программное обеспечение STEP 7 включает в свой состав стандартную библиотеку *Standard Library*, которая содержит следующие библиотечные программы:

- Организационные блоки (Organization Blocks);
- Системные функциональные блоки (System Function Blocks);
- Функциональные блоки IEC (IEC Function Blocks);
- Блоки преобразования S5-S7 (S5-S7 Converting Blocks);
- Блоки преобразования TI-S7 (TI-S7 Converting Blocks);
- Блоки PID-управления (PID Control Blocks);
- Коммуникационные блоки (Communication Blocks).

Из указанных библиотечных программ вы можете копировать блоки и описания интерфейсов в собственные проекты.

25.1 Организационные блоки

(Пр = приоритетный класс по умолчанию)

ОВ	Пр	Назначение
1	1	Главная программа
10	2	Прерывание по времени суток 0
11	2	Прерывание по времени суток 1
12	2	Прерывание по времени суток 2
13	2	Прерывание по времени суток 3
14	2	Прерывание по времени суток 4
15	2	Прерывание по времени суток 5
16	2	Прерывание по времени суток 6
17	2	Прерывание по времени суток 7
20	3	Прерывание задержки времени 0
21	4	Прерывание задержки времени 1
22	5	Прерывание задержки времени 2
23	6	Прерывание задержки времени 3
30	7	Циклическое прерывание 0 (5 с)
31	8	Циклическое прерывание 1 (2 с)

ОВ	Пр	Назначение
32	9	Циклическое прерывание 2 (1 с)
33	10	Циклическое прерывание 3 (500 мс)
34	11	Циклическое прерывание 4 (200 мс)
35	12	Циклическое прерывание 5 (100 мс)
36	13	Циклическое прерывание 6 (50 мс)
37	14	Циклическое прерывание 7 (20 мс)
38	15	Циклическое прерывание 8 (10 мс)
40	16	Аппаратное прерывание 0
41	17	Аппаратное прерывание 1
42	18	Аппаратное прерывание 2
43	19	Аппаратное прерывание 3
44	20	Аппаратное прерывание 4
45	21	Аппаратное прерывание 5
46	22	Аппаратное прерывание 6
47	23	Аппаратное прерывание 7
60	25	Мультипроцессорное прерывание
70	25	Ошибка резервирования входа/выхода ¹⁾
72	28	Ошибка резервирования CPU
73	25	Ошибка коммуникационного резервирования
80	26	Временная ошибка ¹⁾
81	26	Сбой электропитания ¹⁾
82	26	Диагностическое прерывание ¹⁾
83	26	Прерывание вставки/удаления модуля ¹⁾
84	26	Сбой оборудования CPU ¹⁾
85	26	Ошибка приоритетного класса ¹⁾
86	26	Ошибка DP ¹⁾
87	26	Коммуникационная ошибка ¹⁾
90	29	Фоновая обработка
100	27	Полный рестарт
101	27	Рестарт
102	27	«Холодный» рестарт
121	-	Программная ошибка
122	-	Ошибка доступа к входу/выходу

¹⁾ Пр = 28 при рестарте

25.2 Системные функциональные блоки

IES-таймеры и IES-счетчики

SFB	Имя	Назначение
0	CTU	Счетчик прямого действия (по возрастанию)
1	CTD	Счетчик обратного действия (по убыванию)
2	CTUD	Счетчик прямого/обратного действия
3	TP	Импульсный
4	TON	Задержка включения
5	TOF	Задержка выключения

Коммуникация через сконфигурированные соединения

SFB	Имя	Назначение
8	USEND	Некоординированная передача
9	URVC	Некоординированный прием
12	BSEND	Ориентированная на блок передача
13	BRCV	Ориентированный на блок прием
14	GET	Чтение данных партнера
15	PUT	Запись данных в партнера
16	PRINT	Передача данных на принтер
19	START	Инициация полного рестарта партнера
20	STOP	Перевод партнера в состояние STOP
21	RESUME	Инициация рестарта партнера
22	STATUS	Опрос состояния партнера
23	USTATUS	Получение данных о состоянии партнера
SFC	Имя	Назначение
62	CONTROL	Опрос состояния коммуникации

Встроенные функции CPU 312/314/614

SFB	Имя	Назначение
29	HS_COUNT	Высокоскоростной счетчик
30	FREQ_MES	Частотомер
38	HSC_A_B	Элемент управления «Счетчик А/В»
39	POS	Элемент управления «Позиционирование»
41	CONT_C	Элемент непрерывного циклического управления
42	CONT_S	Элемент управления пошаговыми действиями
43	PULSEGEN	Генерирование импульсов

SFC	Имя	Назначение
63	AB_CALL	Вызов ассемблерного блока

Системная диагностика

SFC	Имя	Назначение
6	RD_SINFO	Чтение стартовой информации
51	RDSYSST	Чтение подписки SYS ST
52	WR_USMSG	Ввод в диагностический буфер

Создание сообщений, относящихся к блоку

SFB	Имя	Назначение
33	ALARM	Сообщения с окном подтверждения
34	ALARM_8	Сообщения без сопутствующих значений
35	ALARM_8P	Сообщения с сопутствующими значениями
36	NOTIFY	Сообщения с окном подтверждения
37	AR_SEND	Передача архивных данных

SFC	Имя	Назначение
9	EN_MSG	Разрешение сообщений
10	DIS_MSG	Запрет сообщений
17	ALARM_SQ	Сообщения, которые могут быть подтверждены
18	ALARM_S	Сообщения, которые всегда подтверждаются
19	ALARM_SC	Определение состояния подтверждения

Внутренний таймер CPU и счетчик рабочего времени

SFC	Имя	Назначение
0	SET_CLK	Установка таймера
1	READ_CLK	Чтение таймера
2	SET_RTM	Установка счетчика рабочего времени
3	CTRL_RTM	Модифицирование счетчика рабочего времени
4	READ_RTM	Чтение счетчика рабочего времени
48	SNC_RTCB	Синхронизация таймеров ведомых
64	TIME_TCK	Чтение системного времени

Барабан

SFC	Имя	Назначение
32	DRUM	Барабан

Копирование и функции для работы с блоками

SFC	Имя	Назначение
20	BLKMOV	Копирование области данных
21	FILL	Предварительное назначение области данных
22	CREAT_DB	Генерирование блока данных
23	DEL_DB	Удаление блока данных
24	TEST_DB	Тестирование блока данных
25	COMPRESS	Сжатие памяти
44	REPL_VAL	Ввод заменяющего значения
81	UBLKMOV	Копирование области данных без промежутков

Адресация модулей

SFC	Имя	Назначение
5	GADR_LGC	Определение логического адреса
49	LGC_GADR	Определение слота
50	RD_LGADR	Определение всех логических адресов

Распределенные входы/выходы

SFC	Имя	Назначение
7	DP_PRAL	Инициация аппаратного прерывания
11	DPSYN_FR	SYNC/FRRZE
12	D_ACT_DP	Деактивирование или активирование DP-ведомого
13	DPNRM_DG	Чтение диагностических данных
14	DPRD_DAT	Чтение данных ведомых
15	DPWR_DAT	Запись данных ведомых

Программное управление

SFC	Имя	Назначение
43	RE_TRIGR	Перезапуск монитор времени цикла
46	STP	Переход в режим STOP
47	WAIT	Ожидание в течение времени задержки

Передача записи данных

SFC	Имя	Назначение
54	RD_DPARM	Чтение предопределенного параметра
55	WR_PARM	Запись динамического параметра
56	WR_DPARM	Запись предопределенного параметра
57	PARM_MOD	Параметризация модуля
58	WR_REC	Запись элемента данных
59	RD_REC	Чтение записи данных

Обновление образа процесса

SFC	Имя	Назначение
26	UPDAT_PI	Обновление входной таблицы образа процесса
27	UPDAT_PO	Обновление выходной таблицы образа процесса
79	SET	Установка битового поля входа/выхода
80	RSET	Сброс битового поля входа/выхода

События прерываний

SFC	Имя	Назначение
28	SET_TINT	Установка прерывания по времени суток
29	CAN_TINT	Отмена прерывания по времени суток
30	ACT_TINT	Активация прерывания по времени суток
31	QRY_TINT	Запрос прерывания по времени суток
32	SRT_DINT	Запуск прерывания задержки времени
33	CAN_DINT	Отмена прерывания задержки времени
34	QRY_DINT	Запрос прерывания задержки времени
35	MP_ALM	Вызов мультипроцессорного предупреждения
36	MSK_FLT	Маскирование синхронных ошибок
37	DMSK_FLT	Демаскирование синхронных ошибок
38	READ_ERR	Чтение регистра состояния события
39	DIR_IRT	Запрет асинхронных ошибок
40	EN_IRT	Разблокировка асинхронных ошибок
41	DIS_AIRT	Задержка асинхронных ошибок
42	EN_AIRT	Разрешение асинхронных ошибок

Коммуникация через неконфигурированные соединения

SFC	Имя	Назначение
65	X_SEND	Внешняя передача данных
66	X_RCV	Внешний прием данных
67	X_GET	Внешнее чтение данных
68	X_PUT	Внешняя запись данных
69	X_ABORT	Разрыв внешнего соединения
72	I_GET	Внутренне чтение данных
73	I_PUT	Внутренняя запись данных
74	I_ABORT	Разрыв внутреннего соединения

Коммуникация глобальных данных

SFC	Имя	Назначение
60	GD_SND	Передача GD-пакета
61	GD_RCV	Прием GD-пакета

H-CPU

SFC	Имя	Назначение
90	H_CTRL	Управление рабочими режимами в H-CPU

25.3 Функциональные блоки IEC

Операции сравнения

FC	Имя	Назначение
9	EQ_DT	Сравнение DT «равно»
28	NE_DT	Сравнение DT «неравно»
14	GT_DT	Сравнение DT «больше»
12	GE_DT	Сравнение DT «больше или равно»
23	LT_DT	Сравнение DT «меньше»
18	LE_DT	Сравнение DT «меньше или равно»
10	EQ_STRING	Сравнение STRING «равно»
29	NE_STRING	Сравнение STRING «неравно»
15	GT_STRING	Сравнение STRING «больше»
13	GE_STRING	Сравнение STRING «больше или равно»
24	LT_STRING	Сравнение STRING «меньше»
19	LE_STRING	Сравнение STRING «меньше или равно»

Функции даты и времени

FC	Имя	Назначение
3	D_TOD_DT	Комбинирование DATE и TOD в DT
6	DT_DATE	Получение DATE из DT
7	DT_DAY	Получение дня недели из DT
8	DT_TOD	Получение TOD из DT
33	S5TI_TIM	Преобразование S5TIME в TIME
40	TIM_S5TI	Преобразование TIME в S5TIME
1	AD_DT_TM	Сложение TIME с DT
35	SB_DT_TM	Вычитание TIME из DT
34	SB_DT_DT	Вычитание DT из DT

Математические функции

FC	Имя	Назначение
22	LIMIT	Ограничитель
25	MAX	Выбор максимума
27	MIN	Выбор минимума
26	SEL	Двоичный выбор

Функции для работы со строками

FC	Имя	Назначение
21	LEN	Длина STRING
20	LEFT	Левая часть STRING
32	RIGHT	Правая часть STRING
26	MID	Средняя часть STRING
2	CONCAT	Конкатенация STRING
17	INSERT	Вставка STRING
4	DELETE	Удаление STRING
31	REPLACE	Замена STRING
11	FIND	Поиск STRING
16	I_STRING	Преобразование INT в STRING
5	DI_STRING	Преобразование DINT в STRING
30	R_STRING	Преобразование REAL в STRING
38	STRING_I	Преобразование STRING в INT
37	STRING_DI	Преобразование STRING в DINT
39	STRING_R	Преобразование STRING в REAL

25.4 Блоки преобразования S5-S7

Арифметические операции над числами с плавающей точкой

FC	Имя	Назначение
61	GP_FPGP	Преобразование числа с фиксированной точкой в число с плавающей точкой
62	GP_GFPF	Преобразование числа с плавающей точкой в число с фиксированной точкой
63	GP_ADD	Сложение чисел с плавающей точкой
64	GP_SUB	Вычитание чисел с плавающей точкой
65	GP_MUL	Умножение чисел с плавающей точкой
66	GP_DIV	Деление чисел с плавающей точкой
67	GP_VGL	Сравнение чисел с плавающей точкой
68	GP_RAD	Извлечение квадратного корня из числа с плавающей точкой

Базовые функции

FC	Имя	Назначение
85	ADD_32	Сложение 32-битных чисел с фиксированной точкой
86	SUB_32	Вычитание 32-битных чисел с фиксированной точкой
87	MUL_32	Умножение 32-битных чисел с фиксированной точкой
88	DIV_32	Деление 32-битных чисел с фиксированной точкой
89	RAD_16	Извлечение квадратного корня из 16-битного числа с фиксированной точкой
90	REG_SCHB	Регистр побитного сдвига
91	REG_SCHW	Регистр сдвига слова
92	REG_FIFO	Буфер (FIFO)
93	REG_LIFO	Стек (LIFO)
94	DB_COPY1	Копирование области данных (прямое)
95	DB_COPY2	Копирование области данных (косвенное)
96	RETTEN	Сохранение сверхоперативной памяти (S5-155U)
97	LADEN	Загрузка сверхоперативной памяти (S5-155U)
98	COD_B8	Преобразование BCD – двоичное число, 8 разрядов
99	COD_32	Преобразование двоичное число – BCD, 8 разрядов

Встроенные функции

FC	Имя	Назначение
81	COD_B4	Преобразование BCD – двоичное число, 4 разряда
82	COD_16	Преобразование двоичное число – BCD, 4 разряда
83	MUL_16	Умножение 16-битных чисел с фиксированной точкой
84	DIV_16	Деление 16-битных чисел с фиксированной точкой

Сигнальные функции

FC	Имя	Назначение
69	MLD_TG	Генератор импульса таймера
70	MLD_TGZ	Генератор импульса таймера с функцией таймера
71	MLD_EZW	Одиночное мигание начального значения размером в слово / Initial value single blinking wordwise
72	MLD_EDW	Двойное мигание начального значения размером в слово / Initial value double blinking wordwise
73	MLD_SAMW	Групповой сигнал размером в слово
74	MLD_SAM	Групповой сигнал
75	MLD_EZ	Одиночное мигание начального значения / Initial value single blinking
76	MLD_ED	Двойное мигание начального значения / Initial value double blinking
77	MLD_EZWK	Меркер одиночного мигания начального значения (размером в слово) / Initial value single blinking (wordwise) memory bit
78	MLD_EZDK	Меркер двойного мигания начального значения (размером в слово) / Initial value double blinking (wordwise) memory bit
79	MLD_EZK	Меркер одиночного мигания начального значения / Initial value single blinking memory bit
80	MLD_EDK	Меркер двойного мигания начального значения / Initial value double blinking memory bit

Аналоговые функции

FC	Имя	Назначение
100	AE_460_1	Модуль аналогового входа 460
101	AI_460_2	Модуль аналогового входа 460
102	AI_463_1	Модуль аналогового входа 463
103	AE_463_2	Модуль аналогового входа 463
104	AE_464_1	Модуль аналогового входа 464
105	AE_464_2	Модуль аналогового входа 464
106	AE_466_1	Модуль аналогового входа 466
107	AE_466_2	Модуль аналогового входа 466
108	RLG_AA1	Модуль аналогового выхода
109	RLG_AA2	Модуль аналогового выхода
110	PER_ET1	Распределенный вход/выход ET 100
111	PER_ET2	Распределенный вход/выход ET 100

Математические функции

FC	Имя	Назначение
112	SINUS	Синус
113	COSINUS	Косинус
114	TANGENS	Тангенс
115	COTANG	Котангенс
116	ARCSIN	Арксинус
117	ARCCOS	Арккосинус
118	ARCTAN	Арктангенс
119	ARCCOT	Арккотангенс
120	LN_X	Натуральный логарифм
121	LG_X	Логарифм по основанию 10
122	B_LOG_X	Логарифм по любому основанию
123	E_H_N	Экспонента по основанию e
124	ZEHN_H_N	Экспонента по основанию 10
125	A2_H_A1	Экспонента по любому основанию

25.5 Блоки преобразования TI-S7

FB	Имя	Назначение
80	LEAD LAG	Алгоритм задержки сигнала (Lead/lag algorithm)
81	DCAT	Прерывание дискретного контрольного времени
82	MCAT	Прерывание времени управления мотором
83	IMC	Сравнение матрицы индексов
84	SMC	Матричный сканер
85	DRUM	Событие маскируемого барабана
86	PACK	Сбор/распределение табличных данных
FC		
	Имя	Назначение
80	TONR	Блокировка задержки включения
81	IBLKMOV	Косвенная передача области данных
82	RSET	Побитовый сброс образа процесса
83	SET	Побитовая установка образа процесса
84	ATT	Ввод значения в таблицу
85	FIFO	Вывод первого значения таблицы
86	TBL FIND	Поиск значения в таблице
87	LIFO	Вывод последнего значения таблицы
88	TBL	Выполнение операции с таблицей
89	TBL WRD	Копирование значения из таблицы
90	WSR	Сохранить элемент данных
91	WRD TBL	Объединить элемент таблицы
92	SHRB	Сдвиг бита в регистре побитового сдвига
93	SEG	Битовый шаблон для 7-сегментного отображения
94	ATH	Преобразование ASCII – шестнадцатеричная форма
95	HTA	Преобразование шестнадцатеричная форма – ASCII
96	ENCO	Наименьший значащий установленный бит
97	DECO	Установка бита в слове
98	BCDCPL	Генерирование дополнительного кода в десятичной системе
99	BITSUM	Подсчет установленных битов
100	RSETI	Побайтовый сброс PQ
101	SETI	Побайтовая установка PQ
102	DEV	Вычисление стандартного отклонения
103	CDT	Таблицы корреляционных данных
104	TBL TBL	Комбинирование таблиц
105	SCALE	Значение коэффициентов масштабирования
106	UNSCALE	Значение коэффициентов демасштабирования

25.6 Блоки PID-управления

FB	Имя	Назначение
41	CONT_C	Непрерывное управление
42	CONT_S	Пошаговое управление
43	PULGEN	Генерирование импульса

25.7 Коммуникационные блоки

FC	Имя	Назначение
1	DP_SEND	Передача данных
2	DP_RECV	Прием данных
3	DP_DIAG	Диагностика
4	DP_CTRL	Управление

Содержание главы 26

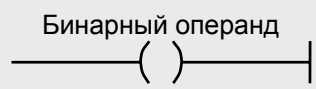
<u>26</u>	<u>Набор функций LAD</u>	4
<u>26.1</u>	<u>Базовые функции</u>	4
<u>26.2</u>	<u>Числовые функции</u>	7
<u>26.3</u>	<u>Управление программным потоком</u>	10

26 Набор функций LAD

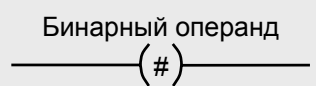
26.1 Базовые функции

Функции для работы с памятью

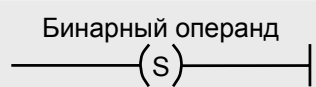
Одиночная катушка



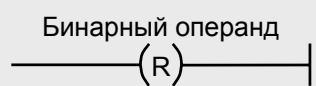
Коннектор



Катушка установки



Катушка сброса



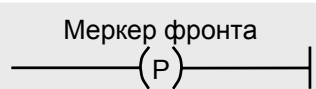
Блочный элемент SR



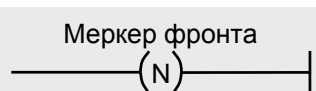
Блочный элемент RS



Положительный фронт
в электрическом токе



Отрицательный фронт
в электрическом токе



Положительный фронт
в операнде

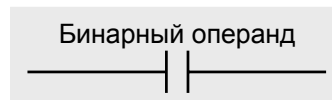


Отрицательный фронт
в операнде



Считывание и комбинирование бинарных значений

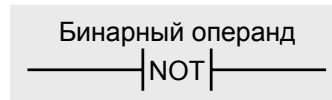
NO-контакт



NC-контакт



NOT-контакт



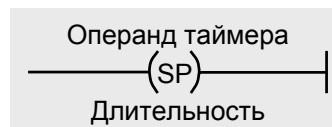
Функции таймера

Блочный элемент
таймера

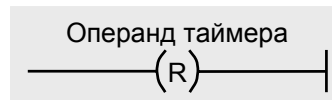


Отдельные элементы

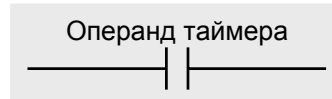
Катушка запуска с
временными характеристиками



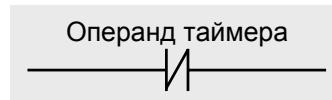
Катушка сброса



NO-контакт



NC-контакт

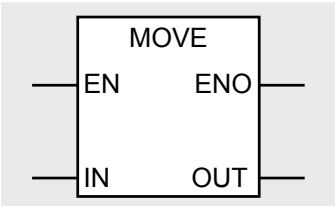


С характеристиками таймера:

S_PULSE	SP	Импульсный
S_PEXT	SE	Расширенный импульсный
S_ODT	SD	Задержка включения
S_ODTS	SS	Задержка включения с запоминанием
S_OFFDT	SF	Задержка выключения

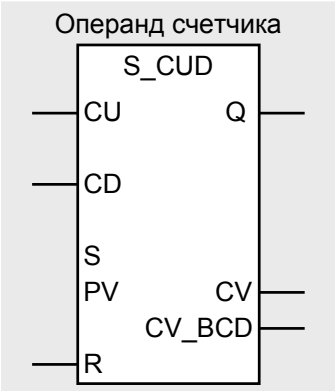
Функции перемещения

Блочный элемент MOVE



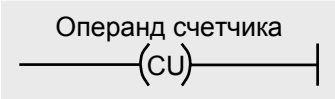
Функции счетчика

Блочный элемент счетчика

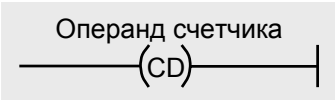


Отдельные элементы

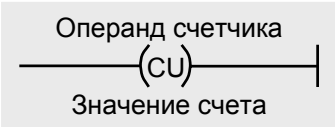
Катушка счета
по возрастанию



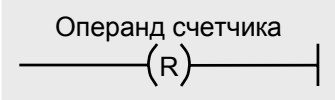
Катушка счета
по убыванию



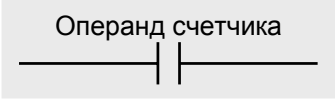
Катушка установки
со значением счета



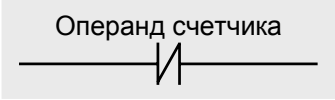
Катушка сброса



NO-контакт



NC-контакт

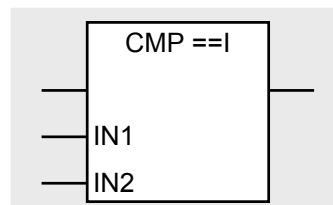


- С характеристиками счетчика:
- S_CUD Счетчик прямого/обратного счета
 - S_CU Счетчик прямого счета
 - S_CD Счетчик обратного счета

26.2 Числовые функции

Функции сравнения

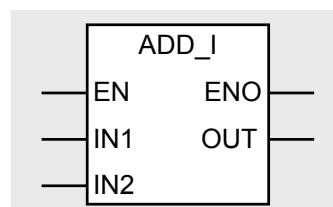
Блочный элемент
сравнения



Сравнение	В соответствии с		
	INT	DINT	REAL
«Равно»	==I	==D	==R
«Не равно»	<>I	<>D	<>R
«Больше»	>I	>D	>R
«Больше или равно»	>=I	>=D	>=R
«Меньше»	<I	<D	<R
«Меньше или равно»	<=I	<=D	<=R

Арифметические функции

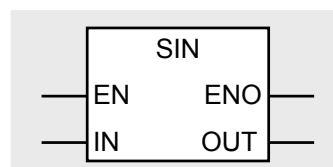
Арифметический
блочный элемент



Операция	В соответствии с		
	INT	DINT	REAL
Сложение	ADD_I	ADD_DI	ADD_R
Вычитание	SUB_I	SUB_DI	SUB_R
Умножение	MUL_I	MUL_DI	MUL_R
Деление с частным в качестве результата	DIV_I	DIV_DI	DIV_R
Деление с остатком в качестве результата	-	MOD_DI	-

Математические функции

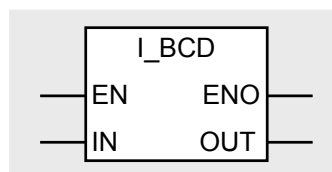
Математический
блочный элемент



SIN	Синус
COS	Косинус
TAN	Тангенс
ASIN	Арксинус
ACOS	Арккосинус
ATAN	Арктангенс
SQR	Возведение в квадрат
SQRT	Извлечение квадратного корня
EXP	Определение экспоненты
LN	Нахождение логарифма

Функции преобразования

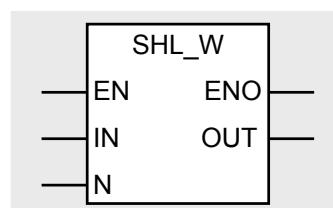
Блочный элемент
преобразования



I_DI	Преобразование INT в DINT
I_BCD	Преобразование INT в BCD
DI_BCD	Преобразование DINT в BCD
DI_R	Преобразование DINT в REAL
BCD_I	Преобразование BCD в INT
BCD_DI	Преобразование BCD в DINT
CEIL	Преобразование REAL в DINT с округлением до следующего большего числа
FLOOR	до следующего меньшего числа
ROUND	до следующего целого
TRUNC	без округления
INV_I	Обратный код INT
INV_DI	Обратный код DINT
NEG_I	Инвертирование INT
NEG_DI	Инвертирование DINT
NEG_R	Инвертирование REAL
ABS	Генерирование абсолютного значения REAL

Функции сдвига

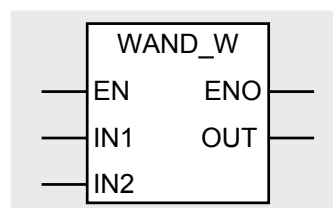
Блочный элемент
сдвига



SHL_W	Сдвиг слова влево
SHL_DW	Сдвиг двойного слова влево
SHR_W	Сдвиг слова вправо
SHR_DW	Сдвиг двойного слова вправо
SHR_I	Сдвиг слова со знаком
SHR_DI	Сдвиг двойного слова со знаком
ROL_DW	Циклический сдвиг влево
ROR_DW	Циклический сдвиг вправо

Побитовые логические операции

Блочный элемент
побитовой логической операции

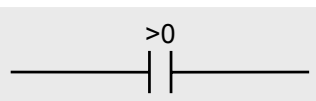


WAND_W	AND с переменными размером в слово
WOR_W	OR с переменными размером в слово
WXOR_W	Исключающее OR с переменными размером в слово
WAND_DW	AND с переменными размером в двойное слово
WOR_DW	OR с переменными размером в двойное слово
WXOR_DW	Исключающее OR с переменными размером в двойное слово

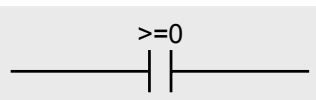
26.3 Управление программным потоком

Биты состояния

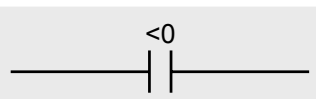
Результат больше
нуля



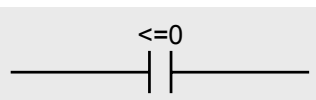
Результат больше или
равен нулю



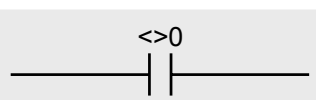
Результат меньше
нуля



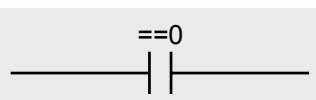
Результат меньше или
равен нулю



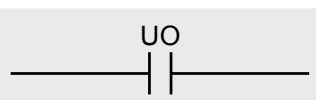
Результат не равен
нулю



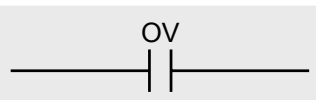
Результат равен
нулю



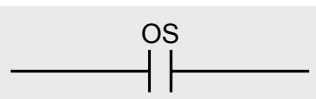
Результат недействителен
(неупорядочен)



Выход за границы области
значений числа (переполнение)



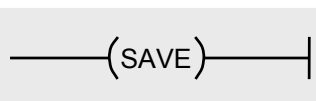
Сохраненное
переполнение



Бинарный
результат



Катушка SAVE
(сохранение)

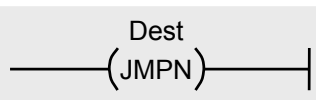


Функции перехода

Переход,
если RLO = «1»



Переход,
если RLO = «0»



Точка назначения
перехода



Главное реле управления (MCR)

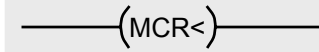
Активирование
MCR-области



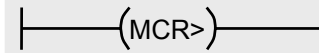
Деактивирование
MCR-области



Открытие
MCR-зоны

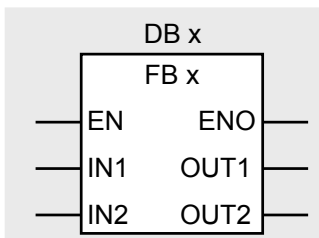


Закрытие
MCR-зоны

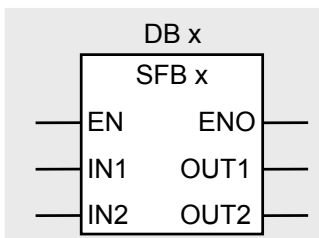


Функции для работы с блоками

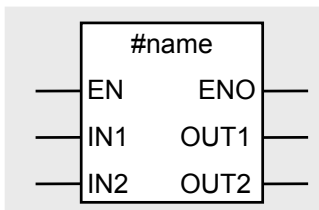
Вызов функционального блока
(с блоком данных)



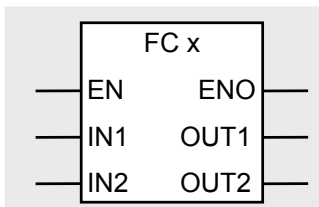
Вызов системного функционального
блока (с блоком данных)



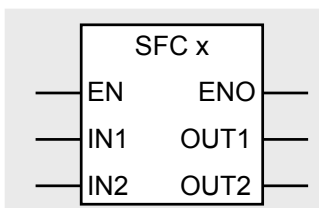
Вызов функционального блока или
системного функционального блока
(как локальный экземпляр)



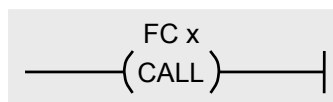
Вызов функции



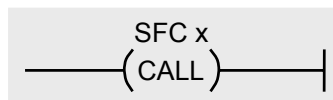
Вызов системной функции



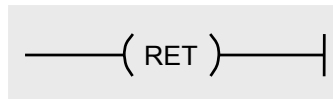
Вызов функции без
параметров



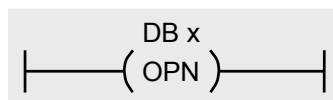
Вызов системной функции
без параметров



Катушка RET
(условное завершение блока)



Открытие блока
данных



Содержание главы 27

<u>27</u>	<u>Набор функций FBD</u>	4
<u>27.1</u>	<u>Базовые функции</u>	4
<u>27.2</u>	<u>Числовые функции</u>	7
<u>27.3</u>	<u>Управление программным потоком</u>	10

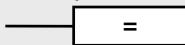
27 Набор функций FBD

27.1 Базовые функции

Функции для работы с памятью

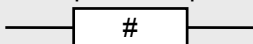
Назначение

Бинарный операнд



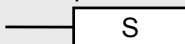
Коннектор

Бинарный операнд



Установка

Бинарный операнд



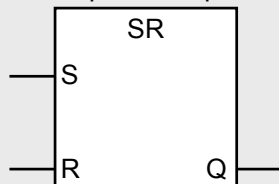
Сброс

Бинарный операнд



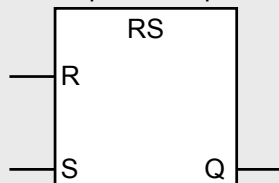
Блочный элемент SR

Бинарный операнд

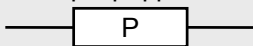


Блочный элемент RS

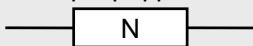
Бинарный операнд

Положительный фронт
результата RLO

Меркер фронта

Отрицательный фронт
результата RLO

Меркер фронта

Положительный фронт
в операнде

Бинарный операнд

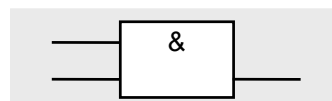
Отрицательный фронт
в операнде

Бинарный операнд

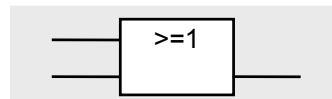


Считывание и комбинирование бинарных значений

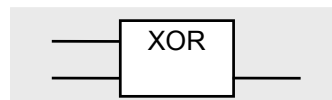
Функция AND



Функция OR



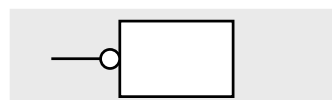
Функция исключающее OR



Считывание сигнального состояния «1»



Считывание сигнального состояния «0»



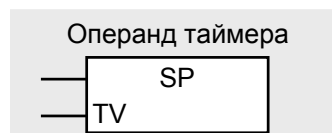
Функции таймера

Блочный элемент таймера

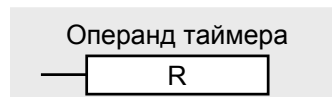


Отдельные элементы

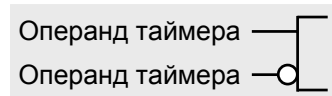
Блочный элемент запуска с характеристиками таймера



Блочный элемент сброса



Опрос состояния таймера

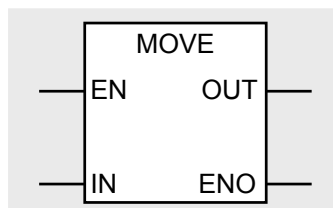


С характеристиками таймера:

S_PULSE	SP	Импульсный
S_PEXT	SE	Расширенный импульсный
S_ODT	SD	Задержка включения
S_ODTS	SS	Задержка включения с запоминанием
S_OFFDT	SF	Задержка выключения

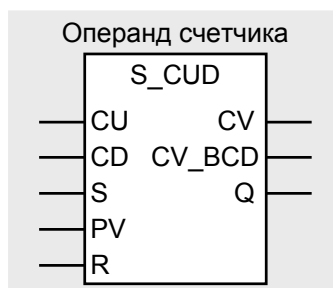
Функции перемещения

Блочный элемент MOVE



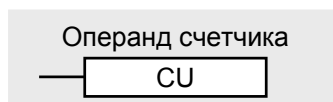
Функции счетчика

Блочный элемент счетчика

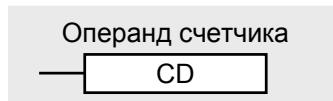


Отдельные элементы

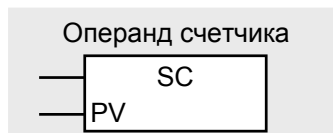
Блочный элемент счета
по возрастанию



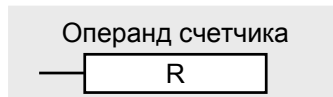
Блочный элемент счета
по убыванию



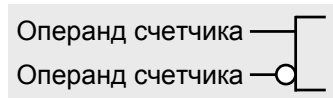
Блочный элемент установки
со значением счета



Блочный элемент сброса



Опрос состояния счетчика



С характеристиками счетчика:

S_CUD Счетчик прямого/обратного счета

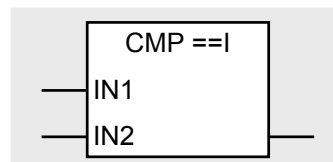
S_CU Счетчик прямого счета

S_CD Счетчик обратного счета

27.2 Числовые функции

Функции сравнения

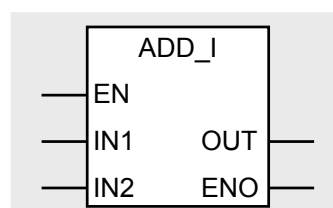
Блочный элемент
сравнения



Сравнение	В соответствии с		
	INT	DINT	REAL
«Равно»	==I	==D	==R
«Не равно»	<>I	<>D	<>R
«Больше»	>I	>D	>R
«Больше или равно»	>=I	>=D	>=R
«Меньше»	<I	<D	<R
«Меньше или равно»	<=I	<=D	<=R

Арифметические функции

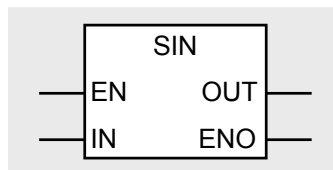
Арифметический
блочный элемент



Операция	В соответствии с		
	INT	DINT	REAL
Сложение	ADD_I	ADD_DI	ADD_R
Вычитание	SUB_I	SUB_DI	SUB_R
Умножение	MUL_I	MUL_DI	MUL_R
Деление с частным в качестве результата	DIV_I	DIV_DI	DIV_R
Деление с остатком в качестве результата	-	MOD_DI	-

Математические функции

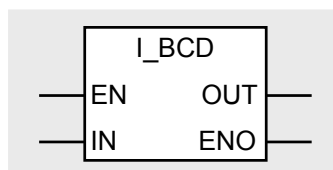
Математический
блочный элемент



SIN	Синус
COS	Косинус
TAN	Тангенс
ASIN	Арксинус
ACOS	Арккосинус
ATAN	Арктангенс
SQR	Возведение в квадрат
SQRT	Извлечение квадратного корня
EXP	Определение экспоненты
LN	Нахождение логарифма

Функции преобразования

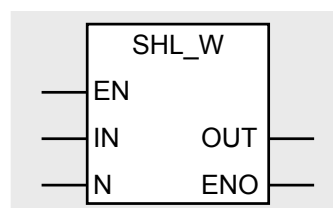
Блочный элемент
преобразования



I_DI	Преобразование INT в DINT
I_BCD	Преобразование INT в BCD
DI_BCD	Преобразование DINT в BCD
DI_R	Преобразование DINT в REAL
BCD_I	Преобразование BCD в INT
BCD_DI	Преобразование BCD в DINT
CEIL	Преобразование REAL в DINT с округлением до следующего большего числа
FLOOR	до следующего меньшего числа
ROUND	до следующего целого
TRUNC	без округления
INV_I	Обратный код INT
INV_DI	Обратный код DINT
NEG_I	Инвертирование INT
NEG_DI	Инвертирование DINT
NEG_R	Инвертирование REAL
ABS	Генерирование абсолютного значения REAL

Функции сдвига

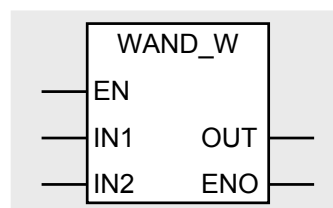
Блочный элемент
сдвига



SHL_W	Сдвиг слова влево
SHL_DW	Сдвиг двойного слова влево
SHR_W	Сдвиг слова вправо
SHR_DW	Сдвиг двойного слова вправо
SHR_I	Сдвиг слова со знаком
SHR_DI	Сдвиг двойного слова со знаком
ROL_DW	Циклический сдвиг влево
ROR_DW	Циклический сдвиг вправо

Побитовые логические операции

Блочный элемент
побитовой логической операции



WAND_W	AND с переменными размером в слово
WOR_W	OR с переменными размером в слово
WXOR_W	Исключающее OR с переменными размером в слово
WAND_DW	AND с переменными размером в двойное слово
WOR_DW	OR с переменными размером в двойное слово
WXOR_DW	Исключающее OR с переменными размером в двойное слово

27.3 Управление программным потоком

Биты состояния

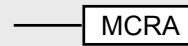
Результат больше нуля	
Результат больше или равен нулю	
Результат меньше нуля	
Результат меньше или равен нулю	
Результат не равен нулю	
Результат равен нулю	
Результат недействителен (неупорядочен)	
Выход за границы области значений числа (переполнение)	
Сохраненное переполнение	
Чтение бинарного результата BR	
Назначить бинарный результат BR	

Функции перехода

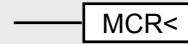
Переход, если RLO = «1»	
Переход, если RLO = «0»	
Точка назначения перехода	

Главное реле управления (MCR)

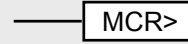
Активирование
MCR-области



Открытие
MCR-зоны



Закрытие
MCR-зоны

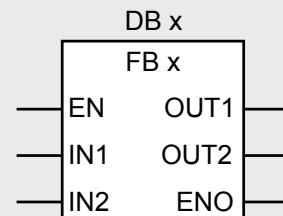


Деактивирование
MCR-области

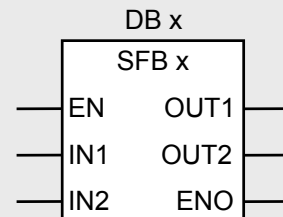


Функции для работы с блоками

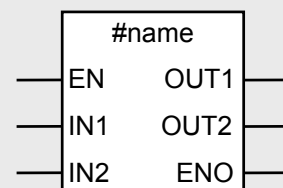
Вызов функционального блока
(с блоком данных)



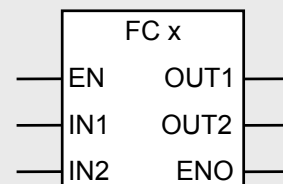
Вызов системного функционального
блока (с блоком данных)



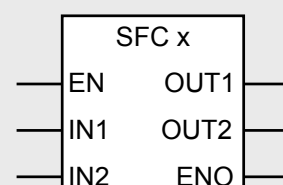
Вызов функционального блока или
системного функционального блока
(как локальный экземпляр)



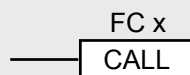
Вызов функции



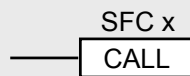
Вызов системной функции



Вызов функции без
параметров



Вызов системной функции
без параметров



Условное завершение
блока



Открытие блока
данных

